



**Faculty of Computer Science and Information Technology**

**A Framework for Parameter Optimization and Transfer Learning on  
Quartznet for Iban Automatic Speech Recognition**

**Steve Olsen Maikol @ Michael**

**Master of Science  
2024**

# A Framework for Parameter Optimization and Transfer Learning on Quartznet for Iban Automatic Speech Recognition

Steve Olsen Maikol @ Michael

A thesis submitted

In fulfillment of the requirements for the degree of Master of Science

(Computer Science)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2024

## DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



Signature

Name: Steve Olsen Maikol @ Michael

Matric No.: 21020006

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date : 25/3/2024

## **ACKNOWLEDGEMENT**

First of all, I extend my sincere gratitude to Dr Sarah Flora Samson Juan, my supervisor, for her invaluable support and guidance throughout my master's journey. Her expert direction in conducting experimental tasks, insightful advice in identifying potential solutions, and, above all, her unwavering assistance in completing the writing of this research thesis have been instrumental in my success.

In addition, I would like to express my heartfelt appreciation to Dr Edwin Mit, my co-supervisor, for his invaluable assistance and guidance in the publication of my research paper.

Furthermore, I am deeply grateful to my family and friends for their unwavering support, both morally and financially, throughout my academic journey. Their constant encouragement and belief in my abilities have been a tremendous source of strength and motivation. I am truly fortunate to have such loving and supportive people by my side.

My sincere gratitude also to the Centre for Graduate Studies, the management of the Universiti Malaysia Sarawak, and my sponsor, Biasiswa Kerajaan Negeri Sabah, for making it possible for me to complete my study here in Sarawak.

Last but not least, I wish to express my profound gratitude to the almighty God for bestowing upon me the energy, resilience, and unwavering spirit needed to overcome the challenges encountered during my master's journey. I am humbled by the divine guidance and blessings that have sustained me throughout this endeavour, and I am deeply thankful for the strength and inspiration received from above.

## ABSTRACT

The development of automatic speech recognition (ASR) systems for under-resourced languages poses challenges due to the lack of written resources required to train such systems. Traditionally, researchers have used language models to improve ASR model accuracy, some also resorts to the integration of pronunciation dictionaries, but these methods require abundance of written resources, which under-resourced languages often lack. The Iban language, spoken by the majority people of Sarawak in Malaysia, is an example of an under-resourced language for which previous attempts at developing an ASR system involved building a pronunciation dictionary and language model, transfer learning, and using DNN-HMM acoustic models. However, these methods proved challenging and costly. In this research, we propose a framework that uses a convolutional neural network (CNN) as an acoustic model to build an end-to-end ASR model for the Iban language. Three techniques are proposed to optimize the model without requiring additional data resources, including hyperparameter optimization, data augmentation and transfer learning. We report a significant reduction in word error rate (WER) in our experiments, demonstrating the effectiveness of our techniques. Overall, the proposed framework offers a promising approach for developing ASR systems for under-resourced languages that lack the necessary written resources for traditional methods.

**Keywords:** End-to-end, speech recognition, low-resource language, convolutional neural network, parameter optimization

## ***Rangka Kerja untuk Pengoptimuman Parameter dan Pembelajaran Pemindahan pada Quartznet untuk Pengecaman Pertuturan Automatik Iban***

### **ABSTRAK**

*Pembangunan sistem pengecaman pertuturan automatik (ASR) untuk bahasa bersumber rendah menimbulkan cabaran kerana kekurangan sumber bertulis yang diperlukan untuk melatih sistem tersebut. Secara tradisinya, penyelidik telah menggunakan kamus sebutan atau model bahasa untuk meningkatkan ketepatan model ASR, tetapi kaedah ini memerlukan banyak sumber bertulis yang sering bahasa sumber rendah tidak miliki. Bahasa Iban, yang digunakan oleh penduduk Sarawak di Malaysia adalah contoh bahasa sumber rendah yang mana percubaan sebelumnya untuk membangunkan sistem ASR melibatkan pembinaan kamus sebutan dan model bahasa, pemindahan pembelajaran, dan menggunakan model akustik DNN-HMM . Walau bagaimanapun, kaedah ini terbukti mencabar dan mahal. Dalam penyelidikan ini, kami mencadangkan rangka kerja yang menggunakan rangkaian neural konvolusi (CNN) sebagai model akustik untuk membina model ASR hujung ke hujung untuk bahasa Iban. Tiga teknik dicadangkan untuk mengoptimumkan model tanpa memerlukan sumber data tambahan, termasuk pengoptimuman hiperparameter, penambahan data dan pembelajaran pemindahan. Kami melaporkan pengurangan ketara dalam kadar ralat perkataan (WER) dalam eksperimen kami, menunjukkan keberkesanan teknik kami. Secara keseluruhan, rangka kerja yang dicadangkan menawarkan pendekatan yang menjanjikan untuk membangunkan sistem ASR untuk bahasa bersumber rendah yang tidak mempunyai sumber bertulis yang diperlukan untuk kaedah tradisional.*

**Kata kunci:** *Hujung ke hujung, pengecaman pertuturan, bahasa sumber rendah, rangkaian neural konvolusi, pengoptimuman parameter*

## **TABLE OF CONTENTS**

|                                                        | <b>Page</b> |
|--------------------------------------------------------|-------------|
| <b>DECLARATION</b>                                     | i           |
| <b>ACKNOWLEDGEMENT</b>                                 | ii          |
| <b>ABSTRACT</b>                                        | iii         |
| <b>ABSTRAK</b>                                         | iv          |
| <b>TABLE OF CONTENTS</b>                               | v           |
| <b>LIST OF TABLES</b>                                  | xii         |
| <b>LIST OF FIGURES</b>                                 | xiv         |
| <b>LIST OF ABBREVIATIONS</b>                           | xvi         |
| <b>CHAPTER 1 INTRODUCTION</b>                          | <b>1</b>    |
| 1.1 Background                                         | 1           |
| 1.2 Convolutional Neural Network in Speech Recognition | 1           |
| 1.3 Problem Statement                                  | 2           |
| 1.4 Research Questions and Objectives                  | 3           |
| 1.5 Scope of Research                                  | 3           |
| 1.6 Significance of Study                              | 4           |
| 1.7 Research Outlines                                  | 5           |
| <b>CHAPTER 2 LITERATURE REVIEW</b>                     | <b>7</b>    |
| 2.1 Introduction                                       | 7           |

|       |                                                                       |    |
|-------|-----------------------------------------------------------------------|----|
| 2.2   | Automatic Speech Recognition (ASR) and Under-Resourced Languages      | 7  |
| 2.3   | Neural Network Acoustic Model                                         | 10 |
| 2.4   | Convolutional Neural Network (CNN) as an Acoustic Model               | 11 |
| 2.4.1 | CNN in speech technology                                              | 11 |
| 2.4.2 | Under-resourced language - The Iban language                          | 13 |
| 2.5   | The previous Iban ASR system with Deep Neural Network acoustic model. | 14 |
| 2.6   | Architecture of CNN                                                   | 15 |
| 2.6.1 | Learning Process in CNN                                               | 16 |
| 2.6.2 | Traditional VS End-to-end CNN pipeline                                | 21 |
| 2.6.3 | Quartznet: an end-to-end CNN                                          | 22 |
| 2.6.4 | Epoch, block, and sub-block of Quartznet                              | 24 |
| 2.7   | Existing Works on Improving ASR Model Performance                     | 24 |
| 2.7.1 | Hyperparameter Optimization                                           | 26 |
| 2.7.2 | Spectrogram Augmentation                                              | 29 |
| 2.7.3 | Cross-lingual Transfer Learning Method                                | 31 |
| 2.8   | Available Tools for CNN Modelling                                     | 35 |
| 2.8.1 | Kaldi                                                                 | 35 |
| 2.8.2 | CMUSphinx                                                             | 36 |
| 2.8.3 | Nvidia NeMo                                                           | 37 |
| 2.8.4 | Google Colaboratory                                                   | 38 |

|                              |                                                                    |           |
|------------------------------|--------------------------------------------------------------------|-----------|
| 2.9                          | Study gap on the previous existing Iban ASR model                  | 39        |
| 2.10                         | Chapter Summary                                                    | 40        |
| <b>CHAPTER 3 METHODOLOGY</b> |                                                                    | <b>41</b> |
| 3.1                          | Introduction                                                       | 41        |
| 3.2                          | Dataset                                                            | 41        |
| 3.2.1                        | Iban (Target Language)                                             | 41        |
| 3.2.2                        | Malay                                                              | 42        |
| 3.2.3                        | English Pre-trained Model                                          | 43        |
| 3.3                          | General Experimental Setup                                         | 43        |
| 3.3.1                        | Preparation of Dataset                                             | 43        |
| 3.4                          | Framework to improve CNN Iban ASR through optimization approaches. | 45        |
| 3.4.1                        | Monolingual Approach                                               | 46        |
| 3.4.2                        | Cross-lingual Approach                                             | 47        |
| 3.4.3                        | Speech Recognition Engine and Tools                                | 49        |
| 3.5                          | Evaluation Metrics of ASR models                                   | 51        |
| 3.5.1                        | Performance Measurement Tools                                      | 52        |
| 3.6                          | Preliminary Experimental Setup                                     | 52        |
| 3.6.1                        | Preliminary experiment - Experiment I                              | 53        |
| 3.6.2                        | Preliminary experiment – Experiment II                             | 55        |
| 3.7                          | Optimizing Iban CNN Model – Proposed Techniques                    | 56        |

|                  |                                                             |           |
|------------------|-------------------------------------------------------------|-----------|
| 3.7.1            | Block Optimization                                          | 57        |
| 3.7.2            | Sub-block Optimization                                      | 58        |
| 3.7.3            | Epoch Optimization                                          | 60        |
| 3.7.4            | Spectrogram Augmentation                                    | 61        |
| 3.7.5            | Cross-lingual Transfer Learning Method                      | 65        |
| 3.8              | Experimental Steps with Figures                             | 72        |
| 3.8.1            | Training a model from scratch                               | 72        |
| 3.8.2            | Carrying out Transfer Learning                              | 79        |
| 3.8.3            | Evaluate model accuracy                                     | 82        |
| 3.9              | YAML File                                                   | 84        |
| 3.10             | Chapter Summary                                             | 86        |
| <b>CHAPTER 4</b> | <b>RESULTS, ANALYSIS AND DISCUSSION</b>                     | <b>87</b> |
| 4.1              | Introduction                                                | 87        |
| 4.2              | Preliminary Experiment Result                               | 87        |
| 4.2.1            | Experiment I Results                                        | 87        |
| 4.2.2            | Experiment II results                                       | 89        |
| 4.2.3            | Comparison of Transcription                                 | 91        |
| 4.3              | Hyperparameter Optimization of Block, Sub-block, and Epoch. | 92        |
| 4.3.1            | Block                                                       | 93        |
| 4.3.2            | Sub-block                                                   | 94        |

|       |                                                                 |     |
|-------|-----------------------------------------------------------------|-----|
| 4.3.3 | Epoch                                                           | 95  |
| 4.3.4 | Comparison between Epoch, Block, and Sub-block                  | 97  |
| 4.3.5 | Technique #1 Comparison of Reference and Hypothesis             | 100 |
| 4.4   | Spectrogram Augmentation                                        | 102 |
| 4.4.1 | Rectangle Masking                                               | 102 |
| 4.4.2 | Time-Frequency Masking                                          | 104 |
| 4.4.3 | Comparison between Rectangle and Time Frequency Masking         | 106 |
| 4.4.4 | Technique #2 Comparison of Reference and Hypothesis             | 107 |
| 4.5   | Cross-lingual Transfer Learning Method                          | 108 |
| 4.5.1 | Basic TL                                                        | 109 |
| 4.5.2 | TL with Spectrogram Augmentation                                | 110 |
| 4.5.3 | TL with Optimized Betas                                         | 116 |
| 4.5.4 | Findings from the implementation of the three TL settings       | 119 |
| 4.5.5 | Technique #3 Comparison of Reference and Hypothesis             | 121 |
| 4.5.6 | TL with Encoder Freezing                                        | 124 |
| 4.6   | Results after Implementing the Framework                        | 125 |
| 4.7   | Comparison with previous Iban ASR model reported in literature. | 127 |
| 4.7.1 | Comparison of Monolingual model                                 | 128 |
| 4.7.2 | Comparison of Cross-lingual model (Transfer Learning)           | 131 |

|                                              |                                                                                             |            |
|----------------------------------------------|---------------------------------------------------------------------------------------------|------------|
| 4.8                                          | Discussion on using English and Malay language as source language for Transfer Learning     | 132        |
| 4.8.1                                        | Performance of the English language when used as Iban ASR Source Language in our framework. | 133        |
| 4.9                                          | Discussion on CNN as an acoustic model for an end-to-end ASR for under-resourced language   | 135        |
| 4.9.1                                        | Advantages of the CNN in modelling Iban ASR                                                 | 136        |
| 4.9.2                                        | Disadvantages of the CNN in modelling an Iban ASR                                           | 138        |
| 4.10                                         | Chapter Summary                                                                             | 140        |
| <b>CHAPTER 5 CONCLUSION AND FUTURE WORKS</b> |                                                                                             | <b>141</b> |
| 5.1                                          | Introduction                                                                                | 141        |
| 5.2                                          | Summary of Research                                                                         | 141        |
| 5.3                                          | Research Contribution                                                                       | 144        |
| 5.3.1                                        | CNN ASR framework for Iban and under-resourced languages                                    | 144        |
| 5.3.2                                        | A CNN Iban ASR performance benchmark                                                        | 144        |
| 5.3.3                                        | Framework to optimize block, sub-block, and epoch of Quartznet CNN                          | 144        |
| 5.3.4                                        | Studies on how Spectrogram Augmentation benefits the CNN in modelling an Iban ASR.          | 144        |
| 5.3.5                                        | Studies on Transfer Learning in modelling a CNN Iban ASR                                    | 145        |
| 5.3.6                                        | Production of the first CNN Iban ASR model                                                  | 145        |
| 5.4                                          | Research Limitation                                                                         | 146        |

|     |                   |     |
|-----|-------------------|-----|
| 5.5 | Future Work       | 147 |
|     | <b>REFERENCES</b> | 149 |
|     | <b>APPENDICES</b> | 159 |

## LIST OF TABLES

|                                                                                                               | <b>Page</b> |
|---------------------------------------------------------------------------------------------------------------|-------------|
| Table 2.1: Contribution of Transfer Learning technique to various research                                    | 33          |
| Table 3.1: Distribution of Train and Test set for the Iban and Malay Dataset                                  | 44          |
| Table 3.2: Protocol of Experiment I -Train Set                                                                | 53          |
| Table 3.3: Preliminary Experiment I Model Architecture                                                        | 54          |
| Table 3.4: Protocol of Experiment I – Test Set                                                                | 55          |
| Table 3.5: Preliminary Experiment II Model Architecture                                                       | 56          |
| Table 3.6: Block Experiment’s Model Architecture                                                              | 57          |
| Table 3.7: Sub-block Experiment’s Model Architecture                                                          | 59          |
| Table 3.8: Epoch Experiment’s Model Architecture                                                              | 60          |
| Table 3.9: Rectangle Masking Experiment's Model Architecture                                                  | 63          |
| Table 3.10: Hyperparameter values set in the rectangle masking experiment                                     | 63          |
| Table 3.11: Time-Frequency Masking Experiment' Model Architecture                                             | 64          |
| Table 3.12: Hyperparameter values set in the time-frequency masking experiment                                | 64          |
| Table 3.13: Basic TL’s Model Architecture                                                                     | 67          |
| Table 3.14: Hyperparameter values set for each setting in TL with spectrogram augmentation experiment         | 68          |
| Table 3.15: TL with Spectrogram Augmentation's Model Architecture                                             | 69          |
| Table 4.1: Results of Experiment I for test set increase                                                      | 88          |
| Table 4.2: Comparison of output with the original transcript at epoch 5500                                    | 91          |
| Table 4.3: Rate of WER reduction produced after applying the different hyperparameter values (increment mode) | 98          |
| Table 4.4: Comparison of hypothesis produced by the models from the three hyperparameter experiments          | 100         |
| Table 4.5: WER differences of model with rectangle masking and model without rectangle masking                | 103         |

|             |                                                                                                   |     |
|-------------|---------------------------------------------------------------------------------------------------|-----|
| Table 4.6:  | WERs for Iban CNN models with time-frequency masking and model without time-frequency masking     | 105 |
| Table 4.7:  | Summary of lowest WER for Technique 2                                                             | 106 |
| Table 4.8:  | Comparison of hypothesis produced by the models from the two spectrogram augmentation experiments | 107 |
| Table 4.9:  | WER results obtained from the Basic TL experiment                                                 | 109 |
| Table 4.10: | WER differences between TL model against our optimized model                                      | 109 |
| Table 4.11: | Results of STTQznet model WER for TL with SA setting                                              | 110 |
| Table 4.12: | Result of BM model WER for TL with SA setting                                                     | 113 |
| Table 4.13: | Result of STTQznet model WER for Min to Max to Min run                                            | 116 |
| Table 4.14: | Result of STTQznet model WER for Median run                                                       | 117 |
| Table 4.15: | Result of BM model for TL with optimized betas experiment                                         | 119 |
| Table 4.16: | Summary of result for the lowest WER obtained from technique 3 experiment.                        | 120 |
| Table 4.17: | Comparison of transcription produced by the models from the TL experiments                        | 121 |
| Table 4.18: | Comparison of WER obtained by or model when implementing TL with encoder freezing                 | 125 |
| Table 4.19: | Improvement of WER throughout the implementation of techniques following our proposed framework   | 126 |
| Table 4.20: | WER of English $\times$ Malay TL model on the Malay test set                                      | 133 |
| Table 4.21: | WER of English $\times$ Malay $\times$ Iban TL model on the Iban test set                         | 134 |

## LIST OF FIGURES

|                                                                                       | <b>Page</b> |
|---------------------------------------------------------------------------------------|-------------|
| Figure 2.1: Pipeline of a Traditional ASR                                             | 9           |
| Figure 2.2: Original 5x5 Matrix                                                       | 16          |
| Figure 2.3: 3x3 Kernel Matrix                                                         | 17          |
| Figure 2.4: Convolution process in CNN                                                | 17          |
| Figure 2.5: Max pooling of a rectified feature map                                    | 20          |
| Figure 2.6: Traditional Vs End-to-end ASR development pipeline                        | 21          |
| Figure 2.7: Quartznet 4x1 Architecture                                                | 23          |
| Figure 3.1: Iban ASR Framework                                                        | 45          |
| Figure 3.2: Quartznet 5x1 from $B \times 1$ architecture                              | 58          |
| Figure 3.3: Quartznet 1x9 from $1 \times R$ architecture                              | 59          |
| Figure 3.4: The optimized $B \times R$ - Quartznet $20 \times 1$ Model Architecture   | 61          |
| Figure 3.5: Code snippet for the import of dependencies found in our notebook         | 72          |
| Figure 3.6: Code snippet for the import of necessary libraries used in our experiment | 73          |
| Figure 3.7: Code snippet for the import of librosa and specification of directory     | 73          |
| Figure 3.8: Code snippet for the import of Nvidia Nemo                                | 74          |
| Figure 3.9: Code snippet for building manifest file (STEP 1)                          | 75          |
| Figure 3.10: Code snippet to establish YAML configuration (STEP 2)                    | 75          |
| Figure 3.11: Code snippets to set up modules and manifest files                       | 76          |
| Figure 3.12: Code snippets to create a new model and setup its configurations.        | 77          |
| Figure 3.13: Code snippet to setup the experiment manager                             | 77          |
| Figure 3.14: Code snippet to begin model training                                     | 78          |
| Figure 3.15: Code snippet for model and checkpoint saving.                            | 78          |

|                                                                                                                                                                                                               |     |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Figure 3.16: Code snippet to load model and checkpoint                                                                                                                                                        | 79  |
| Figure 3.17: Code snippets for loading a pretrained model                                                                                                                                                     | 80  |
| Figure 3.18: Code snippets to setup the transfer learning data loaders                                                                                                                                        | 81  |
| Figure 3.19: Code snippets to setup the data loaders (continued) and to display the model configuration.                                                                                                      | 81  |
| Figure 3.20: Code snippet to setup the optimizer and its default values                                                                                                                                       | 82  |
| Figure 3.21: Code snippet to transcribe test set using current model                                                                                                                                          | 83  |
| Figure 3.22: Code snippet to import SCTK library                                                                                                                                                              | 84  |
| Figure 3.23: Code snippet to run SCTK and its instructions                                                                                                                                                    | 84  |
| Figure 4.1: Results of Experiment I for train set increase                                                                                                                                                    | 88  |
| Figure 4.2: Line graph of the preliminary experiment II results                                                                                                                                               | 90  |
| Figure 4.3: Dynamics of model performance in the Block experiment                                                                                                                                             | 93  |
| Figure 4.4: Iban CNN model performances in the Sub-block experiment                                                                                                                                           | 94  |
| Figure 4.5: Graph result of the Epoch experiment for the Quartznet 1×1 model                                                                                                                                  | 96  |
| Figure 4.6: Graph result of the Epoch experiment for the Quartznet 20×1 model                                                                                                                                 | 96  |
| Figure 4.7: Graph of Rate of WER reduction between the three hyperparameter in 9 increments.                                                                                                                  | 99  |
| Figure 4.8: Graph of rectangle masking experiment results                                                                                                                                                     | 102 |
| Figure 4.9: Graph result of time-frequency masking experiment                                                                                                                                                 | 104 |
| Figure 4.10: Graph of WER reduction produced by each spectrogram augmentation settings on the STTQznet model after transfer learning for 100 epochs                                                           | 112 |
| Figure 4.11: Graph of WER reduction produced by each spectrogram augmentation settings on the BM model after transfer learning for 100 epochs                                                                 | 115 |
| Figure 4.12: Chart of model performance (WER) of STTQznet model during TL with beta optimizations                                                                                                             | 118 |
| Figure 4.13: HMM-DNN model architecture as taken from the paper "Exploiting resources from closely-related languages for automatic speech recognition in low-resource languages from Malaysia" by Juan (2015) | 129 |
| Figure 4.14: Example of an error output from our best model                                                                                                                                                   | 139 |

## LIST OF ABBREVIATIONS

|        |                                                    |
|--------|----------------------------------------------------|
| AM     | Acoustic Model                                     |
| ANN    | Artificial Neural Network                          |
| API    | Application Programming Interface                  |
| ASR    | Automatic Speech Recognition                       |
| CEL    | Convolutional Embedding Layer                      |
| CER    | Character Error Rate                               |
| CNN    | Convolutional Neural Network                       |
| CTC    | Connectionist Temporal Classification              |
| DNN    | Deep Neural Network                                |
| DTW    | Dynamic Time Warping                               |
| FedAvg | Federated Averaging Algorithm                      |
| fMLLR  | Feature-Space Maximum Likelihood Linear Regression |
| G2P    | Grapheme to Phoneme                                |
| GMM    | Gaussian Mixture Model                             |
| GPU    | Graphical Processing Unit                          |
| HMM    | Hidden Markov Model                                |
| IPA    | International Phonetic Alphabet                    |
| LDA    | Linear Discriminant Analysis                       |
| LM     | Language Model                                     |
| LSTM   | Long Short-Term Memory                             |
| LVCSR  | Large Vocabulary Continuous Speech Recognition     |
| MFCC   | Mel-frequency cepstral coefficients                |

|          |                                     |
|----------|-------------------------------------|
| MLLT     | Maximum Likelihood Linear Transform |
| MLT      | Multi-task Learning                 |
| NGO      | Northern Goshawk Optimization       |
| NLP      | Natural Language Processing         |
| PD       | Pronunciation Dictionary            |
| PED      | Pronunciation Error Detection (PED) |
| ReLU     | Rectified Linear Unit               |
| RNN      | Recurrent Neural Network            |
| SA       | Spectrogram Augmentation            |
| SCTK     | Speech Recognition Scoring Toolkit  |
| STT      | Speech-to-Text                      |
| STTQznet | Speech-to-Text Quartznet            |
| TDNN     | Time Delay Neural Network           |
| TF       | Time-Frequency                      |
| TL       | Transfer Learning                   |
| TTS      | Text-to-Speech                      |
| WER      | Word-Error-Rate                     |
| WRS      | Weighted Random Search              |
| WSJ      | World Street Journal dataset        |

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 Background**

The development of Automatic Speech Recognition (ASR) system has been trending in these recent years and it has been implemented in many software and applications such as Google Assistant and Amazon Siri, whereby this system receives our audio speech and translate it into text data for the system to recognize as input before handling out its programmed tasks. However, building an ASR for under-resourced language is still a challenge as under-resourced languages suffers the issue of data scarcity, causing the ASR developed to have low prediction accuracy due to insufficient training data. The Iban language is an under-resourced language spoken mainly in Sarawak, Malaysia and West Kalimantan, Indonesia (Aman et al., 2019). The local people of Sarawak use Iban a lot in terms of daily communication, however, written data on the language are lacking. To this day, the most prominent work in the development of Iban ASR was done by Juan (2015), in which the author initiated the development of the very first Iban ASR using Deep Neural Network (DNN). Aside from the works done by Juan (2015), no other committed effort involving the Iban language in ASR development was done. Meanwhile, research gaps regarding the studies of Iban ASR development are still many.

#### **1.2 Convolutional Neural Network in Speech Recognition**

Typically, three components are required to build a statistical ASR, these are Acoustic Model, Pronunciation Dictionary and Language Model. Each of these three crucial components is required to be developed using and trained with abundance speech data to help an ASR model to achieve excellent prediction accuracy. This requirement, however, is

an issue for under-resourced language such as Iban as they do not possess enough language resources. CNN is a neural network that recently has present itself as a solution to overcome the issue of data scarcity faced by under-resourced languages in building ASR models by being its acoustic model (Arnel Fajardo, 2020; Lekshmi & Sherly, 2021; Thai et al., 2020). With end-to-end architecture and CNN excellent feature extraction capabilities, it helps researcher to exclude the necessity for the integration of pronunciation dictionary and language model into ASR system while still producing high accuracy predictions (Alsayadi et al., 2021; Parry et al., 2019; Yu et al., 2019; Zhang et al., 2021). However, no studies using CNN in the development of Iban ASR system has ever been conducted previously and no data discussing about its performance as an acoustic model for Iban ASR model, whether it is able to overcome Iban language's data scarcity, has ever been recorded yet. With that said, it serves as our motivation to conduct research on the CNN for Iban ASR to analyse its capability as an acoustic model for under-resourced language and to fill this study gap.

### **1.3 Problem Statement**

As mentioned previously, developing an ASR model using the end-to-end CNN architecture for the Iban language has never been conducted yet and it is known that under-resourced language suffers a lack of language resource in building a statistical ASR. The proper steps to build an Iban ASR using CNN as acoustic model while excluding the integration of pronunciation dictionary and language model in the system and still achieving excellent prediction accuracy has never been documented previously and no framework describing its process has ever been proposed. Furthermore, it was required that for a CNN ASR model to perform well, its network structure has to be optimized (Aszemi & Dominic, 2019; Xie & Yuille, 2017). Currently, there is no known systematic way of building an optimized CNN ASR model for the under-resourced Iban language. Investigating this

research gap would help us to identify a proper method to develop an end-to-end CNN Iban ASR with optimized model structure and propose its systematic framework which will act as a document for future references.

#### **1.4 Research Questions and Objectives**

In response to the problem statement described earlier, we have listed out several research questions and objectives as a guideline for the research to investigate the previously mentioned study gap. The details are as follows:

##### **Research Questions:**

1. How to obtain a CNN acoustic model for Iban ASR?
2. How to determine the parameters that can influence the performance of CNN acoustic modelling in Iban ASR?

##### **Research Objectives:**

- I. To study the general architecture of CNN acoustic modelling in ASR and its benefits for under-resourced language.
- II. To propose a CNN acoustic modelling framework for investigating the WER in Iban ASR.
- III. To investigate the WERs obtained by the CNN-based Iban ASR model through hyperparameter optimization, spectrogram augmentation, and transfer learning and analyse its performance.

#### **1.5 Scope of Research**

Two constraints have been defined to help us focus on our scope when carrying out experiments that covers a wide field of knowledge. Although this research aims to study the ways on how to improve ASR for under-resourced language, only the Iban language that

will be used as the target language in this research. The main corpus that is going to be used for the experiment will be the Iban corpus that was previously collected by (Juan, 2015). Other corpus may be imported but only for the sole purpose of improving the performance of the Iban ASR model (e.g., for transfer learning). Secondly, as the title of the research implies, studies on Neural Network model will be conducted only on the CNN. The scope of our research experiment will focus on the development of ASR model using end-to-end CNN model only and without the integration of pronunciation dictionary and the language model. Despite having to do comparison between CNN and other ANN models in the evaluation stage, thorough analysis and investigation will be done only for the CNN model. The focus of this research will follow these two rules; Iban language being the targeted under-resourced language and CNN being the only neural network model to be investigated, as its core to prevent straying away from the main purpose and the objectives of this research.

## **1.6 Significance of Study**

Through the conduct of this research, we will be able to contribute an improvement towards ASR advancement specifically, for the Iban language, generally, for under-resourced language. First of all, we would be able to document our setup on the CNN architecture that will be used and be presented generally as a reference for future under-resourced language research that wants to implement the same architecture. In addition, this research would be beneficial to researcher as its baseline results produced during the research experiments can be taken for the conduct of comparison between different variant of Iban CNN models in the future. By doing this research, we would be able to identify which hyperparameters in CNN that may affect the performance of an ASR in training the Iban language, beneficially and detrimentally. We would also be able to identify what may be the weakness of CNN as an acoustic model through the conduct of this research experiment.

Moreover, this research would also help us analyse the potential of CNN in overcoming data scarcity issue of under resourced language in the development of ASR without the integration of pronunciation dictionary and language model. Furthermore, it is in our expectation that the very first framework to develop an end-to-end CNN Iban-ASR system will be proposed at the end of this research, thus, it will serve as a reference for future CNN Iban ASR model development and fine-tuning. The result and protocol obtained from this research experiment would also be prove useful for future analysis and reference for identifying effective fine-tuning techniques on the CNN architecture. Finally, this research will help us to set a new benchmark for Iban ASR model performance as we attempt to further improve the accuracy of speech recognition models while implementing other various method of algorithm in predicting Iban words.

## **1.7 Research Outlines**

The thesis is organized into five chapters. Chapter 1 introduces the research work which includes our problem statement, research questions and objectives, as well as scope and significance of research. Chapter 2 discusses the literature review and previous existing works that has been done in improving under-resourced ASR model performance, as well as exploring currently trending CNN techniques while promoting the relevancy of conducting this research. Chapter 3 introduces our methodology and the description of our proposed framework to develop the first optimized Iban end-to-end CNN ASR model. The chapter explains our experimental steps and procedure in developing and optimizing our CNN model which includes implementing different model improvement techniques. Chapter 4 presents the results obtained from the experiments conducted in Chapter 3 as well as its analysis and discussion. Chapter 5 concludes the research thesis with a summary of the work that has

been done, our research contributions and limitations, and potential improvement that can be done to our proposed framework as future work.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Introduction**

This chapter discusses the related works of this research topic. It aims to present the relevancy of this research to the current technology trend and the latest low-resource language speech recognition issues by discussing existing literary resources. This includes the recent work done on Automatic Speech Recognition (ASR) for low-resourced language, the application of Convolutional Neural Network (CNN) in speech technology, and the Iban language. This chapter also discusses the machine learning process of CNN as a neural network, the difference between traditional and end-to-end pipeline of a neural network and the hyperparameter block and sub-block of the CNN as well as epoch. Furthermore, previous existing work that aims to improve the ASR model accuracy for under-resourced language using various method and solution was also presented. Finally, we discussed available tools to be used for CNN modelling before summarizing the chapter.

#### **2.2 Automatic Speech Recognition (ASR) and Under-Resourced Languages**

Automatic speech recognition (ASR) systems are system that takes speech audio data as input and transform it into text data as output (Iwama & Fukuda, 2019). It is a Natural Language Processing (NLP) technology that has helped many industries such as education, information retrieval, home automation and intelligent systems industry in solving voice automation tasks (Panda, 2017). Modern voice assistant applications such as Google Assistant, Amazon's Alexa, Apple's Siri, and Microsoft's Cortana all uses ASR to understand speech input and carry out its programmed functions. Modelling an ASR

typically requires acoustic model, pronunciation dictionary, language model and sufficient targeted language speech data (Juan, 2015; Kurata et al., 2021; Qin et al., 2022).

Under-resourced or low-resourced languages are languages that suffers data scarcity for training ASR systems (Krishna et al., 2021) or that has only a small population of speakers which result with insufficiency of speech data (Qin et al., 2022). It is easy to train ASR system using language such as English as its speech data can be easily acquired from different sources due to it being universally spoken around the world. In order for an ASR model to be able to correctly transcribe speech input with high accuracy, it requires sufficient training and learning of its targeted language which leads to the requirement of the abundance of speech data of the said language. This is a disadvantage and is an issue faced by under-resourced language as collecting speech data for under-resourced language is tedious and involves difficult tasks. Juan (2015) mentioned in their works in developing an ASR model for the under-resourced Iban language that collecting the Iban speech data is a difficult task as it requires them to visit local radio stations that uses the Iban language in their broadcast, hosting a three day Iban language collaborative workshop where eight native Iban speakers were hired and asked to help in transcribing Iban speech audio into speech transcript, and extracting articles for speech data through web crawling from an online news website that publishes news using the Iban language. Collecting speech data for under-resourced language consists of varying tasks, some includes site visits on native villages, hiring of language experts and time-consuming data-trainings (Pulugundla et al., 2018).

As mentioned earlier, typically an ASR model requires three components which are an acoustic model, pronunciation dictionary, and language model. An acoustic model instantiate relations between acoustic information and linguistic unit and is considered to be a critical component in a speech model (Bhatt et al., 2020). It is responsible for feature

extraction and statistical representation calculations of speech input which would heavily influence the recognition process. To train an acoustic model, a pronunciation dictionary is typically required. Pronunciation dictionary maps conventional symbolic transcript of speech to an acoustically/phonetically motivated ones (Lukeš et al., 2019). In a more general term, it consists of words and their respective phoneme sequences (Juan, 2015). Language model on the other hand is required by an ASR to predict word sequence (Soe & Theins, 2015). Nowadays, language model such as the Recurrent Neural Network Language Model (RNNLM) and Bidirectional Encoder Representations from Transformers (BERT) are often being used to help ASRs with rescoring tasks (Fohr & Illina, 2021; Jain et al., 2020; Yi et al., 2021). Figure 2.1 displays the typical pipeline of a traditional ASR.

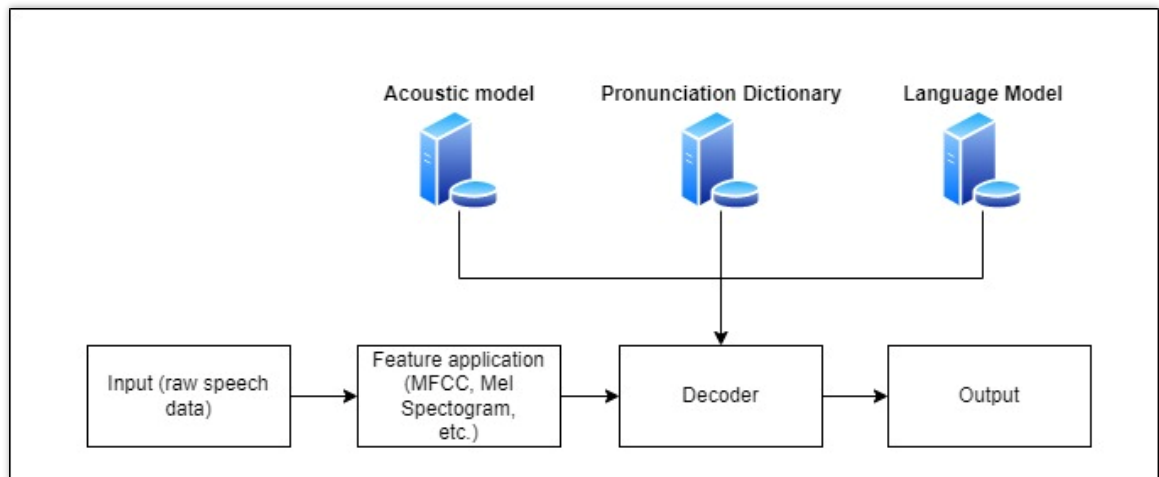


**Figure 2.1:** Pipeline of a Traditional ASR

As observed in Figure 2.1, the three components (acoustic model, pronunciation dictionary, and language model) can be illustrated as three big databases where each of the components requires abundance of speech data in order to be able to function properly and to help the ASR model to transcribe accurately. By acting as a guide for each task, the three components will help the ASR model to learn features of the input speech data (acoustic

model and pronunciation dictionary) and then providing it with reference and a list of probability for the most possible output (language model) in the decoding stage.

### **2.3 Neural Network Acoustic Model**

In the recent years, neural networks have often been proposed to become the acoustic model of ASR systems for under-resourced languages as it is able to provide significant improvements to the model performance (Luo et al., 2022; Mamyrbayev et al., 2020). It was mentioned that using neural networks as an acoustic model that exhibits deep learning algorithms helped to improve the performance of an ASR whereby it possesses the ability to measure unique raw speech features from the limited resources and learns more effectively (Lekshmi & Sherly, 2021). Neural networks or Artificial Neural Networks (ANNs) are adaptive systems that were inspired by the human brain where they possess the ability to restructure their internal design in relation to a function objective (Grossi & Buscema, 2007). They are adept in solving problems of the nonlinear type, thus, making them suitable to be used as acoustic model as speech data are nonlinear.

In the current trend, the CNN has recently been the preferred neural network for speech recognition task due to its exceptional feature extraction capability although it was first designed to solve image recognition tasks (Arnel Fajardo, 2020; Lekshmi & Sherly, 2021; Mon, 2019; Thai et al., 2020). In multiple research that studies the capabilities of CNN for ASR development, CNN ASR models has displayed promising result in terms of performance with some having WER of 0.03 using beam-search decoder with an external neural language model (Li et al., 2019) and 0.02 when using ContextNet, a fully convolutional encoder that incorporates global context information into convolution layers by adding squeeze-and-excitation modules (Han et al., 2020).

## **2.4 Convolutional Neural Network (CNN) as an Acoustic Model**

Recalling the issue stated earlier, the development of ASR for under-resourced language has proven to be quite a challenge. Albeit, past research has shown that ASR with single deep learning framework acoustic model such as the CNN was able to produce promising results using only raw unprocessed speech data in estimating phoneme (Palaz et al., 2013) and in producing ASR with excellent WER as compared to other ANN such as the DNN model architecture (Palaz et al., 2015).

Evidently, Palaz et al. (2015) has proven that CNN was able to outperform or produce similar result as compared to a DNN-based system, which takes standard cepstral features as input. They were able to prove that from their previous findings where poor ASR performance was achieved with raw speech signal as input to DNN, their study on Large Vocabulary Speech Recognition (LVCSR) indicates that CNNs have an edge over DNNs in modelling raw speech signal. From this study, they were able to get a WER of 6.7 using neural model of 3 CNN layers with 1 Max Pooling layer plus a hidden layer and HMM as decoder but taking only raw speech signal as its input. Meanwhile, their HMM-ANN model with 1 hidden layer and uses MFCC to pre-process its input scores a WER of 7.0. Both were tested and trained on the Wall Street Journal (WSJ) corpus. Increasing the layers, with 3 hidden layers for each model, their CNN model outperforms the ANN model even further with a WER of 5.6, while the ANN achieved a WER of 6.4.

### **2.4.1 CNN in speech technology**

Furthermore, much recent research has been done that shows CNN outperforms many other speech model architectures including the famous Recurrent Neural Network (RNN) in building ASR for low-resourced languages. CNN were stated by Fajardo (2020), Thai et al. (2020), and Mon (2019) to have a very exceptional feature extraction capability,

in which all the authors were conducting experiment on their respective under-resourced language using the CNN architecture and achieved improved prediction accuracy ASR models. Since CNN is the state-of-the-art architecture for image and graphic classification, it is not surprising that this neural network would be able to perform better in feature extraction. Taking raw speech signal as spectrogram, the CNN were able to distinguish even the smallest difference in features, leading to accurate prediction of spoken words. Mon (2019) mentioned optimization of hyperparameter in CNN architecture was able to produce better performance, as well as providing great impact on the performance of ASR through the setting of feature maps number and pooling sizes of the CNN. Thai et al. (2020) on the other hand stated that both deep RNN and HMM/GMM models were outperformed by their fully convolutional model, in which it yields significant accuracy improvements as compared to the result produced by the RNN and HMM/GMM models. Moreover, the same author further mentioned that CNN mitigates the issue of incompatibility in parallelization on modern hardware, and that CNN is advantageous as it requires only few parameters to collect enough important features for accurate prediction that leads to the reduction of computational cost.

The use of end-to-end CNN architecture in ASR modelling also includes works on under-resourced languages which has been presented in a lot of research and even in recent paper publications. In a much recent work, Lekshmi & Sherly (2021) stated that for their Malayalam language ASR, the CNN is a better deep learning model due to its capability to reduce spectral variations and in modelling the spectral correlation between signals (Lekshmi & Sherly, 2021). Zhan et al. (2021) in their attempt to develop a reliable ASR system for low-resourced languages proposed a novel representation extraction model for their domain adversarial based multi-task learning model that combines deep CNN-based

model and domain adversarial learning. They mentioned in their work that CNN have a better ability to capture the articulator's information. The author van der Westhuizen (2021) in their work to develop an ASR free keyword spotting system for an under-resourced language stated that they achieve computationally efficient keyword spotting system by also training a convolutional neural network (CNN) to emulate dynamic time warping (DTW)-based scoring. They mentioned that their resulting CNN-DTW keyword classifier is much faster in keyword spotting than direct DTW, since no alignment is required.

#### **2.4.2 Under-resourced language - The Iban language**

Focusing on the issue of data scarcity in ASR development, our targeted under-resourced language is the Iban language. Iban language is a native language from Sarawak, Malaysia, in which it is still being used widely by its local people as a means of communication as well as official matter proceedings, though, this rarely applies to their written records. Sarawak is stated to be the largest state in Malaysia, with an estimated population of 2.9 million in the year 2020 as recorded by the Department of Statistics Malaysia. Sarawak is situated on the northwest of the Borneo Island and shares borders with the state of Sabah, Brunei and Kalimantan from Indonesia (Juan, 2015). The community in Sarawak speaks in many different languages which includes Iban, Sarawak Malay (Malay dialect), Melanau, Bidayuh, Penan, and Kelabit.

The Iban language is mostly spoken by the Iban, an ethnic group that comprises the largest percentage (28.8% as of 2023) of Sarawak population. The Iban language originated from the Malayo-Polynesian branch of the Austronesian language family, similar to the Malay language, and falls in the Ibanic language group (Juan, 2015). It was stated that Iban is historically related to Malay where both languages belong to one branch called, Malay-Iban (Rohani, 2003). This hypothesis was theorized due to the many similarities between

Malay and Iban as compared to other languages from other subgroups. Juan (2015) mentioned that the Iban language has been taught in schools in Sarawak at the primary and secondary level as an elective subject since the early 90s. The language has also been introduced as a subject in the university level where it was offered as an elective to undergraduate students. There are local radio stations in Sarawak such as Radio Televisyen Malaysia (RTM) and Cats FM that broadcasts news and programs in Iban. Previously, local newspaper, “The Borneo Post”, includes Iban language in its article. Up to this day, the Iban language is kept being standardized and is being preserved in culture and literature by several organizations in Sarawak such as Tun Jugah Foundation, Majlis Adat Istiadat Sarawak (Sarawak Native Customs) and the state’s education department.

The Iban language has its own speech corpus which was previously developed by Juan (2015), the elaborated details of the corpus is included in section 3.2.1 of Chapter 3. It has since been used in the developments of NLP applications, which includes, the development of an under-resourced language ASR model using DNN with Multi-Task Learning (MTL) acoustic landmarks by He et al. (2018) and the development of an under-resourced Seneca CNN ASR model by Thai et al. (2020) in which the developed speech model was tested on the Iban corpus for performance evaluation.

## **2.5 The previous Iban ASR system with Deep Neural Network acoustic model.**

Juan's (2015) Iban ASR with its best WER of 0.15 was developed using Deep Neural Network (DNN) acoustic model. Their DNN was obtained via network training using state-level minimum Bayes risk (sMBR) and it consists of 7 layers of which 6 are hidden layers that consist of 1024 hidden units (Juan, 2015). The network was also trained using Mel-Frequency Cepstrum Coefficients (MFCC) for 11 consecutive frames and HMM states were used as targets. As for its initial weight, Restricted Boltzmann Machines (RBMs) were used

which resulted with 6 stacks RBMs of deep belief network. Furthermore, the DNN was fine-tuned using Stochastic Gradient Descent with per-utterance updates while its learning rate were kept constant at 0.00001 for 4 epochs. The DNN experiments were ran using GPU and CUDA toolkit to speed up computational processes (Juan, 2015).

Using 7 hours of Iban speech training data, Juan (2015) was able to produce a DNN Iban ASR with 0.18 WER. As a step to improve its performance, cross-lingual transfer learning was carried out on the model using two source languages: Malay and English. The process begins with obtaining two speaker-adapted DNN system that is trained specifically on the Malay and English data. For the DNN targets, 3K context-dependent triphone states for Malay and English that was obtained from Juan's (2015) previously developed HMM/GMM system were used. To acquire GMM systems, 39 MFCC with deltas with deltas-deltas as well as applied LDA, MLLT and fMLLR were employed by Juan (2015). For cross-lingual approach, Juan (2015) removed the DNN's final layer (non-hidden layer) for both Malay and English data. The same step as described above was then repeated for the Iban data instead. Finally, Juan (2015) built a language-specific DNN for the Iban language via the fine-tuning of the hidden layers used during the Malay and English DNN experiment onto the Iban 7h training data. Both Malay and English hidden layers produced similar performance of 0.15 WER when trained on the Iban 7h dataset.

## **2.6 Architecture of CNN**

As for our chosen neural network architecture, we first observe the uniqueness of the CNN architecture as compared to other ANNs. Instead of using fully connected hidden layers as what is being implemented by its similar variant, DNN, the CNN implements a special network structure, composing an alternating so-called convolution together with pooling layers. The idea of the architecture of CNN was adapted from the pattern of the

neuron's connection in the human brain and the arrangement of the Visual Cortex (Saha, 2018). Relating to CNN, the convolution process mimics the function of neurons in the brain where its main purpose is to extract features from a visual input. It is highly useful in image processing and identification. Through the learning of image features using small squares of input data, convolution is able to preserve the spatial relationship between pixels as an image is basically a matrix of pixel values (Ujjwalkarn, 2016). Essentially, a matrix of pixel values can be used to represent every image as well as waveforms. Typically, CNN consists of four major operations: Convolution, Non-Linearity - Rectified Linear Unit (ReLU), Pooling step, and Classification (Fully Connected Layer). The next section explains how learning process occurs in the architecture of the CNN.

### 2.6.1 Learning Process in CNN

In Convolution, consider a 5x5 image matrix whose pixel values are in a special case of only 0 and 1 as shown in Figure 2.2 below,

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

**Figure 2.2:** Original 5x5 Matrix

Also, consider another 3x3 matrix as shown in Figure 2.3 below. This small matrix is known as a “Kernel”, “Filter”, or “Feature Detector”.

|   |   |   |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

**Figure 2.3:** 3x3 Kernel Matrix

As represented in Figure 2.4, imagine our 3x3 matrix (Blue) slide over the 5x5 image matrix by 1 pixel, this process is known as a ‘stride’. As we slide, element wise multiplication was computed between the two matrices, and its multiplication outputs was added to obtain the final integer, in which, forms a single element of the output matrix (Green).

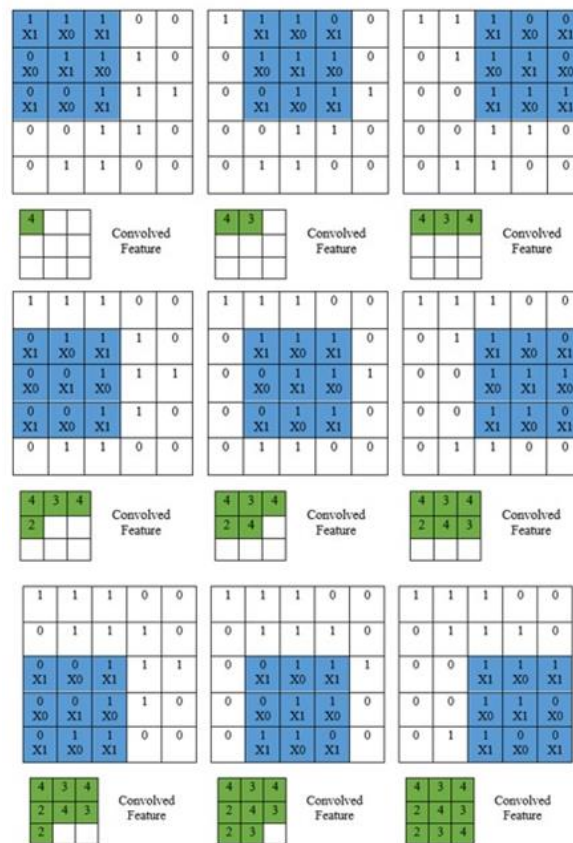


**Figure 2.4:** Convolution process in CNN

The output matrix (Green) was formed by sliding the 3x3 kernel matrix over the Original 5x5 matrix. The computation of the dot product is a process called “Convolved

Feature” or “Activation Map” or the “Feature Map”. Note that the kernels detect features from the original input image (5x5 Matrix). On the same input matrix (image), different feature maps are produced by different values of the kernel matrix. During implementation, the values of these kernels is learned by the CNN throughout the training process. Although, parameters such as number of kernels, kernel size, and architecture of the network is still needed to be specified before the training process. Furthermore, before the convolution step is performed, three parameters have to be specified as the size of the Convolved Feature is controlled by it. The parameters are depth; which represents the amount of kernel used in convolution, stride; which represents the number of pixels that was slide over the input matrix by our kernel matrix, and zero-padding; zeros that pads around the border of the input matrix in which, allows the application of the kernel to the bordering elements of the input matrix. Higher number of kernels corresponds to higher image features extraction as well as better pattern recognition in unseen images by the network.

Next, the Non-Linearity - Rectified Linear Unit (ReLU) applies non-linearity into the CNN as Convolution is a linear operation whereas most real-world data are non-linear. The purpose of activation function is to transform the summed weighted input from the node into the activation of the node or output for that input of a neural network (Brownlee, 2020). For CNN, ReLU is the most commonly used activation function as it has been found to perform better in most situations as compared to the other non-linear functions such as Tanh or Sigmoid. ReLU also has become a common activation function for many types of neural networks as it offers ease of training process and often provides better performance. It is a piecewise linear function that outputs the input directly if it is positive, if it is negative, zero. (Brownlee, 2020). From the previous example, when an image was formed from the output feature map, it would be a mix of black and white shades as it is a result of: Black = negative

values, White = positives values. Applying ReLU to the CNN means that we would be removing all negatives values from the feature map and will only be taking the non-negative values. This ease the CNN to identify the image of a feature map as there is only non-negative values to indicate the differences of layers in the feature map. A feature map that has been applied with ReLU is also called as a “rectified” feature map.

During the Pooling step (Spatial pooling) or also known as subsampling, this step reduces each feature map’s dimensionality all the while retaining important key information. Usually, pooling layer comes after the finish of the Convolutional layer. It serves as a layer to reduce the input volume’s spatial dimensions (width x height) for the next Convolutional Layer, though, the depth dimension of the volume is not affected. This operation is also known as subsampling because loss of information also occurs alongside the reduction of the dimensions. This loss however is more beneficial to the CNN, mainly for these two reasons:

- i. Next layers in the network will have less computational overhead.
- ii. Prevents overfitting.

Similar to Convolution, the pooling layer takes a sliding window in which it is another newly defined matrix window that is to be striding along the rectified feature map. On this operation, the window will be transforming the values from the feature map either by taking the maximum value or the average value, as observed under the window (known as ‘Max Pooling’). Due to its better performance characteristics, max pooling has been favoured in most situation. Figure 2.5 depicts the operation of Max Pooling on a rectified feature map using a 2x2 window.

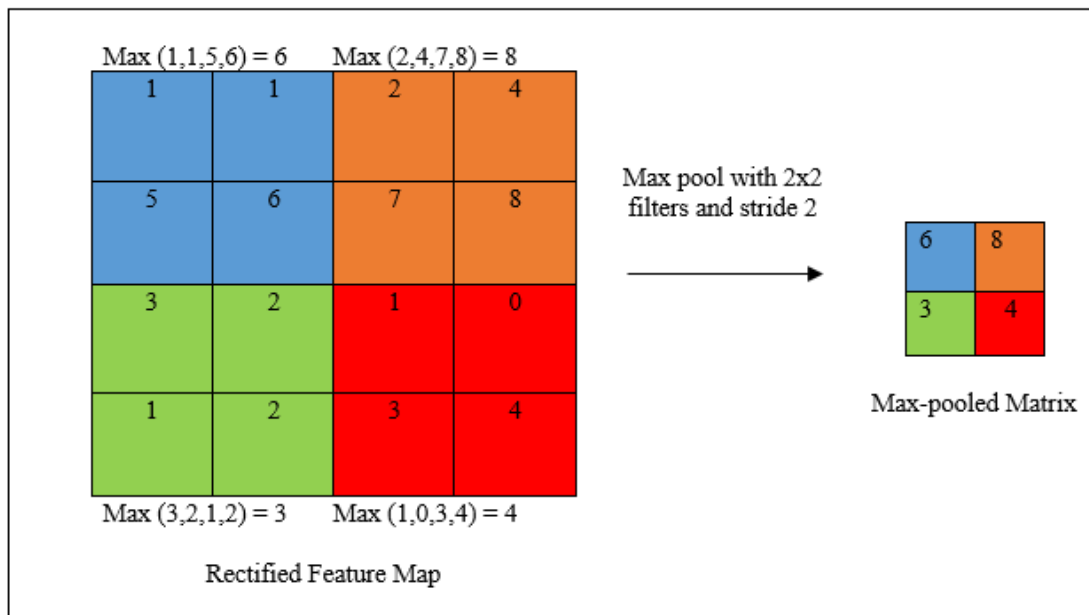


**Figure 2.5:** Max pooling of a rectified feature map

Max pooling can be deduced as an operation of applying a function over input values. No new parameters were introduced during this phase and it uses fixed sized portions at a time, with the size configurable as a parameter. Due note that although it is applicable, pooling operations is excluded in most CNN architecture. So far, the operations have reduced the size of the original matrix to be even smaller than it was before as it was divided into smaller sized feature maps. As the operation progress further to the second convolution, then the second pooling step, and then onwards to the third convolution and also the third pooling step and so on, the size of the matrix will further be reduced and more feature maps will be produced until the network can finally conclude one or more possible outputs for the original image.

Finally, in Classification or Fully Connected layer, this represents the characteristic of the CNN layers whereby all of them are fully interconnected with each other. In other words, each neuron in the layer before is connected to every neuron on the next layer. Outputs from Convolution and Pooling Step represents high-level features of the original

image. Fully Connected layer uses these features for various class classification for the input image based on the training dataset (Ujjwalkarn, 2016). Moreover, by implementing fully connected layers, this enhances the learning of non-linear combinations of those features. A combination of the features gained through convolution and pooling step operation would be very helpful in classifying an image. Through the usage of Softmax function, the sum of the output probabilities is ensured to be equal to 1. Softmax as an activation function in the output layer of a fully connected layered network, takes a vector of arbitrary real-valued scores, and compresses it to a vector of values between zero and one that sum to one.

### 2.6.2 Traditional VS End-to-end CNN pipeline

Different from a traditional ASR pipeline, CNN has an end-to-end architecture; it is considered to be an enhanced NLP model that is based on the deep learning framework for speech recognition. We can observe the differences between a traditional and end-to-end pipelined ASR in Figure 2.6.

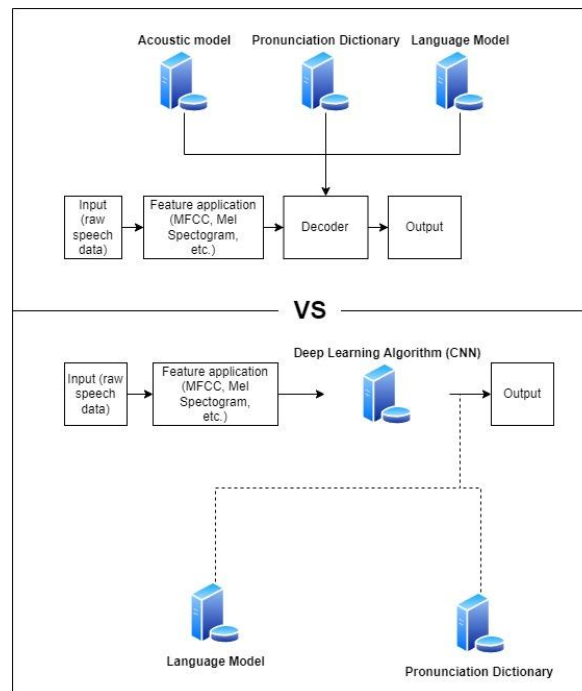


**Figure 2.6:** Traditional Vs End-to-end ASR development pipeline

For a traditional ASR pipeline as shown in Figure 2.6, the training process for the acoustic model, pronunciation dictionary (PD), and the language model (LM) of the ASR were carried out separately, whereas for an end-to-end system, it requires only the acoustic model as the neural network of the ASR model.

In end-to-end models, training takes place throughout the whole system where its components (layers and sub-layers) are interconnected to each other. It requires only the acoustic model which is composed of the targeted corpus' audio data files and its transcript. End-to-end architecture allows the exclusion of using LM and PD (connected with dotted line as shown in Figure 2.6) which are core components of conventional ASR and that would cost a lot of time and resource to be developed as mentioned earlier. With the issue of lacking in written resources and the lack of abundance text data, the CNN end-to-end architecture present itself as solution for many under-resourced languages in the world. An example of CNN end-to-end model includes the Quartznet and Jasper.

### **2.6.3 Quartznet: an end-to-end CNN**

The Quartznet architecture is a variant of Jasper (Li et al., 2019) and was proposed by Krizan et al. (2020). The model is built out of multiple blocks with residual connections between them where each was trained using CTC loss. Each block consists of modules with 1D time-channel separable convolutional layers, batch normalization, and RELU layers. It exhibits an end-to-end architecture whereby all of its components are interconnected to each other. Generally, the Quartznet model consists of  $B \times R$  configuration where B stands for Block and R stands for Repeat which represent the number of Sub-block repetitions in a block. The model consists of two types of blocks; blocks that is a variant of  $B \times R$  and those that are invariant of  $B \times R$ . A variant of  $B \times R$  blocks are blocks with changings configurations

throughout the experiment sessions. These are the blocks in which its configuration would be specified according to our preferences.

Invariant  $B \times R$  blocks on the other hand are blocks with fixed configuration. Invariant blocks are for pre-processing and post-processing of the model (Li et al., 2019). Figure 2.7 displays an example of a Quartznet model, namely, the Quartznet  $4 \times 1$  model which stands for Quartznet with 4 blocks and 1 sub-block. This architecture is selected as case study because it was the most basic loadable model configuration that was found in the repository of our used toolkit.

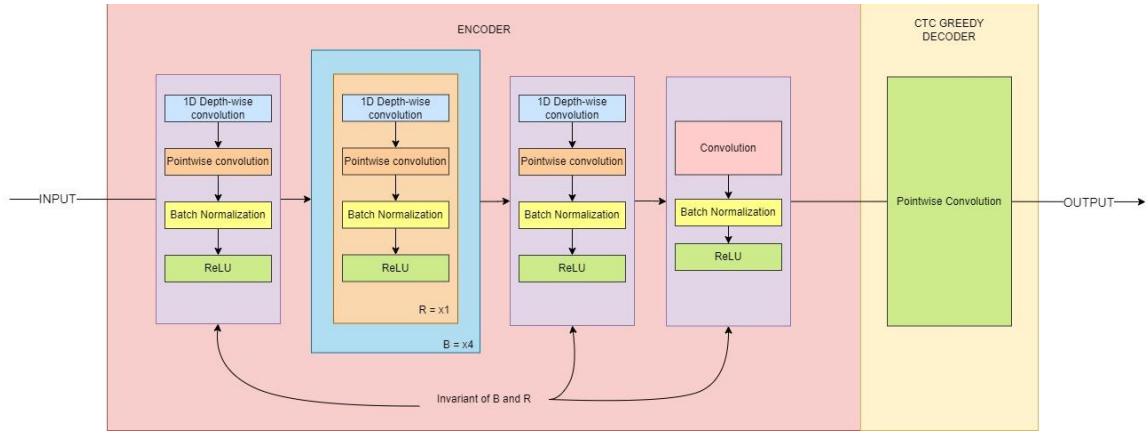


**Figure 2.7:** Quartznet 4x1 Architecture

As observed in Figure 2.7, each block in the model consists of four layers: variable sized depthwise convolution, pointwise convolution, normalization, and the ReLU layer. It comprises an amount of 4 variant  $B \times R$  blocks with each having only a single sub-block and Connectionist Temporal Classification (CTC) as its decoder. The total number of blocks that can be found in this model architecture is 7 which includes the first and the last two subsequent block where all three is invariant of  $B \times R$ . The first block typically comprises of 128 filters with single sub-block, while the following 4 controlled blocks,  $B$ , were configured to have sub-blocks,  $R = 1$ , where each block comprises 256 filters. The next two subsequent

blocks are invariant of  $B \times R$  ending blocks where each typically consists of 256 and 1024 filters respectively with a single sub-block.

#### **2.6.4 Epoch, block, and sub-block of Quartznet**

The Epoch, Block, and Sub-block parameter are three of the many hyperparameters that can be found in a Quartznet CNN model. These three hyperparameters needed to be specified initially before the training of a model began. An epoch is when one pass of a training was completed over the entire dataset (Afaq & Rao, 2020), a block (variant and invariant of  $B \times R$ ) in a Quartznet CNN is a component that allows the learning of information in the architecture's model in which each of them consists of the same base module that contains four layers: a variable-sized depthwise convolutional layer with channels, a pointwise convolution layer, a normalization layer, and the ReLU, while a sub-block is a repetition of the whole process in a block of a Quartznet CNN (Kriman et al., 2020). Although there are correlations between these three hyperparameters in setting up an optimized network structure for the Quartznet CNN, no works have been done in studying these connections.

### **2.7 Existing Works on Improving ASR Model Performance**

Currently, various type of methods has been designed to improve ASR model performances. This includes implementing ANNs algorithm such as Deep Neural Network (DNN), Recurrent Neural Network (RNN), and the Convolutional Neural Network (CNN) into acoustic models as well as merging of acoustic models and using multilingual acoustic models. There are already a number of existing research and papers written to investigate how to further improve the performance of an ASR model for under-resourced languages as well. Implementing multilingual neural network ASR architecture for under-resourced language is a popular method in achieving better accuracy. Biswas (2018) has adopted

multilingual neural network ASR architecture on their effort to improve the performance of their ASR for the native language of South Africa, isiZulu, in which has an issue of ‘code-switching’, a mixing of different language with their native language in the usage of their everyday conversation, with English. Their team implemented 3 types of acoustic model, two implements multilingual method as an alternative to see the matter solve. They were able to enhance the performance of their ASR to an overall WER of above 0.55 from an initial WER performance baseline of 0.63, which is an increased performance of roughly 0.08 reduced error-rate, for each experiment they have conducted. Their acoustic model was a variant of GMM-HMM, Multilingual DNNs and Multilingual TDNN-LSTMs. The effort was a good improvement for their ASR in terms of WER performance, although, their results still scale as a “high-error rate” ASR performance. They concluded that a higher amount of monolingual data for training is necessary in order to achieve a very good WER performance for their under-resourced language.

Furthermore, He et al. (2018) has also conducted a research experiment on improving WER of an ASR for an under-resourced language, Iban, through the means of adopting a Multi-task Learning (MLT) and acoustic landmarks method of ASR training. They use an ASR trained with the TIMIT corpus using MLT method and then further trained it for detection and classification landmarks. The same ASR was then cross-lingually adapted to the Iban corpus as a secondary task. From their experiment, they were able to increase the performance of an ASR trained with very low resources, “Iban 10%”, which is a training on the ASR with only 40 minutes of Iban data, to a reduction of 0.087 WER on monophone and 0.0617 on triphone ASR. Although, while a cross-language Landmark detector provides useful information complementary to the orthographic transcription, visual inspection indicates that a cross-language landmark detector is not as accurate as a same-language

landmark detector (He et al., 2018). They later suggest training a more accurate landmark detector using RNN methods that could be applied to multilingual trained corpora as a mean to overcome the aforementioned belief as their future work.

To further improve the performance of their Malay ASR model, Tan et al. (2014) implements the merging of acoustic models using models with multilingual speakers. The author proposed method combines two acoustic models; the first one being a context dependent Malay Gaussians (GMM) acoustic model that was previously trained on MASS Malay speech corpus that consist of 120 hours of training data, the second one being an English Gaussians acoustic model that was trained on a non-native 9-hour English training data. The merging includes the process of phone mapping using the International Phonetic Table (IPA) table for similar phones and using perception test for unique phones (Tan et al., 2014). Through the implementation of acoustic model merging, the author was able to achieve a relative improvement in WER to its ASR model by 3% to 10%.

### **2.7.1 Hyperparameter Optimization**

Trending nowadays, optimization of hyperparameters in model and CNN architecture is a popular approach in improving the performance of a ASR system. To improve the performance of CNN+RNN (CRNN) speech recognition models, Mou et al. (2020) studies the importance of the number of filter of the convolutional layer and conduct optimizations. The author aims to achieve the development of a CRNN model which has high word accuracy but having light model- weight. Their studies found that the number of filters of the convolutional layer is a key parameter that is able to significantly change the training dynamics of the said model. The author mentioned that when adding CNN to RNN speech recognition models, there exists a threshold number of filters that which can render it useless unless the number of filters was within its range. The author uses a variant of

convolutional layer called the convolutional embedding layer (CEL) and stated that the larger the number of filters of the CEL is, the better the word accuracy will be achieved for the LVCSR (a variant of ASR) model. Based on their experiment results, the author proposed that not all CNNs can improve word accuracy but only a subset of those with optimized parameters were able to improve model performance when added to RNN speech recognition models (Mou & Li, 2020). Another work done by Andonie & Florea (2020) focuses on studying the Weighted Random Search (WRS) algorithm as an optimization technique in optimizing the CNN's hyperparameter. The author ran several state-of-the-art optimization techniques to be compared against the WRS method to determine whether it could outperform the other algorithms and propose an optimized CNN model structure. The author stated that based on their experimental result, the WRS optimization algorithm did outperforms several state-of-the art hyperparameter optimization methods. However, a challenge exists in determining the optimal architecture for neural network models, that is: different dataset demands different model optimization. Therefore, adjustments are required for each individual case. The author pointed out that according to WRS, the best CNN architecture has six convolution layers and one fully-connected layer, with  $C1 = 736$ ,  $C2 = 508$ ,  $C3 = 664$ ,  $C4 = 916$ ,  $C5 = 186$ ,  $C6 = 352$  and  $F1 = 1229$ , where  $C1$  to  $C6$  being the number of output filters for each convolution layers and  $F1$  being the number of neurons for each fully connected layer (Andonie & Florea, 2020). Guliani et al. (2021) proposed a framework to train speech recognition model which utilizes federated learning. Federated learning implements optimization on model training using federated averaging algorithm (FedAvg) that consists of two levels: the conduct of local optimization on a number of participating clients, and a global model update server step. The work was focused on optimizing, hyperparameter tuning, and regularizing model trainings to achieve balance in

the cost for computational resource while compensating for quality output of speech recognition model. Through adjusting random client data sampling on non-independent and identically distributed (non-IID) data, they stated that their proposed framework was able to produce a better model at lower cost that outperforms a baseline model that implements training optimization using the Adam-SGD optimizer (Guliani et al., 2021). Finally, in a recent work, Santosh Kumar (2022) introduces an optimization algorithm for feature selection of ASR model to improve its performance that was based on population-based algorithm called the Northern Goshawk Optimization (NGO), taking the characteristic of the Northern Goshawk hunting technique into the optimization algorithm. The optimization technique focuses on the feature selection stage of speech recognition whereby the algorithm is comprised of three parts, which are the initialization of the population (variables), the identification of the prey (feature exploitation), and the capture of the prey (feature selection). The NGO consists of similar idea implemented to the techniques in population-based algorithm but with different unique mathematical formulas that was proposed to carry out each of the process involved. Using the Librispeech as their dataset and Mel-Frequency Cepstrum Coefficients (MFCC) as their feature, they stated that the NGO algorithm produces model accuracy that outperforms the result obtained by models using existing methods in feature selection (S. K. S & S H, 2023).

To obtain a CNN model with high accuracy, it is required for the model to have an optimized hyperparameter (Aszemi & Dominic, 2019). Based on all of the previous works that has been presented, the architecture of model which is determined by its hyperparameter is critical in producing an ASR that could produce accurate prediction, thus, it is important for the model to be in an optimized state.

### 2.7.2 Spectrogram Augmentation

Spectrogram Augmentation is a process of feature masking whereby it masks unnecessary features (noise) that is found in the audio spectrogram input to increase quality and efficiency. The technique present itself as another trending solution to further improve ASR model performance especially for under-resourced languages. Since CNN is an artificial neural network that receives sound waves-like images as input, we could implement spectrogram augmentation on each input as what was being done in an image classification model training.

Spectrogram augmentation is the process of applying augmentation to the input waveform as an effort to increase the effective size of an existing audio data. In the work done by DeVries & Taylor (2017), the process of Cutout or also known as rectangle masking is a simple regularization technique for CNNs where it uses partially occluded versions of existing samples that was obtained through the process of removing contiguous sections of our initial input images, effectively augmenting our dataset. In the work, it was mentioned that Cutout can be interpreted as an extension of Dropout in input space, which is the setting of hidden unit activations to zero with some fixed probability during training, but with a spatial prior applied. This is similar to how CNNs apply a spatial prior to outperform the performance achieved by feed-forward networks on image data.

In another work that utilizes spectrogram augmentation by Park et al. (2019), time and frequency masking or time-frequency masking, an augmentation technique in their proposed spectrogram augmentation method called ‘SpecAugment’, is a process of masking a block of consecutive time steps or mel-frequency channels. This augmentation method operates on the log-mel spectrogram of the input audio, rather than the raw audio itself. It modifies the spectrogram by warping it in the time direction, masking blocks of consecutive

frequency channels, and masking blocks of utterances in time. This helps the network to be robust against deformations in the time direction, partial loss of frequency information and partial loss of small segments of speech of the input.

Anoop and Ramakrishnan (2021) implements SpecAugment data augmentation on their low-resourced Sanskrit ASR model to improve model performance. Their ASR model uses end-to-end CNN architecture with CTC as decoder with an initial WER of 0.08 model performance. After implementing spectrogram augmentation, their model achieved an absolute improvement of 13.86% in WER on Weighted finite-state transducer (WFST) decoding (A. C. S. & Ramakrishnan, 2021). Meng et al. (2021) proposed a new data augmentation technique MixSpeech which mixes two speech sequences in the input and mixes two loss functions regarding the text output, to outperform previous baseline ASR model performance and its result with the implementation of SpecAugment. Their work presents spectrogram augmentation (MixSpeech) being applied to the Transformer model where they state that their augmentation technique has further increased both model performance on two datasets; Wall Street Journal (WSJ) and Hong Kong University of Science and Technology (HKUST) Mandarin corpus, outperforming their baseline WER and also when implemented with SpecAugment. The MixSpeech augmentation was able to further reduce 0.12 and 0.07 WER to the baseline and SpecAugment performance of the WSJ respectively, while further reducing 0.1 and 0.06 WER to the baseline and SpecAugment performance of the HKUST Mandarin corpus respectively, achieving 0.05 WER on the WSJ dataset and 0.23 WER on the HKUST dataset (Meng et al., 2021). Song et al. (2020) also presents a new spectrogram augmentation technique inspired by SpecAugment called SpecSwap which augmentation policy involves the swapping blocks of frequency channels and time steps. Instead of masking noises in the time and frequency

channel, the technique focuses on swapping both channels to introduce input efficiency. They conduct experiment using Transformer-based model on Aishell-1 Mandarin speech corpus and was able to outperform several baseline result of existing hybrid and end-to-end model Character Error Rate (CER) performance, achieving 7.60% CER (Song et al., 2020).

By focusing on the increase of robustness of our speech input, spectrogram augmentation helps to improve model performance by masking noises and unnecessary features found in the speech input spectrogram or deforms the input speech waveform to a simpler feasible structure, directly increasing the efficiency of feature extraction process of the CNN without the need to collect additional speech training data.

### **2.7.3 Cross-lingual Transfer Learning Method**

Being a popular solution back in the previous years for the development of under-resourced language ASR, transfer learning method is still a trending approach even to this day. It is a method that exploits a previously trained model for a specific underlying base task, to reuse it as a starting point to train a different model for another new task. This bootstrap method is especially effective when the new task involves under-resourced data and is similar to the base task (Das et al., 2021). Cross-lingual transfer learning method is the process of knowledge transfer from an ASR model of a language to another ASR model of a different language. The idea of this method is to provide alternative resources for under-resourced languages to learn as it improves their ASR model performance via the extraction of knowledge from an existing ASR model that was trained using high resourced language such as English. The Transfer learning method has been very popular in overcoming the issue of data scarcity in developing low-resourced ASR model, recently there are many works that promotes the technique.

Xu et al. (2022) Develops a novel Chinese dialects discrimination framework using transfer learning and data augmentation to overcome the issue of lack of annotation resource. They proposed a framework that includes transfer learning as one of its modules, taking advantage of a large Chinese dialect corpora to generate a source-side ASR model. The transfer learning was necessary as it helps to capture speech features that had already been learned by the large corpus ASR model to be utilized in recognizing speech features in the under-resourced Chinese dialect ASR model. They reported that their model outperforms state-of-the-art methods significantly on two benchmark Chinese dialects corpora which are Gan Chinese dialect dataset with 63% correct word prediction accuracy and the Hakka Chinese dialect dataset with 80% correct word prediction accuracy. Wei et al. (2022) in their work to investigate approaches that helps to improve the WER and PED performance on learners' speech of the Dutch language, a pluricentric language, stated that the transfer learning method is one of the alternatives that they adopted to improve their model performance. They used a conventional DNN ASR model and carried out transfer learning that utilizes native and non-native Netherlandic Dutch dataset to improve their non-native under-resourced Flemish Dutch ASR model. From their experiment, they reported that their produced transfer models attained better results than their plainest conventional model, achieving 0.53 - 0.54 WER (Wei et al., 2022). Qin et al. (2022) implements transfer learning in their work to improve their speech recognition performance of the under-resourced Lhasa dialect language. They proposed two strategies to solve the issue of data scarcity suffered by the Tibetan Lhasa dialect language which include multilingual transfer learning of Mandarin, Sinhala, Bengali, and Nepali dataset. Implementing transformer-based end-to-end model, they reported that they were able to outperform their baseline performance obtained using conventional DNN-HMM and end-to-end system, improving 8.4% CER to their previous

model performance (Qin et al., 2022). Hassan et al. (2022) attempts to solve the issue of the inaccuracies of English ASR model when predicting inputs from non-native south Asian English accent speakers through the conduct of transfer learning using pre-trained native English accent ASR models to non-native English speaker dataset. Using the Librispeech corpus as their native English dataset, they produced a pretrained RNN model to be trained on the Deep2Speech non-native English dataset under three different transfer learning settings which involves retaining low-level features and freezing convolutional layers. They reported that by carrying out transfer learning under the three settings, they were able to reduce their model WER performance from 43% to 18.08% at a lower validation cost (Hassan et al., 2022). Table 2.1 presents the improvement contributed by implementing transfer learning technique in all of the previously mentioned works.

**Table 2.1:** Contribution of Transfer Learning technique to various research

| Research Title                                                                                            | Author            | Contribution achieved by Transfer Learning technique                                                                                                                                        |
|-----------------------------------------------------------------------------------------------------------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Low-Resource Language Discrimination toward Chinese Dialects with Transfer Learning and Data Augmentation | (Xu et al., 2022) | Outperform previous baseline model by <b>increasing 0.08 accuracy</b> in Gan Chinese dialect discrimination.<br><br>Outperform previous baseline model by <b>increasing 0.1 accuracy</b> in |

**Table 2.1** continued

|                                                                                                                                                   |                       |                                                                                  |
|---------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|----------------------------------------------------------------------------------|
|                                                                                                                                                   |                       | Hakka Chinese dialect discrimination.                                            |
| Automatic Speech Recognition and Pronunciation Error Detection of Dutch Non-native Speech: cumulating speech resources in a pluricentric language | (Wei et al., 2022)    | Outperform existing baseline conventional model performance by <b>0.25 WER</b> . |
| Improving low-resource Tibetan end-to-end ASR by multilingual and multilevel unit modelling                                                       | (Qin et al., 2022)    | <b>Reduce 0.08 CER</b> to their previous model performance.                      |
| Improvement in Automatic Speech Recognition of South Asian Accent Using Transfer Learning of DeepSpeech2                                          | (Hassan et al., 2022) | Further <b>reduce 0.25 WER</b> from their baseline model performance.            |

Based on these presented works, it shows us that transfer learning helps to mitigate the issue of data scarcity of low-resourced language by allowing low-resourced ASR model to learn new classification method that has previously been learned by a high resourced language ASR model through the utilization of high-resourced pretrained ASR model. It also displays how optimizing the process in transfer learning such as implementing encoder freezing, spectrogram augmentation, and adjusting transfer learning hyperparameter values helped in further improving an ASR model beyond its baseline performance. But to

implement these techniques, we must first identify available tools to be used in modelling a speech recognition model.

## **2.8 Available Tools for CNN Modelling**

There are currently several toolkits that is available to be used for modelling neural networks and CNN. This includes Kaldi, CMUSphinx and Nvidia NeMo. This section generally describes the detail of these toolkits and its usefulness in modelling an ANN.

### **2.8.1 Kaldi**

Kaldi is a speech recognition toolkit that allows users to code their own ASR using mainly C++ and Python, with ease of modification and extension. The engine is originally coded using the programming language, C++. It has its own GitHub repository where users from all around the world can have access to it and use it as their ASR's speech engine toolkit. As mentioned from their website ("Kaldi: About the Kaldi project," n.d.), some of the important features that made up Kaldi as speech engine toolkit includes code-level integration with Finite State Transducers (FSTs), where Kaldi uses OpenFst toolkit as their library and compile against it. Kaldi also consists of extensive linear algebra support where they include a matrix library that wraps standard BLAS and LAPACK routines. Furthermore, Kaldi exhibits the feature of extensive design. The entire algorithm in Kaldi is design to be as generic as possible, for example their decoders are templated on an object that provides a score indexed by a (frame, fst-input-symbol) tuple. This allows the decoder to be able to work from any suitable source of scores, such as a neural net.

Kaldi is very helpful when used in modelling personal ANN as it holds many complete speech recognition recipes that could be used as reference or guides in building our very own model. Kaldi's goal is to make available complete recipes for building speech

recognition systems, that work from widely available databases such as those provided by the Linguistic Data Consortium (LDC) to the public. Thus, using Kaldi enables you to access to so many speech recognition resources such as the codes, algorithm and even the design of a model released by other developers worldwide into the Kaldi repositories. Kaldi also provides tutorials for first time Kaldi-users before jumping-in into the development of personal ASR system. Kaldi is still being updated with progress and new tools until this very day.

### **2.8.2 CMUSphinx**

The CMUSphinx is another speech recognition toolkit that consists of various tools that is used to build a speech application. This toolkit is the result of collections gathered from over 20 years of CMU research. CMUSphinx provides four major packages for different tasks and applications. This includes PocketSphinx, a lightweight recognizer library written in C; this library allows user to develop mobile speech application in androids, Sphinxbase, a support library that is required by PocketSphinx as well as all CMUSphinx projects as it provides the common functionalities that is provided by the CMUSphinx toolkit, Sphinx4, an adjustable, modifiable recognizer written in Java, and SphinxTrain, a tool used for training and modelling acoustic models (“Overview of the CMUSphinx toolkit – CMUSphinx Open Source Speech Recognition,” n.d.). CMUSphinx toolkit is designed specifically for low-resource platforms, although, this toolkit focuses more on practical application development rather than research.

Furthermore, CMUSphinx enables their users to build their own phonetic dictionary with their g2p-seq2seq tool, a tool that is based on neural networks implemented in the Tensorflow framework and provides a state-of-the-art accuracy of conversion. CMUSphinx also allows their user to build their very own language model, adapts an existing acoustic

model as well as training them, and fine-tuning acoustic models through implementing functions that is provided in its libraries. Overall, the CMUSphinx is design made favour more towards application developers rather than researchers. CMUSphinx is still active in development and has an active community with more than 400 users to discuss or share resources with on its LinkedIn group account.

### **2.8.3 Nvidia NeMo**

Nvidia NeMo or Neural Module (Kuchaiev et al., 2019) is an open-source toolkit for researchers to develop state-of-the-art conversational A.I models. Although, it also consists of tools that can be used to develop speech recognition system. This toolkit is slowly building its reputation in the ASR development field with its ability to provide ease in designing a Neural Network model. Nvidia NeMo uses Python to build, train, and fine-tune GPU-accelerated conversational A.I models using a simple interface. NeMo ease developers in building their own model by providing them easy-to-use application programming interface (API). Using Tensor Cores in Nvidia GPU, NeMo runs mixed precision compute and can scale up to multiple GPU easily to produce highest training performance possible. NeMo consists of pre-built models for its users to use as template for further development or simply just to train their own personal dataset, as such, users could have a kick-start in developing their model.

Furthermore, NeMo also works closely with Pytorch Lightning library, a library that provide extremely useful tools for A.I model training and research. All of its function is implementable in NeMo due to high compatibility design of the toolkit towards Pytorch. This saves developers the time to design the complex training architecture of their ASR model. This is achieved via allowing developers to design their training model using a config YAML file that specifies the parameters and details of the model's structure. NeMo

possesses this feature whereby the architecture design of a training model is setup using a YAML file, easing developers in changing and improvising their model over-time without the hassle of removing or implementing a lot of components when changes is required. With the YAML feature, developers need only to change the details of the model's parameters in the YAML configuration file.

In addition, this feature also eases developers to build their own end-to-end ASR system whereby the ASR components are all connected to each other and simultaneous training can be done in all parts of the system. NeMo provides a collection of tutorials for its new users in developing end-to-end ASR that can be found in Google Colaboratory. With its YAML feature, this provides NeMo the flexibility to add and remove components with ease from its architecture. This compliments the design of an end-to-end ASR architecture where all components of the architecture are easily connected to each other and training is done simultaneously.

Finally, NeMo is currently still growing but in a fast rate as an ASR toolkit. It has its own Github repository for external use and a programming community, although it is still in a small number. This new toolkit does provide progressively updated tutorial for its users to learn the basic function and understanding of the tool.

#### **2.8.4 Google Colaboratory**

Google Colaboratory or better known as Google Colab, is an online cloud platform that uses Jupyter Notebooks to distribute machine learning education and research. It offers a pre-configured runtime environment specifically for deep learning and allows free access to a powerful GPU. A Jupyter notebook has the ability to function both locally on a computer or in a cloud-based environment. The notebook is made up of various cells, which can contain script language or markdown code, and the output is included in the document.

Examples of outputs that can be generated include text, tables, charts, and graphics. Utilizing this technology simplifies the sharing and reproduction of scientific works, as the experiments and results are presented in a self-contained format (Carneiro et al., 2018).

Google Colab notebooks are based on Jupyter and work as a Google Docs object where it allows multiple users to collaborate on the same notebook and can be shared. These notebooks come with pre-configured Python 2 and 3 runtimes that include essential machine learning and A.I libraries like TensorFlow, Matplotlib, and Keras. The virtual machine under the runtime is deactivated after a certain period, causing the loss of user data and configurations. However, the notebook is preserved, and users can transfer files from the virtual machine's hard disk to their Google Drive account. Google Colaboratory also offers a GPU-accelerated runtime that is fully configured with the previously mentioned list of software. The Google Colaboratory infrastructure is hosted on the Google Cloud platform.

## **2.9 Study gap on the previous existing Iban ASR model**

A Deep Neural Network (DNN) ASR model has been previously developed for the under-resourced Iban language using Kaldi speech recognition toolkit (Juan, 2015). The model takes advantage of DNN architecture with G2P pronunciation dictionary and language model. To our knowledge, the Iban language ASR system so far has only ever been developed using traditional pipeline of ASR development and no studies has been done towards developing it using CNN and end-to-end architecture. With the recent rise in trend as discussed in this chapter of using CNN end-to-end architecture in the development of ASR and the showcase of CNN's powerful feature extraction capability to overcome the weakness of data scarcity in under-resourced languages, this pique the interest of studying the proper framework or steps to implement end-to-end CNN architecture for the Iban language. Thus, we have decided to design a systematic framework for the development of

Iban ASR model using CNN end-to-end architecture as we discuss its methodology in the next chapter.

## **2.10 Chapter Summary**

In this chapter, we have conducted literature review and discussed existing work on the development of ASR as well as explored the latest trend and the most recent techniques being implemented into ASR development. We have studied the pipelines in developing an ASR system as well as the issues surrounding under-resourced languages in the development of speech recognition models. We investigated the how neural networks is being implemented as the acoustic model of ASR systems and the end-to-end architecture of CNN as a variant of ANN. We have also introduced the background of our targeted language, the Iban language and its cultural heritage. Furthermore, we have explained the process of how CNN learns information and extract features from a speech input and reviewed the strength of CNN end-to-end architecture in overcoming issues faced during the development of under-resourced languages. Additionally, we presented many existing works and efforts that has been done in the past and in recent years to improve the model performance of ASR models. Currently trending techniques being implemented which includes hyperparameter optimization, spectrogram augmentation, and transfer learning has also been studied in this chapter. Finally, we have also identified tools that is available to be used in modelling speech recognition system and developing CNN ASR models. In the next chapter, we will introduce our methodology and proposed framework to develop a CNN Iban ASR model. This displays how the techniques presented earlier is being implemented into our framework to develop an ASR for our under-resourced Iban language which would serve as solutions to our research questions and objectives.

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 Introduction**

This chapter describes the research' experiment implementation stage as well as its methodology. The main focus of this chapter is to describe the techniques proposed in our framework in optimizing a CNN-ASR model and to present the technical knowledge that we have recorded through experimenting with end-to-end CNN-ASR models. Also, it describes the steps taken by us in answering our research questions and objectives. The first section of this chapter presents the details of our proposed framework in building our very own Iban CNN-ASR model. Each stage in the framework will be discussed in detail on the next subsequent section which includes data preparation, preliminary experiment conduct, obtaining baseline performance, and model improvement stage. Furthermore, this chapter also discusses the method of evaluation and metrics as well as tools used to carry out all of the experiments.

#### **3.2 Dataset**

This section describes the dataset that we have used in our research. The Iban dataset is our target language while the Malay and English dataset was used in our third technique, cross-lingual transfer learning.

##### **3.2.1 Iban (Target Language)**

The Iban language is an under-resourced language from Sarawak, Malaysia. For this research, we have chosen the Iban language as our target language. We use the corpus that was previously collected in (Juan, 2015) as our Iban dataset.

The corpus mainly consists of news data that were gathered from Radio Televisyen Malaysia (RTM) Berhad, a local news station in Malaysia. A total of 3GB audio data for pre-processing was extracted, but due to the rest being poor in quality, i.e., long music commercial breaks, excessive distortion, and unclear speeches, only 1.1GB was viable to be used. The news data were transcribed into text via a data collection workshop that was organized in the Faculty of Computer Science and Technology, Universiti Malaysia Sarawak. In total, 3000 sentences uttered by 23 speakers in 8 h of clean speech were gathered during the workshop (Juan, 2015).

### **3.2.2 Malay**

The Malay language is a language that is closely related to the Iban language (Juan et al., 2015), and is also the national language of the country Malaysia. Both Malay and Iban language belongs to Austronesian stock, specifically, the proto-Malay-Iban parent language (Rohani, 2003). It is one of the source languages we have chosen to be used for our cross-lingual transfer learning.

Our Malay dataset was obtained from a Malay text-to-speech GitHub repository whereby we collected a size of 31.2GB of audio samples from the main repository to be used as dataset. The dataset composed of sentences ranged between 2 to 20 words. It comprises of two speakers and was divided with equal male to female ratio with 1:1 male to female speaker. Specifically, it consists of 99.4h of female audio samples and 94.5h of male audio samples which accumulates to a total of 193.9h of speech data. Source of dataset can be found from: <https://github.com/huseinzol05/malaya-speech/tree/master/data/azure-tts>.

### **3.2.3 English Pre-trained Model**

The English language is a universal language and is the second source language used in our cross-lingual transfer learning technique. For the English source language, we imported a pretrained model which has been trained previously on six English speech datasets totalling a 7057 hours of audio data. The model was previously trained on these datasets: 2903h LibriSpeech read English speech dataset, 245h Wall Street Journal read English speech dataset, 1906h Fisher English Training Speech (conversation English speech dataset), 316h Switchboard conversational English speech dataset, 1090h Mozilla Common Voice read English speech dataset, and 1857h NSC Singapore read English (Part 1) speech dataset (Kriman et al., 2020)

## **3.3 General Experimental Setup**

This section presents the initial procedure we have taken to implement the framework we have proposed as well as our tools and materials.

### **3.3.1 Preparation of Dataset**

In preparing our Iban dataset, a total of 473 lines of Iban sentences were chosen for testing and another 2659 lines of sentences for training. We distributed our training and test set into a ratio of around 50:9. For the test set, the Iban corpus comprised a total speaker of 6, a male-to-female ratio of 1:2, 473 sentences, 11,000 words, and a total of 71 min (~1H) audio data duration. For the train set, 17 speakers were in total, a male-to-female ratio of 7:10, 2659 sentences, 61,000 of words, and a total of 408 (~7H) min audio data duration.

For the Malay dataset, we divided the corpus into 75k spoken lines for training set and 25k lines for testing set from the total line of sentences. The training and test set has a ratio of 3:1. For its training set, the Malay dataset comprised of ~71H and ~75H audio

duration for male and female speakers respectively, accumulating around 146H audio data. On the other hand, for the test set, it comprised of ~23H and ~24H audio duration for male and female speakers, accumulating around 47H audio data. The number of male to female speakers for each set is divided into equal ratio of 1:1 since there is only 1 speaker for each gender. Table 3.1 describes the detail of the distribution of the Iban and Malay dataset training and testing set.

**Table 3.1:** Distribution of Train and Test set for the Iban and Malay Dataset

| Dataset | Set   | # Of Lines | Audio Duration |
|---------|-------|------------|----------------|
| Iban    | Train | 2659       | ~7H            |
|         | Test  | 473        | ~1H            |
| Malay   | Train | 75000      | ~146H          |
|         | Test  | 25000      | ~47H           |

There were no distribution of train and test set for the English dataset as we only used it in a pretrained model imported from an open-source repository. Thus, there was not a need for data preparation.

### 3.4 Framework to improve CNN Iban ASR through optimization approaches.

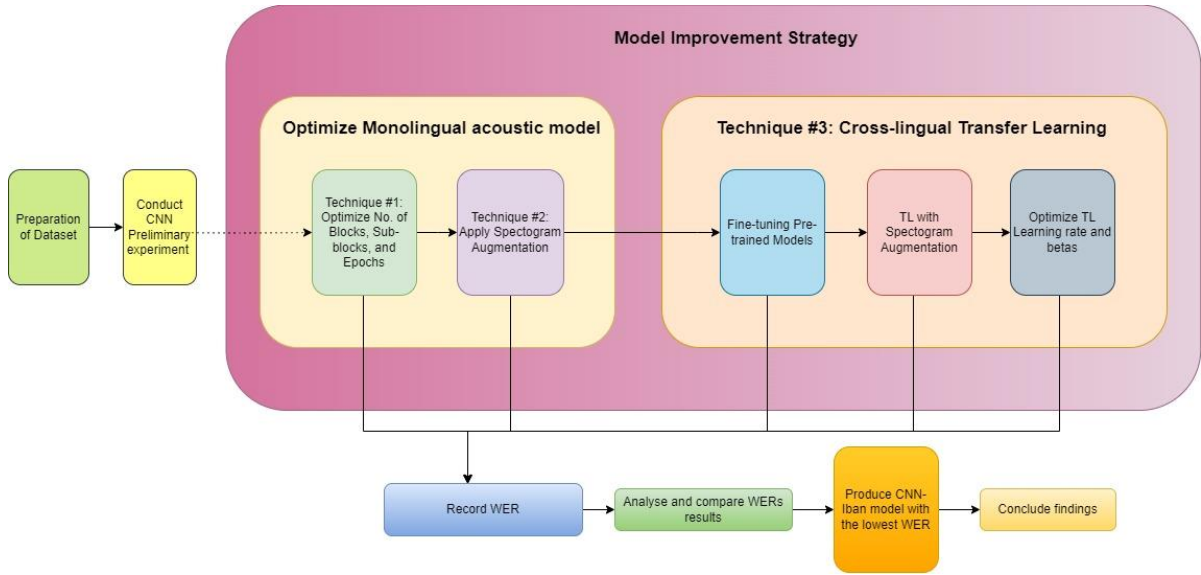


**Figure 3.1:** Iban ASR Framework

The proposed framework is illustrated as shown in Figure 3.1. We divide it into three major parts which are: obtaining initial baseline result, improving the CNN model, and analysing Word-Error-Rates (WERs) from the proposed techniques. The first part involves the process of preparing data for initial experiment setup and conducting preliminary experiment to obtain baseline result, the second part of the framework introduces our proposed techniques in our model improvement strategy to further improve the baseline result, finally the third part of the framework involves analysing WERs obtained after implementing the proposed techniques. The section below explains our framework.

**I) Preparation of Dataset** – The early stage of the framework involves the distribution of train and test set. It is to determine the training and testing size of our model. The allocation of training and testing set allows for the assess of model performance in the later stage. The size of the training set will represent the resource-richness of our model.

**II) Conduct Preliminary Experiment** – The start of the research begins with the conduct of an initial experiment. We investigate the architecture of the CNN for the first time as well as to obtain a baseline performance. To observe the initial performance of the CNN as an acoustic model, we run a preliminary experiment using its most basic setup. Conducting such experiment allow us to observe how the CNN operates and also to obtain our initial WERs, which serves as the starting point of our research. The result obtained during this stage represents our model baseline performance.

### **3.4.1 Monolingual Approach**

After obtaining a baseline performance, we focus on the model improvement strategy to enhance the baseline performance of our CNN ASR model. Our strategy consists of two approaches: monolingual and cross-lingual. We propose three techniques to improve our model, the first two techniques mainly focus on monolingual approach while the third technique implements cross-lingual approach.

**III) Optimization of Block, Sub-block, and Epoch (Technique #1)** – The first technique in improving the model is the optimization of the number of blocks, sub-blocks, and epochs. We implement this technique to obtain us the most optimal model using the CNN architecture. A block in CNN architecture consists of multiple convolutional layers which extract features. It is where the process of gaining new knowledge occurs for the CNN. In theory, the more extracting mechanism the CNN possess e.g.: higher no. of blocks, the more features it will be able to extract and the higher the accuracy of speech recognition. Sub-block is the repetition process of feature extraction done by the layers in the block. Theoretically, it helps the model to detect previously missed feature from the input and increase the accuracy of the model. Meanwhile, Epoch represents the total completion of training rounds done on the whole dataset.

All three hyperparameters influences the accuracy of the model prediction. However, the advantages are limited to a certain extent and that the limit varies from one architecture to another. It is observed that in some cases when the increase in one hyperparameter value is much higher as compared to the other two hyperparameters, the model performance degrades instead or increase rather in a slower rate. Thus, to obtain the most optimal CNN architecture for the targeted language, we should investigate and optimize these hyperparameters as a combined architecture.

**IV) Apply Spectrogram Augmentation (Technique #2)** – Secondly, to improve the model performance without the need of extra speech data, we propose the application of spectrogram augmentation to our model training as our second technique. Spectrogram augmentation increases the robustness of under-resourced language via masking non-meaningful input features. To increase the accuracy of our model, we implement spectrogram augmentation to mask the unnecessary features in our already low resourced dataset. This allows us to fully and efficiently use the resource that we currently possess. It also helps us to overcome the need to gather extra speech resource for model improvement, which would result in a difficult task.

### **3.4.2 Cross-lingual Approach**

The third technique leverages on using other languages to improve the prediction accuracy of our model. Different from the first two techniques, the third proposed technique relies on other pretrained model of different languages in order to produce better WER result.

**V) Cross-lingual Transfer Learning (Technique #3)** – In our third technique, to further increase our model performance as well as reduce the reliance on the abundance of speech data, we conduct cross-lingual transfer learning to the targeted language corpus. We utilize the knowledge learned previously by a pre-model into our targeted language model. For

CNN model, the accuracy of transcription increases as more input feature is learned, thus, implementing transfer learning gives our model a head-start and helps to eliminate the requirement for the abundance of speech data.

Furthermore, to achieve the best result from transfer learning, we conduct optimization which is via applying spectrogram augmentation that would provide us the aforementioned advantages and adjusting its beta values as previous study has shown that the transfer learning's learning curve is dependent on its learning rate and beta values.

**VI) Record WER** – We record all the WER produced by each experiment for model performance evaluation. WER will serve as data for the assessment of the model performance in the later stage of the framework and also as documentation for future work.

**VII) Analyse and Compare WER results** – After the improvement stage, we conduct analysis and comparison on the WER produced by the model in each implementation of techniques as to investigate its effectiveness. Furthermore, to assess the performance of CNN end-to-end architecture as an acoustic model to an ASR, we conduct comparison of WER with previous work which involves the development of ASR for the Iban language.

**IX) Produce Iban CNN ASR model with lowest WER** – As the final product of the research framework, we choose techniques that helps us to achieve the lowest WER possible during this research and determine its order of execution. We do it for the best model and best monolingual model obtainable from using the framework.

**X) Conclude Findings** – We determine whether the framework has helped us to obtain our targeted language CNN ASR model to conclude our research.

### 3.4.3 Speech Recognition Engine and Tools

There were several tools that was used during the implementation of our framework which includes the Nvidia Neural Module, Google Colaboratory, and Python programming language.

#### I) Nvidia Neural Module (NeMo) Toolkit

As mentioned in section 2.8.3 of CHAPTER 2, the Nvidia NeMo is an A.I conversational toolkit that helps developers to build neural network models. It includes three major collections that could help developers to design an ASR model. These are Automatic Speech Recognition (ASR), Natural Language Processing (NLP), and Text-to-Speech (TTS) models. Each collections have its own prebuilt modules that can be easily customized, extended, and composed in creating new model architecture and for developers to train their data.

We used the Nvidia NeMo ASR module to design our model architecture. It enabled us the ability to configure and design our model architecture through feature specifications in a YAML config file. This granted us the flexibility to design our CNN architecture without the hassle of spending too much time in configuring our model manually. The content of the YAML config file has an entry labelled as “encoder” with a field called “jasper,” where it comprised members that specify the criteria of one block as such:

*filters: 128*

*repeat: 1*

*kernel: [11]*

*stride: [2]*

*dilation: [1]*

*dropout: 0.2*

*residual: false*

*separable: true*

*se: true*

*se\_context\_size: -1*

Each member in a block described above corresponds to specific components in a model. By specifying values in each of the parameters, we could design our model architecture based on our preference.

## **II) Google Colaboratory**

Google Colaboratory acts as our programming environment, where it provides us with space to code and can later be saved as Google Doc. We coded our model in the cells of a Google Colaboratory notebook that was based on Jupyter Notebook and integrated it with our local GPU for experimentation. We separate our code into different cells according to their functionalities. This manner of programming allows us to implement and run the codes either separately or simultaneously.

## **III) Python**

In the research experiment, Python is our main programming language. It is a versatile language that is compatible with multiple development methodologies and has extensive standard libraries that can be expanded upon. The language is known for its simplicity, being freeware and open-source, higher-level programming language, platform independence, and interactive typing. Almost all of our model structure is coded using

Python with the exception of the YAML configuration files, the model's hyperparameters and several other smaller programming modules.

### 3.5 Evaluation Metrics of ASR models

Our method to evaluate the performance of an ASR is through the observation and analysis of the collection of Word-Error-Rate (WER) results. WER serves as a standard metrics for evaluating the performance of Large Vocabulary Continuous Speech Recognition (LVCSR) system (Ali & Renals, 2020) such as the CNN that will be built for this research. In general, the formula to get a WER is:

$$\text{WER} = (S+D+I)/N = (S+D+I)/(S+D+C)$$

Where S is the number of Substitutions, D is the number of Deletions, I is the number of Insertions, C is the number of correct words, and N is the number of words in the reference. In a simpler form, WER is basically the numbers of error obtained after a transcription process of a trained ASR model divided by the total words used in reference for its model training. As for the errors, a Substitution is the event when a word from the reference gets replaced by other word, for example the word "Him" was recorded by the system as "Hymn". Next, as for the occurrence of Deletion, it is when a word is left out from the transcript completely, e.g., the phrase "I have an apple" was transcribed by the system as "I have apple". In this example, the phrase has undergone Deletion for the word "an". Finally for Insertion, as what the name implies, it is when a word is inserted that was not in the reference. An example would be when the system transcribed an input as "essay tea", but the original phrase was supposed to be "SAT". The word "essay" would be considered as a Substitution, but the word "tea" is considered as an Insertion as the original phrase only consists of one word.

Thus, summing up the total errors obtained from the transcription of an ASR divided by the total amount of words used as reference would give us the percentage of how accurate the system was able to perform. WER will be the main metrics to be investigated, analysed, and compared in evaluating the performance of ASR systems that is designed and build during the research.

### **3.5.1 Performance Measurement Tools**

As mentioned in the previous section, for the evaluation of model performance, we observe the differences in WER achieved by our model. This is done by importing the Speech Recognition Scoring Toolkit (SCTK) module to help us in accuracy measurement of our model's text transcription.

#### **Speech Recognition Scoring Toolkit (SCTK)**

To evaluate the performance of ASR models trained from our experiment, we imported the SCTK as a third party WER scoring toolkit into the Google Colaboratory notebook. The toolkit allows us to retrieve an in-depth analysis of our model's transcription performance provided that we possess a hypothesis (result of model transcription) and reference (ground truth text of the transcribed audio inputs) file. Upon loading the two files, the SCTK will analyse the differences that occurs in both files. Based on the result, the toolkit will produce 4 output files which holds the information regarding the accuracy of the model transcription and other detail such as the error occurred in each specific sentence.

### **3.6 Preliminary Experimental Setup**

Following the next stage of our framework, we proceed to the conduct of preliminary experiments. The purpose of this stage is to investigate the CNN architecture for the first time and to obtain a baseline model performance, thus, we setup our experiment focusing on

investigating two objectives: to investigate the dependency of CNN model towards the amount of resource needed for model training (Experiment I) and to perform sensitivity analysis on the CNN parameters (Experiment II).

For the first objective, we took several samples of train and test set from the Iban corpus to be used in a speech recognition task. The second objective on the other hand was more focused on increasing the values of hyperparameters.

### 3.6.1 Preliminary experiment - Experiment I

In Experiment I, to investigate the effect of the amount of resource towards the performance of a CNN ASR model, we increased the number of sentences in the train set incrementally throughout the experiment while keeping the number of sentences in the test set at a fixed amount. We randomly selected a sample of 2000 sentences for the train set and 10 sentences for the test set, the protocol for the train set is as shown in Table 3.2.

**Table 3.2:** Protocol of Experiment I -Train Set

| <b>Train set size (sentence)</b> | <b># Speakers</b> | <b># Female speakers</b> | <b># Male speakers</b> | <b># Sentences spoken by female</b> | <b># Sentences spoken by male</b> |
|----------------------------------|-------------------|--------------------------|------------------------|-------------------------------------|-----------------------------------|
| 200                              | 10                | 5                        | 5                      | 100                                 | 100                               |
| 400                              | 10                | 5                        | 5                      | 200                                 | 200                               |
| 600                              | 10                | 5                        | 5                      | 300                                 | 300                               |
| 800                              | 11                | 6                        | 5                      | 400                                 | 400                               |
| 1000                             | 13                | 7                        | 6                      | 500                                 | 500                               |

**Table 3.2** continued

|      |    |    |   |      |      |
|------|----|----|---|------|------|
| 1200 | 13 | 7  | 6 | 600  | 600  |
| 1400 | 15 | 8  | 7 | 700  | 700  |
| 1600 | 16 | 9  | 7 | 800  | 800  |
| 1800 | 16 | 9  | 7 | 900  | 900  |
| 2000 | 17 | 10 | 7 | 1000 | 1000 |

As observed from the table, we first use only 200 sentences, before we subsequently add another 200 lines of sentences until it reaches the total amount of 2000 lines. On each increment, we tried to equalize the ratio between male and female speakers. In total, the train set consists of 17 speakers with 1000 sentences spoken by both male and female speakers as displayed in Table 3.2. Following the protocol, we ran a model training for a hundred epoch on each increment. We decided to opt for a very basic CNN end-to-end model architecture for this setup as it is our first experiment. Table 3.3 summarize the structure of our model in the preliminary experiment I.

**Table 3.3:** Preliminary Experiment I Model Architecture

| Experiment   | Model     | # of Block | # of Sub-block | # of Epochs |
|--------------|-----------|------------|----------------|-------------|
| Preliminary  | Quartznet | 4          | 1              | 100         |
| Experiment I | CNN       |            |                |             |

The architecture used was a Quartznet CNN model that consist of 4 blocks and a single sub-block repetition. Full detail of the YAML file of the model architecture is specified in APPENDIX B. Once the model training has completed, we extracted two types of files from

the training which was the model and checkpoint file. Each file is used for fine-tuning and performance evaluation in the later stage.

To further investigate the first objective, we have also ran another round of experiment with the increased number of sentences in the test set. Table 3.4 describes the protocol of the increased size of test set.

**Table 3.4:** Protocol of Experiment I – Test Set

| <b>Test set<br/>size<br/>(sentence)</b> | <b>Total # of<br/>speakers</b> | <b># of female<br/>speakers</b> | <b># of male<br/>speakers</b> | <b># of lines<br/>spoken by<br/>female</b> | <b># of lines<br/>spoken by<br/>male</b> |
|-----------------------------------------|--------------------------------|---------------------------------|-------------------------------|--------------------------------------------|------------------------------------------|
| 100                                     | 6                              | 4                               | 2                             | 50                                         | 50                                       |
| 200                                     | 6                              | 4                               | 2                             | 100                                        | 100                                      |
| 300                                     | 6                              | 4                               | 2                             | 150                                        | 150                                      |
| 400                                     | 6                              | 4                               | 2                             | 200                                        | 200                                      |

Similar to the train set experiment, the test set was incrementally increased. We increase 100 lines of sentences on each increment up to 400 lines. On each increment however, the male to female speaker ratio is 1:2. Increasing the size of the test set helps us to investigate whether the performance of our model remained consistent on each increase of transcription tasks. During this round of experiment, we kept the size of our train set at 2000 line of sentences.

### **3.6.2 Preliminary experiment – Experiment II**

Preliminary Experiment II focused on performing sensitivity analysis on the hyperparameters that exists in the CNN architecture. This investigates the impact of the change of values in the parameters towards the CNN model performance in transcribing

speech data. For the second preliminary experiment, we have experimented with the hyperparameter, epoch.

Using the model architecture in Experiment I, we ran a model training using the same train and test set of 2000 and 400 lines of sentences respectively with an increasing amount of epoch. Table 3.5 summarizes the model architecture used in Preliminary Experiment II.

**Table 3.5:** Preliminary Experiment II Model Architecture

| Experiment    | Model     | # of Block | # of Sub-block | # of Epochs |
|---------------|-----------|------------|----------------|-------------|
| Preliminary   | Quartznet | 4          | 1              | 500 - 5500  |
| Experiment II | CNN       |            |                |             |

For Experiment II, the epoch acted as the hyperparameter that determines the performance of our model. In this manner, we were able to analyse the sensitivity of the CNN hyperparameter via the increase and decrease of the epoch's value. We recorded the WER produced by our model on each increment of 500 until an epoch of 5500 to identify the best result as our baseline Iban CNN ASR performance.

### 3.7 Optimizing Iban CNN Model – Proposed Techniques

We proposed three techniques to optimize our CNN ASR model in which it does not involve the process of increasing our target language' speech data. The three proposed techniques are CNN's hyperparameter optimization, application of spectrogram augmentation, and cross lingual transfer learning. The first two technique was inspired by the works of Aszemi & Dominic (2019) where the author attempted to optimize the CNN via hyperparameter optimization and data augmentation while the third technique was

inspired by the works of Huang et al. (2020) where they carried out transfer learning using various different language on the Quartznet CNN.

We first started investigating the potential improvement that could be achieved by our CNN model through the optimization of its hyperparameter. We identified three hyperparameters from the CNN architecture which are: block, sub-block, and epoch, then configured experiments setup focusing on these hyperparameters.

### 3.7.1 Block Optimization

To optimize the number of blocks in a CNN ASR model, we ran an experiment using the Quartznet B×1 configuration where B represents the number of blocks. In this experiment, we increase the number of blocks until a specified maximum value and observe the changes in performance if any produced by our model throughout the experiment. Table 3.6 summarize our model architecture while Figure 3.2 illustrates an example of one of the architectures of the model that was used during the Block experiment.

**Table 3.6:** Block Experiment's Model Architecture

| Experiment         | Model            | # of Block | # of Sub-block | # of Epochs |
|--------------------|------------------|------------|----------------|-------------|
| Block Optimization | Quartznet<br>CNN | 1-20       | 1              | 100         |

The value of B was set to a range of 1 to 20. Each block in the model was set to only have a single sub-block but the number of blocks was increased incrementally throughout the sessions.

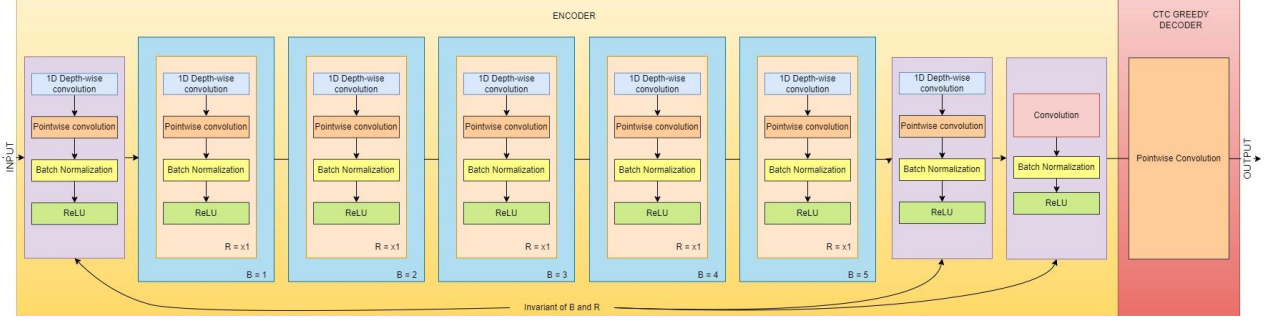


**Figure 3.2:** Quartznet 5x1 from B×1 architecture

The total amount of blocks that can be found in this architecture follow the structure of  $1 + \chi[B \times R] + 2$  blocks which includes the first and the last two subsequent block where all three is invariant of  $B \times R$  block. The  $\chi$  holds the value of the total number of  $B \times R$  variant block in the current model in which for the case of Figure 3.2, it was set to the value of 5 which corresponds to 5 number of blocks. We conducted experiment starting from the value  $\chi = 1$  to 20, where on each increments the model is trained for a hundred epoch. The value for the number of blocks that produced the highest accuracy of transcription were recorded and taken as the most optimal value. A detailed information regarding the structure of the YAML file for this model architecture is attached in APPENDIX C.

### 3.7.2 Sub-block Optimization

To optimize the number of sub-blocks, we configured the model architecture to differ only slightly from the design that was implemented previously in the block experiment. For the sub-block experiment, our model follows the configuration of Quartznet 1×R where R represents the number of sub-blocks. In this experiment, the number of sub-blocks becomes the focus where we increased its number until a specified maximum value and observe the changes in performance throughout the experiment, similar to what was done during the block optimization. Table 3.7 summarize our model architecture while Figure 3.3 shows an

example of one of the architectures of the model that was used during the sub-block experiment.

**Table 3.7:** Sub-block Experiment's Model Architecture

| Experiment   | Model     | # of Block | # of Sub-block | # of Epochs |
|--------------|-----------|------------|----------------|-------------|
| Sub-block    | Quartznet | 1          | 1-20           | 100         |
| Optimization | CNN       |            |                |             |

Based on Table 3.7, we kept the number of blocks,  $B$ , to be equal to 1, while increasing the number of sub-blocks,  $R$ , from 1 to 20. Furthermore, in each increment, we train the model for a hundred epoch.

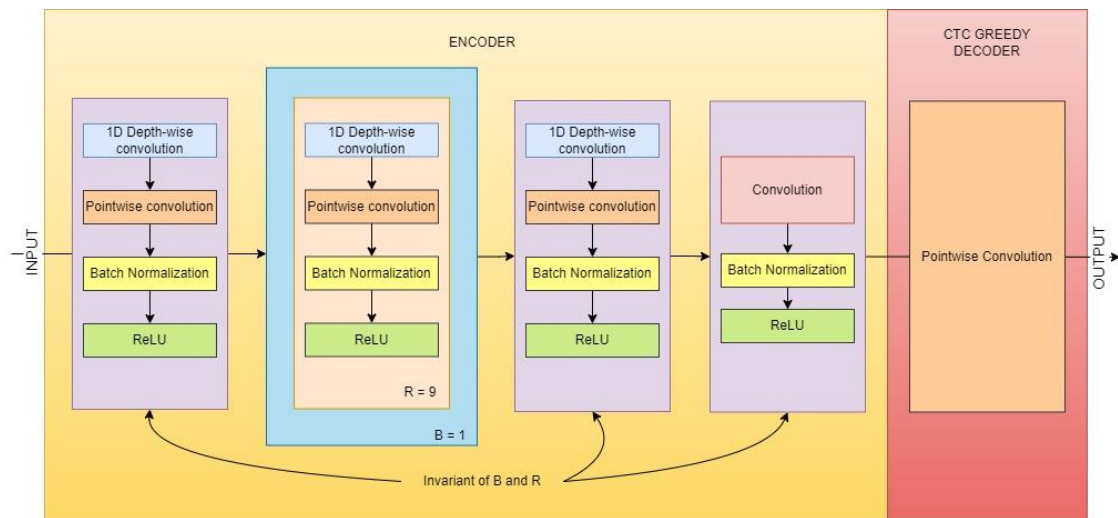


**Figure 3.3:** Quartznet  $1 \times 9$  from  $1 \times R$  architecture

For the sub-block experiment as observed in Figure 3.3, our model architecture follows the structure of  $1 + (1 \times R) + 2$  blocks where total number of blocks that can be found is 4. The variable  $R$  holds the value of the total number of sub-blocks in a  $B \times R$  variant block. For the case of Figure 3.3, it was set to the value of 9 which corresponds to 9 number of sub-blocks. Similar to the block experiment, the number of sub-blocks that produced the highest

accuracy of transcription would be recorded to be taken as the most optimal value. A detailed information regarding the structure of the YAML file for this model architecture is attached in APPENDIX D.

### 3.7.3 Epoch Optimization

For the third hyperparameter, epoch, we used a Quartznet 1×1 configuration which represent a basic model with 1 block and 1 sub-block. Also, we used a Quartznet 20×1 model that we have obtained from the previous experiments, in which the model represents an optimized model with 20 blocks and single sub-block. By doing this, we are able to investigate the impact of epoch on a model with simple-complexity and a model with extreme-complexity as we increase its values. Quartznet 1×1 architecture is similar to that of the Quartznet 1×9 structure which was depicted in Figure 3.3 in the sub-block experiment. The Quartznet 20×1 model architecture is presented in Figure 3.4. Table 3.8 summarizes the model architecture that was used during the epoch optimization experiment.

**Table 3.8:** Epoch Experiment's Model Architecture

| Experiment         | Model            | # of Block | # of Sub-block | # of Epochs |
|--------------------|------------------|------------|----------------|-------------|
| Epoch Optimization | Quartznet<br>CNN | 1          | 1              | 100 - 1000  |
| Epoch Optimization | Quartznet<br>CNN | 20         | 1              | 100 - 1000  |

As observed in Table 3.8, we kept the number of blocks and sub-blocks of both model at a fixed number while incrementally increase the number of epochs.

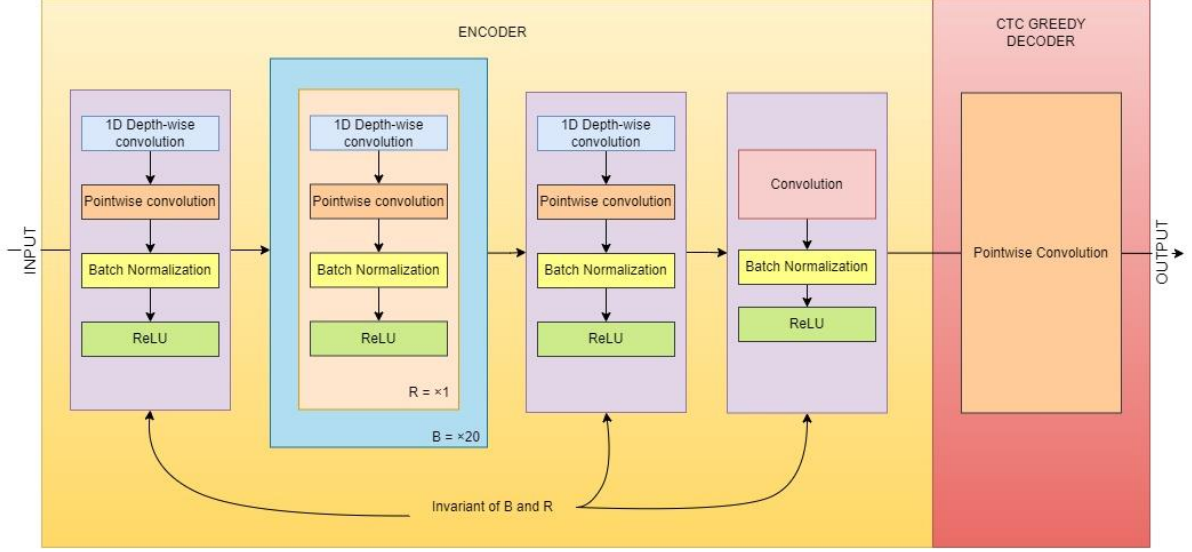


**Figure 3.4:** The optimized  $B \times R$  - Quartznet  $20 \times 1$  Model Architecture

The Quartznet  $20 \times 1$  model is an optimized  $B \times R$  model that we have obtained from the block experiment in which its number of blocks equals to 20 and sub-blocks equals to 1. The model totals a number of 23 blocks including the first and the last two invariant of  $B \times R$  blocks.

The number of blocks and sub-blocks for both model that was used throughout the epoch experiment were kept the same. However, we trained them with an increasing amount of epoch. The training consists of 10 increments that amounts to 100 epochs each, which totals to a thousand epoch. The number of epochs that helped our model to achieve the best transcription accuracy was recorded and is considered to be the most optimal value. A detailed information regarding the structure of the YAML file for both model architecture that was used during this experiment implementation has been attached in APPENDIX E and F.

### 3.7.4 Spectrogram Augmentation

Following the optimization of hyperparameters, we proposed a second technique to enhance our CNN ASR prediction accuracy, that is the implementation of spectrogram

augmentation. This technique is based on the works done by Park et al. (2019) and DeVries & Taylor (2017) as described in chapter 2 in 29. Two types of spectrogram augmentation were implemented to our model, the two augmentations are rectangle masking or also known as “cutout” as presented by DeVries & Taylor (2017), and time-frequency masking or also known as SpecAugment by Park et al. (2019).

Since the Iban language is an under-resourced language which faces the issue of data scarcity, thus, we decided to use spectrogram augmentation to mask non-meaningful data whilst increasing the robustness of our train set, avoiding the necessity to increase our speech data to improve model performance. To investigate the impact of both rectangle and time-frequency masking augmentations towards our model performance, we ran a model training implementing the two augmentations using the optimized Quartznet 20×1 model. We begin the augmentation experiment starting with the rectangle masking first followed by the time-frequency masking augmentation.

#### 3.7.4.1 Rectangle Masking (SpecCutout)

Rectangle masking also known as ‘Cutout’ which was based on DeVries & Taylor (2017) zeroes out rectangles in spectrograms. To implement rectangle masking, we specified values into several parameters that shapes our masks. We first have to define the size of our rectangular cuts that would masks along the spectrogram’s frequency and time dimensions as well as the number of rectangular masks to be implemented in each given spectrogram. Table 3.9 displays our model architecture during the implementation of the rectangle masking experiment while Table 3.10 displays the value set for each of the augmentation’s hyperparameter used in the rectangle masking augmentation experiment.

**Table 3.9:** Rectangle Masking Experiment's Model Architecture

| Experiment   | Model     | # of Block | # of Sub-block | # of Epochs | Spectrogram Augmentation |
|--------------|-----------|------------|----------------|-------------|--------------------------|
| Spectrogram  | Quartznet | 20         | 1              | 900         | Rectangle                |
| Augmentation | CNN       |            |                |             | Masking                  |

**Table 3.10:** Hyperparameter values set in the rectangle masking experiment

| Hyperparameter                                               | Value |
|--------------------------------------------------------------|-------|
| Max size of the cut rectangles along the Frequency dimension | 50    |
| # of rectangular masks to be cut from any given spectrogram  | 5     |
| Max size of cut rectangles along the Time dimension          | 120   |

Using rectangle masking, we trained our optimized Quartznet CNN 20×1 model for 900 epochs. The value observed in Table 3.10 was set following the default value recommended by our speech engine. Throughout the experiment, we recorded the WER of our model on every 100 epochs for 9 increments and then analysed its trend to determine the most optimal number of epochs needed by our model to be trained while implementing rectangle masking augmentation.

### 3.7.4.2 Time-Frequency Masking

Time-Frequency (TF) masking augmentation was based on the works of Park et al. (2019). SpecAugment, which was presented by the author zeroes out vertical and horizontal sections of spectrograms through the implementations of frequency and time masking. Similar to the rectangle masking augmentation, we have specified values to several hyperparameters that defines our augmentation masks for the time-frequency masking augmentation. Table 3.11 displays our model architecture during the implementation of the time-frequency masking experiment while Table 3.12 displays the values set for each hyperparameter used in the time-frequency masking augmentation.

**Table 3.11:** Time-Frequency Masking Experiment' Model Architecture

| Experiment               | Model         | # of Block | # of Sub-block | # of Epochs | Spectrogram Augmentation |
|--------------------------|---------------|------------|----------------|-------------|--------------------------|
| Spectrogram Augmentation | Quartznet CNN | 20         | 1              | 900         | Time-Frequency Masking   |

**Table 3.12:** Hyperparameter values set in the time-frequency masking experiment

| Hyperparameter                                | Value |
|-----------------------------------------------|-------|
| # of frequency segments to be cut             | 1     |
| Max # of frequencies to be cut in one segment | 5     |

**Table 3.12** continued

|                                                 |   |
|-------------------------------------------------|---|
| # of time segments should be cut                | 1 |
| Max # of time steps to be cut in one<br>segment | 5 |

As observed from Table 3.11, we conducted experiment similar to the previous augmentation technique to investigate the impact of SpecAugment towards our model. We used the optimized Quartznet CNN 20×1 model to be trained for 900 epochs but with the time-frequency mask instead. The hyperparameter values shown in Table 3.12 was the most optimal value obtained after we conducted several rounds of experiment using the aforementioned masking technique whereby the model was able to produce the highest accuracy of transcription. To determine the most optimal number of epochs needed by our model to be trained with time-frequency masking, we recorded the WER produced by our model on every 100 epochs for 9 increments, similar to the rectangle masking experiment, and then analysed its trend. Each augmentation was implemented separately on two different experiments; thus, two ASR model was produced. The full information regarding the structure of the YAML file for the model architecture used in the spectrogram augmentation experiment is attached in APPENDIX G and H.

### 3.7.5 Cross-lingual Transfer Learning Method

The previous two proposed techniques revolved around monolingual approach where we only involved our Iban dataset in the process of fine-tuning our model. The third technique proposed is cross-lingual transfer learning (TL) method which involves the use of source languages to further improve our model prediction accuracy. We investigate the effectiveness of transfer learning practices towards under-resourced language end-to-end

CNN-ASR models using two different pre-trained models as our source model. One was previously trained using a closely related language (Malay) while the other was a non-closely related language (English) ASR model. We carried out transfer learning using the two models under three settings: basic TL, TL with spectrogram augmentations, and TL with optimized beta. To investigate the impact of each setting, we collected the result of each model's performance after each knowledge transfer and observed the change in WER experience by our model. As mentioned previously, for the English source language we used a pre-trained model named Speech-to-Text English Quartznet 15×5 (STTQznet) model. For the Malay dataset, we first trained our very own Malay 20×1 Quartznet CNN model, before using it in the third technique as our source model for closely related language.

#### 3.7.5.1 Pretrained Model – STTQznet model

The Speech-to-Text English Quartznet 15×5 (STTQznet) model is a pre-trained model that can be loaded from the Nvidia NeMo repository. The model has been previously trained on a combination of six English speech datasets totalling a 7057 hours of audio samples for an epoch of 300 with Apex optimization level of 01 (“STT En Quartznet15x5 | NVIDIA NGC,” n.d.). It is available to be used in the NeMo toolkit and can be used as a pre-trained checkpoint for inference or for fine-tuning on another dataset. The model consists of 79 layers that totals an 18.9 million parameters, with fifteen blocks that repeats five times plus four additional convolutional blocks.

#### 3.7.5.2 Bahasa Malaysia (Malay language) model

For the Malay dataset, we trained our own model using the Nvidia NeMo toolkit. We imported an open-source Malay dataset repository that we found online to be used as its training data. Our Malay model follows the architecture of Quartznet CNN trained using CTC loss but with an amount of 20 blocks that repeats only 1 time plus three additional

convolutional blocks. It consists of 64 layers that totals 3.7 million parameters. The model was trained for 10 epochs which results with a WER of 0.08.

### 3.7.5.3 Basic TL

We started off our third proposed technique using basic transfer learning settings. This was done by simply carrying out training on both STTQznet and Malay model using our Iban dataset without any additional settings. We imported both models into our notebook before we began transfer learning. Table 3.13 describes our model architecture when we conducted the transfer learning experiment using basic settings.

**Table 3.13:** Basic TL's Model Architecture

| <b>Experiment</b> | <b>Model</b>            | <b>Dataset</b> | <b># of<br/>Block</b> | <b># of<br/>Sub-<br/>block</b> | <b># of<br/>Epochs</b> | <b>Spectrogram<br/>Augmentation</b> |
|-------------------|-------------------------|----------------|-----------------------|--------------------------------|------------------------|-------------------------------------|
| Basic TL          | STTQuartznet<br>EN 15×5 | English        | 15                    | 5                              | 100                    | None                                |
| Basic TL          | Quartznet<br>Malay 20×1 | Malay          | 20                    | 1                              | 100                    | None                                |

We conducted transfer learning using both models on the Iban dataset for 100 epochs each, there were no other additional settings being implemented during the conduct of this round experiment.

#### 3.7.5.4 TL with Spectrogram Augmentation

Next, we applied spectrogram augmentation to our basic TL method. We applied both rectangle and time-frequency masking into the transfer learning. We also investigated the performance of our model after carrying out transfer learning with both masking techniques combined. Table 3.14 displays the value for each hyperparameters used in each of the three TL-spectrogram augmentation settings while Table 3.15 summarizes the model architecture used during the experiment for implementation of transfer learning with spectrogram augmentation.

**Table 3.14:** Hyperparameter values set for each setting in TL with spectrogram augmentation experiment

| <b>Hyperparameter</b>                                        | <b>Rectangle<br/>Masking</b> | <b>Time-<br/>Frequency<br/>Masking</b> | <b>Combined</b> |
|--------------------------------------------------------------|------------------------------|----------------------------------------|-----------------|
| Max size of the cut rectangles along the Frequency dimension | 50                           | 0                                      | 50              |
| # of rectangular masks to be cut from any given spectrogram  | 5                            | 0                                      | 5               |
| Max size of cut rectangles along the Time dimension          | 120                          | 0                                      | 120             |
| # of frequency segments to be cut                            | 0                            | 1                                      | 1               |

**Table 3.14** continued

|                                               |   |     |     |
|-----------------------------------------------|---|-----|-----|
| Max # of frequencies to be cut in one segment | 0 | 27  | 27  |
| # of time segments should be cut              | 0 | 1   | 1   |
| Max # of time steps to be cut in one segment  | 0 | 100 | 100 |

**Table 3.15:** TL with Spectrogram Augmentation's Model Architecture

| Experiment | Model                   | Dataset | # of Block | # of Sub-block | # of Epochs | Spectrogram Augmentation                                                            |
|------------|-------------------------|---------|------------|----------------|-------------|-------------------------------------------------------------------------------------|
| Basic TL   | STTQuartznet<br>EN 15×5 | English | 15         | 5              | 100         | Rectangle Masking,<br>Time-Frequency Masking,<br>Rectangle + Time-Frequency Masking |

**Table 3.14** continued

|          |                         |       |    |   |     |                                                                                                         |
|----------|-------------------------|-------|----|---|-----|---------------------------------------------------------------------------------------------------------|
| Basic TL | Quartznet<br>Malay 20×1 | Malay | 20 | 1 | 100 | Rectangle<br>Masking,<br>Time-<br>Frequency<br>Masking,<br>Rectangle +<br>Time-<br>Frequency<br>Masking |
|----------|-------------------------|-------|----|---|-----|---------------------------------------------------------------------------------------------------------|

For the experiment's time-frequency masking augmentation, we set its hyperparameter values to follow the settings that was implemented in Park et al. (2019) while the rectangle masking augmentation hyperparameter values were set to the default. The combined settings feature merged values of both masking techniques. The three settings were applied to the transfer learning conducted using both English and Malay model.

#### 3.7.5.5 TL with Optimized Betas

For the third TL setting, we further enhance our TL method by optimizing the beta ( $\beta_1$ ,  $\beta_2$ ) values of the learning curve of our model after applying spectrogram augmentation in the previous experiment. The beta values alongside the learning rate hyperparameter have been known to shape the learning curve of a model in a training, thus, we decided to investigate its impact and attempt to optimize it to obtain the best performance our model could achieve. For this setting, we introduced beta values optimizations into the previous model architecture and ran several rounds of experiment. Previously, all beta

hyperparameters were set to a value of (0.8, 0.5). We proceed to set up experiments with different beta values for each transfer learning operation in order to investigate the impact of this hyperparameter. We first experimented using the STTQznet English model with several combinations of beta values,  $\beta_1$  and  $\beta_2$ , following two protocols that we have established as our investigation guideline for this TL setting. The two protocols were labelled as “Min to Max to Min” and “Median”.

In the Min to Max to Min protocol, the value of  $\beta_1$  were first set to represent the lower boundary of the learning curve (0.1) while the value of  $\beta_2$  represent the value for the upper boundary (0.9), before reversing the role of  $\beta_1$  and  $\beta_2$  to be vice versa, thus the name Min to Max to Min.

The Median protocol on the other hand focuses on investigating a point in the curve to the median. For the first run, it sets the value for  $\beta_1$  to be the lower boundary (0.1) of the learning curve while taking the median value (0.5) to be  $\beta_2$  (the median of 0-1). The next several runs would be taking the median to another median (0.5, 0.5) and then the upper boundary value to the median (0.9, 0.5).

These were the two protocols that have been followed during the process of the beta values optimizations and the conduct of our third TL setting. From the experiments conducted, we observed which combinations of beta values  $\beta_1$  and  $\beta_2$  that optimizes the STTQznet model to achieve the lowest WER possible when carrying out transfer learning, and then implement it in the optimization of our Malay model. The full information regarding the structure of the YAML file for the model architecture used in all of the transfer learning experiments has been attached in APPENDIX I to N.

### 3.8 Experimental Steps with Figures

This section describes steps to run a model training experiment using Google Colaboratory and Nvidia NeMo. We utilized our own notebook “E2E-CNN-Framework-for-Iban-ASR.ipynb” to explain the steps involved in setting up an experiment. Figures are included in the description to demonstrate the execution process involved.

#### 3.8.1 Training a model from scratch

The experiment starts with the import of core dependencies. Simply run the code snippet for the import as shown in Figure 3.5.

```
[ ] ## Install dependencies
    !pip install wget
    !apt-get install sox libsndfile1 ffmpeg
    !pip install unicode
    !pip install matplotlib>=3.3.2
```

**Figure 3.5:** Code snippet for the import of dependencies found in our notebook

The process continues with the import of other necessary libraries and our dataset as shown in Figure 3.6. Then, restart the runtime session for the kernel to pick up any upgraded packages (non-local runtime only), this can be done via going to “Runtime > Restart session”.

▼ Necessary Imports

```
[ ] # Before the conduct of any experiment, we have to first import the necessary libraries.

# Install NeMo
BRANCH = 'main'
!python -m pip install git+https://github.com/NVIDIA/NeMo.git@$BRANCH#egg=nemo_toolkit[all]

# Set up the configuration folder and grab a YAML config file from the NeMo github repository.
# The imported config file holds the detail of a basic CNN model architecture.
!mkdir configs
!wget -P configs/ https://raw.githubusercontent.com/NVIDIA/NeMo/$BRANCH/examples/asr/conf/config.yaml

# Import our targeted language, the Iban dataset by Juan (2015) from Github.
# Take note that for preliminary experiment, we randomly took 2000 lines of sentences from the main corpus
# but with equal ratio of male and female speaker as our training set.
!git clone https://github.com/sarahjuan/iban.git
```

Remember to **restart the runtime** for the kernel to pick up any upgraded packages.

**Figure 3.6:** Code snippet for the import of necessary libraries used in our experiment

Next, we proceed to run the next code snippet which imports the Librosa Python package (for audio analysis) into our notebook and also specifies the directory path of our would-be-downloaded files as shown in Figure 3.7.

```
Remember to restart the runtime for the kernel to pick up any upgraded packages.

[ ] # Import librosa which is needed by the Nvidia NeMo for manifest file building, manifest file tells NeMo the location of our trans
import librosa

# This is where imported directories will be placed. Change this if you don't want the data to be extracted in the current directo
data_dir = '.'
```

**Figure 3.7:** Code snippet for the import of librosa and specification of directory

Moving on, we proceed to setup Nvidia Nemo for experiments, we begin by importing Nemo into our Google Colaboratory notebook as depicted in Figure 3.8.

```
[ ] # Non-local runtime users (colab users) can skip the above snippet to run this one immediately.
    # Import NeMo's "core" package
    import nemo

    # NeMo's ASR collection - this collections contains complete ASR models and building blocks (modules) for ASR development.
    import nemo.collections.asr as nemo_asr
```

**Figure 3.8:** Code snippet for the import of Nvidia Nemo

After Nemo is successfully imported, we can start to build our ASR model. To build a new ASR model, Nemo has to first build its manifest file which comprises details such as the directory path of audio training data and its corresponding transcript. The manifest file also tells the system about the information on each of the lines which includes its audio file path, duration, and text. To begin building the manifest files, you need to first have your ‘train\_text’ and ‘test\_text’ files whereby these two files consist of lines of sentences that comprises of the ID of the audio file you are using for both train and test set together with its transcription. After that, simply run the “STEP 1 – Build manifest file” code snippet while ensuring all path in the code snippet is correctly specified according to the directory of your file (your audio and transcript files for both train and test set directory). Figure 3.9 depicts the above step.

▼ (STEP 1) Build manifest file.

```
[ ] # --- Building Manifest Files --- #
import json
import os

# Function to build a manifest
def build_manifest(transcripts_path, manifest_path, wav_path):
    with open(transcripts_path, 'r') as fin:
        with open(manifest_path, 'w') as fout:
            for line in fin:
                # Lines look like this:
                # ibf_001_001 pukul sepuluh malam

                file_id = line[: line.find(' ')] # to get all text only until before " pukul..."
                file_id = file_id.strip() # to get only "ibf_001_001"

                transcript = line[line.find(' ')+1: ].lower() # lower all alphabet after "ibf_001_001"
                transcript = transcript.strip() # strip the transcript
                audio_path = os.path.join(
                    data_dir, wav_path,
                    file_id[: file_id.rfind('_')], # remember to get the folder too, it is taking all
```

**Figure 3.9:** Code snippet for building manifest file (STEP 1)

After successfully building our manifest file, we proceed to establish the YAML configuration of our model design. Simply run the “STEP 2” code snippet to load our chosen YAML configuration file as shown in Figure 3.10.

▼ (STEP 2) Establish YAML configuration for model design.

```
[ ] # For preliminary experiment, we used 'config.yaml' imported from the NeMo repository as it holds basic model de
# For building a new model based on our preferred design, the details in the YAML file should be change to follow

# --- Config Information ---#
try:
    from ruamel.yaml import YAML
except ModuleNotFoundError:
    from ruamel_yaml import YAML
config_path = './configs/config.yaml' # directory of config should be placed here.

yaml = YAML(typ='safe')
with open(config_path) as f:
    params = yaml.load(f)
print(params)
```

**Figure 3.10:** Code snippet to establish YAML configuration (STEP 2)

Next, after we have our YAML configuration loaded, we proceed to setup our training modules for model training. Simply run all the code snippets that can be found under the label “STEP 3” to setup the modules as shown in Figure 3.11 - Figure 3.12.

```
[ ] # Import Torch and PTL for model training

import torch
import pytorch_lightning as ptl

if torch.cuda.is_available():
    accelerator = 'gpu'
else:
    accelerator = 'cpu'

# Don't forget to specify the number of epoch first
# For our preliminary experiment I (Experiment I in the thesis) - Using the 2000
# For our preliminary experiment II (Experiment II in the thesis) - Using the 20
# 500 - 5500 epoch with an interval of 500 epochs in each increase.
EPOCHS = 1

trainer = ptl.Trainer(devices=1,
                      accelerator=accelerator,
                      max_epochs=EPOCHS,
                      accumulate_grad_batches=1,
                      enable_checkpointing=False,
                      logger=False,
                      log_every_n_steps=5,
                      check_val_every_n_epoch=10)

[ ] # Use OmegaConf to instantiate our model, we tell the system where is our manifest

from omegaconf import DictConfig
params['model']['train_ds']['manifest_filepath'] = train_manifest
params['model']['validation_ds']['manifest_filepath'] = test_manifest
```

**Figure 3.11:** Code snippets to set up modules and manifest files

```
[ ] # Create a new model (SKIP this step if you are using a pre-trained model)
my_model = nemo_asr.models.EncDecCTCModel(cfg=DictConfig(params['model_cfg']))
# note that 'my_model' is the name of our model here, you can change it

[ ] # Setup model with the trainer (change the name 'my_model' accordingly)
my_model.set_trainer(trainer)

# Update the model's internal config (change the name 'my_model' accordingly)
my_model.cfg = my_model._cfg

[ ] # Use OmegaConf to read our created model configuration (change the name accordingly)
from omegaconf import OmegaConf, open_dict
import copy

cfg = copy.deepcopy(my_model.cfg)
```

**Figure 3.12:** Code snippets to create a new model and setup its configurations.

Next, to monitor our training logs, we need to setup our experiment manager. Simply run snippet “STEP 4” to setup the manager as shown in Figure 3.13.

```
▼ (STEP 4) Setup the experiment manager to monitor the logs of our training.

[ ] from nemo.utils import exp_manager

# Environment variable generally used for multi-node multi-gpu training.
# In notebook environments, this flag is unnecessary and can cause logs of multiple threads
os.environ.pop('NEMO_EXPM_VERSION', None)

# Specify the directory of your log in 'exp_dir'
# Specify the name of your experiment log in 'name'
config = exp_manager.ExperimentManagerConfig(
    exp_dir=f'experiments/lang-lban/preliminary/E001',
    name=f'E001-prelim',
    checkpoint_callback_params=exp_manager.CallbackParams(
        monitor="val_wer",
        mode="min",
        always_save_nemo=True,
        save_best_model=True,
    ),
)

config = OmegaConf.structured(config)

logdir = exp_manager.exp_manager(trainer, config)
```

**Figure 3.13:** Code snippet to setup the experiment manager

Once the manager has been setup, we can now proceed to model training. To begin model training, simply run the code snippet of “STEP 5 – Start model training”. Figure 3.14 depicts the process of the steps mentioned.

▼ (STEP 5) Start the model training.

```
[ ] # Start the training
    %%time
    trainer.fit(my_model) # (change the name 'my_model' according to the currently loaded model)
```

Once your model has completed its training, you can proceed to save it. We save two files from the completed training, the model checkpoint and its weight.

**Figure 3.14:** Code snippet to begin model training

Once a model training has been completed, save the model trained together with its checkpoint via code snippet “STEP 6” as shown in Figure 3.15.

▼ (STEP 6) Save model.

```
[ ] # Save NeMo model (Model Weight)
    my_model.save_to("model/E001-prelim.nemo") # (change the name 'my_model'

[ ] # Save trained checkpoint using torch (Model Checkpoint)
    torch.save(my_model.state_dict(), 'model/E001-prelim.pth') # (change the
```

**Figure 3.15:** Code snippet for model and checkpoint saving.

After a model has been saved, we can easily load it again via the function “restore\_from” and “torch.load” as shown in Figure 3.16. Simply run the code snippets shown in the figure to load the model and its checkpoint for fine-tuning.

### ▼ Load Saved Model

This section is for loading a previously trained model. As described in STEP 6, a trained model and its checkpoint and checkpoint file. The code snippet below shows the function to load the saved models. Make sure to run b to conduct fine-tuning on your saved model.

```
[ ] # Load previously trained nemo model. (Load weights)
    my_model = nemo_asr.models.EncDecCTCModel.restore_from('model/E001-prelim.nemo')

[ ] # Establish the trainer of the newly loaded model.
    my_model = nemo_asr.models.EncDecCTCModel(cfg=DictConfig(params['model']), trainer=trainer)

[ ] # Load the model's checkpoint
    state_dict = torch.load('model/E001-prelim.pth')
    my_model.load_state_dict(state_dict)
```

**Figure 3.16:** Code snippet to load model and checkpoint

## 3.8.2 Carrying out Transfer Learning

For transfer learning operation, our first step is the import of a pretrained model into our notebook to be used as our source language. In our notebook, we display two options for model import to carry out transfer learning on the Iban dataset, English model, and Malay model. You can choose to load either one of the models, simply run “STEP 9” snippet found in the notebook to import the model as depicted in Figure 3.17.

#### ✓ (STEP 9) Importing Pre-trained model

To begin Technique #3, we import the pre-trained model that we would like to first use for Transfer Learning. Note the general process of carrying out Transfer learning. Please remember to load the **correct pre-trained model/** to avoid output error.

```
[ ] # Import ENGLISH Pre-trained model as our Transfer learning model named 'tl_model'
    tl_model = nemo_asr.models.EncDecCTCModel.from_pretrained(model_name="stt_en_quartznet15x5")

[ ] # Import MALAY Pre-trained model as our Transfer learning model named 'tl_model'
    tl_model = nemo_asr.models.EncDecCTCModel.restore_from('model/E010-B20R1-BM-rectmasked.nemo')

[ ] # Check what kind of vocabulary/alphabet the TL model has right now
    print(tl_model.decoder.vocabulary)
```

**Figure 3.17:** Code snippets for loading a pretrained model

After the pretrained model has been imported, the next step is to setup its data loaders. Simply run these snippets under the label “STEP 10” as shown in Figure 3.18 - Figure 3.19 to set up the data loaders.

## ✓ (STEP 10) Setting up data loaders

We conduct transfer learning one at a time, thus, we have to specify the data loader

```
[ ] # Use OmegaConf to read our TL model configuration
    from omegaconf import OmegaConf, open_dict
    import copy

    cfg = copy.deepcopy(tl_model.cfg)

[ ] # Setup train, validation configs
    from omegaconf import DictConfig
    # Set train config
    params['model']['train_ds']['manifest_filepath'] = train_manifest
    params['model']['train_ds']['labels'] = [
        ' ', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
        'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'
    ]
    params['model']['train_ds']['normalize_transcripts'] = False
    params['model']['train_ds']['batch_size'] = 5
    params['model']['train_ds']['num_workers'] = 8
    params['model']['train_ds']['pin_memory'] = True
    params['model']['train_ds']['trim_silence'] = True
    # Set validation config
    params['model']['validation_ds']['manifest_filepath'] = test_manifest
    params['model']['validation_ds']['labels'] = [
        ' ', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l'
```

**Figure 3.18:** Code snippets to setup the transfer learning data loaders

```
[ ] # Setup data loaders with new configs

    tl_model.setup_training_data(train_data_config=params['model']['train_ds'])
    tl_model.setup_multiple_validation_data(val_data_config=params['model']['validation_ds'])

[ ] # Display the current training configuration of our TL model
    print(OmegaConf.to_yaml(tl_model.cfg.train_ds))

[ ] # Display the current encoder configuration of our TL model
    print(OmegaConf.to_yaml(tl_model.cfg.encoder))
```

**Figure 3.19:** Code snippets to setup the data loaders (continued) and to display the model configuration.

Furthermore, we also have to set up our optimizer as it influences our transfer learning operation. Similar to the previous step, simply run “STEP 11” snippet in the notebook to set up the optimizers. Ensure the values of the optimizer has been set according to your requirements beforehand before running the cell. Our default values for this step are depicted in Figure 3.20.

▼ (STEP 11) Setting up optimizer

The optimizer play an important role in enhancing the quality of our transfer learning operation. When we conduct transfer learning, we can configure several parameters in the model optimizers to get the best performance obtainable from model training. The snippet below displays the default value of each of the optimizers parameter. For the first TL setting (Basic TL), simply run the code snippet below.

```
[ ] # Display the current optimizer configuration of our TL model
print(OmegaConf.to_yaml(tl_model.cfg.optim))

[ ] # Setting the parameter values of our optimizer
with open_dict(tl_model.cfg.optim):
    tl_model.cfg.optim.name = 'novograd'
    tl_model.cfg.optim.lr = 0.01 # default value is 0.01
    tl_model.cfg.optim.betas = [0.8, 0.5] # beta values set here ([0.9, 0.1] is the best value from setting 3)
    tl_model.cfg.optim.weight_decay = 0.001 # Original weight decay = 0.001
    tl_model.cfg.optim.sched.warmup_steps = None # Remove default number of steps of warmup
    tl_model.cfg.optim.sched.warmup_ratio = 0.05 # 5 % warmup
    tl_model.cfg.optim.sched.min_lr = 1e-5
```

**Figure 3.20:** Code snippet to setup the optimizer and its default values

The next few steps after this follow the action taken during the preparation of model training from scratch. We set up the experiment manager, run the model training, and once the training has finished, we save the model. Our notebook also provides cells with codes for implementing the TL settings that we have used in this research’ experiment. Simply run the cells as instructed in the comment of the snippets to implement the setting into your transfer learning operation. Thus, concludes the steps for carrying out transfer learning operation.

### 3.8.3 Evaluate model accuracy

This section describes the steps taken in evaluating the accuracy of our model using the Speech Recognition Scoring toolkit (SCTK). Once a model has been saved, we proceed to use it to transcribe our test set to obtain our hypothesis file. Our notebook has already

prepared a code snippet for transcribing the test set which is labelled as “STEP 7 – Transcribe test set using model”. To transcribe, simply run the code snippet as shown in Figure 3.21.

```
[ ] # Transcribe all audio files (test set) using the model

def transcribe_all(transcripts_path, output_path, wav_path):
    with open(transcripts_path, 'r') as fin:
        with open(output_path, 'w') as fout:
            for line in fin:
                # line look like this: ibf_001_001 pukul sepuluh malam

                file_id = line[: line.find(' ')] # to get all text just until before " pukul..."
                file_id = file_id.strip() # to get only "ibf_001_001"

                # merging the data from the text files in order to obtain the audio path for transcription
                audio_path = os.path.join(
                    wav_path,
                    file_id[: file_id.rfind('_')], #remember to get the folder too, reverse searching for the last occurrence of "_"
                    file_id + '.wav')

                # get the transcription as text, to be written as output in the metadata fields, to be edited in notepad/notepad++
                transcription = str(my_model.transcribe(paths2audio_files= [audio_path], batch_size=5))
                # (change the name 'my_model' according to the currently loaded model)

                metadata = {

                    transcription : "("+file_id+")" # all non alphabetic character should be remove manually in notepad/notepad++
                }
                json.dump(metadata, fout)
                fout.write('\n')

test_transcripts = data_dir + '/iban/data/test/test_text'
trans_output = data_dir + '/model/model_output.txt' # the output transcript is located in this directory
if not os.path.isfile(trans_output):
    transcribe_all(test_transcripts, trans_output, 'iban/data/wav/')
    print("transcription complete.")
```

**Figure 3.21:** Code snippet to transcribe test set using current model

Once the output transcript (hypothesis file) has been completed and produced, we evaluate the transcript by comparing it with our reference file using SCTK. Thus, the format of both of our output file (hypothesis) and our reference file (reference) has to follow the format required by SCTK. An example of the format is included in the notebook. To run the evaluation using SCTK, simply run all the code cells labelled under “STEP 8” in the notebook and follow the instructions specified in the comments as shown in Figure 3.22 - Figure 3.23.

```
[ ] # Import SCTK library from Github
# If you have already imported SCTK previously, you may skip this snippet
!git clone https://github.com/usnistgov/SCTK.git
```

**Figure 3.22:** Code snippet to import SCTK library

Start SCTK operation in the cell below

```
[ ] # This snippet can only be run in colab settings.
# For local runtime users, consider running STEP 8 in a colab runtime setting by copying the codes into a new cloud server notebook.
!bash
```

Input for bash terminal (You can copy the code below line-by-line into the terminal):

---

```
cd SCTK
make config all check install doc
cd bin
```

---

After successfully running the above snippet, run this code to start scoring:

---

```
./sclite -F -i swb -r /content/reference.txt trn -h /content/hypothesis.txt trn -o dtl all
```

Remember to rename your output file as **"hypothesis.txt"** and your reference file as **"reference.txt"** unless you decide to change the code syntax above.

---

**Figure 3.23:** Code snippet to run SCTK and its instructions

This step concludes our method of model evaluation using the SCTK.

### 3.9 YAML File

This section describes the purpose of all YAML files that was loaded into each of our experiment. Full configuration details of each file are available in APPENDICES B – N in the appendices section.

| Appendix | Purpose                                                                                         |
|----------|-------------------------------------------------------------------------------------------------|
| B        | YAML file of the model architecture specified during the conduct of the preliminary experiment. |

|   |                                                                                                                                                                                           |
|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| C | YAML file of the model architecture specified during the conduct of the Block optimization experiment.                                                                                    |
| D | YAML file of the model architecture specified during the conduct of the Sub-block optimization experiment.                                                                                |
| E | YAML file of the model architecture specified for the Quartznet 1×1 model during the conduct of the Epoch optimization experiment.                                                        |
| F | YAML file of the model architecture specified for the Quartznet 20×1 model during the conduct of the Epoch optimization experiment.                                                       |
| G | YAML file of the model architecture specified during the conduct of the Rectangle Masking Spectrogram Augmentation experiment.                                                            |
| H | YAML file of the model architecture specified during the conduct of the Time-Frequency Masking Spectrogram Augmentation experiment.                                                       |
| I | YAML file of the model architecture specified for the Quartznet 15×5 English model during the conduct of the Basic TL setting experiment.                                                 |
| J | YAML file of the model architecture specified for the Quartznet 15×5 English model during the conduct of the TL with TF-SA (Best Model – with Time-Frequency SA) setting experiment.      |
| K | YAML file of the model architecture specified for the Quartznet 15×5 English model during the conduct of the TL with Beta Optimization (Best Model – with (0.9, 0.1)) setting experiment. |

|   |                                                                                                                                                                                         |
|---|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| L | YAML file of the model architecture specified for the Quartznet 20×1 Malay model during the conduct of the Basic TL setting experiment.                                                 |
| M | YAML file of the model architecture specified for the Quartznet 20×1 Malay model during the conduct of the TL with SA (Best Model – with Time-Frequency SA) setting experiment.         |
| N | YAML file of the model architecture specified for the Quartznet 20×1 Malay model during the conduct of the TL with Beta Optimization (Best Model – with (0.9, 0.5)) setting experiment. |

### 3.10 Chapter Summary

In this chapter, we have presented our proposed framework methodology. We have described our framework to develop a CNN Iban ASR and investigated potential techniques that may help under-resourced ASR model to improve without the need of an increase in speech data. We have also explained the details of our dataset and the steps conducted for its preparation as well as the tool that we are using in our research. Furthermore, we have conducted preliminary experiment and investigated the effectiveness of three proposed techniques which includes hyperparameter optimization, spectrogram augmentation, and transfer learning. We studied the impacts of three different settings in carrying out transfer learning and listed out its protocol of execution. Finally, we included a list that displays the appendixes of each YAML file that we have used in this research experiment at the end of the chapter.

## **CHAPTER 4**

### **RESULTS, ANALYSIS AND DISCUSSION**

#### **4.1 Introduction**

This chapter presents the data and results that we have obtained from our research's experiment and describes the dynamics of our Iban ASR model performances when we optimized specific hyperparameters in the CNN architecture to certain values in an attempt to improve its transcribing accuracy. It includes the results from our preliminary experiment, the hyperparameter optimization experiment, the spectrogram augmentation application experiment, and the transfer learning method experiment. We also include the analysis of our results in this chapter as well as performance comparison of our CNN Iban ASR model with other previously developed Iban ASR model. All discussions regarding the strengths and weaknesses of the CNN model are also presented in this chapter.

#### **4.2 Preliminary Experiment Result**

This section discusses the results that we have obtained from our preliminary experiment, which was done to investigate the CNN architecture for the first time and to obtain a baseline model performance while focusing on investigating two objectives: to investigate the dependency of CNN model towards the amount of resource needed for model training (Experiment I) and to perform sensitivity analysis on the CNN parameters (Experiment II).

##### **4.2.1 Experiment I Results**

As mentioned in CHAPTER 3, Experiment I investigate the effect of the amount of resource towards the performance of a CNN ASR model. Figure 4.1 illustrates the results

obtained when we increase the number of sentences in the train set experiment while Table 4.1 shows the results obtained when we increase the number of sentences in the test set.

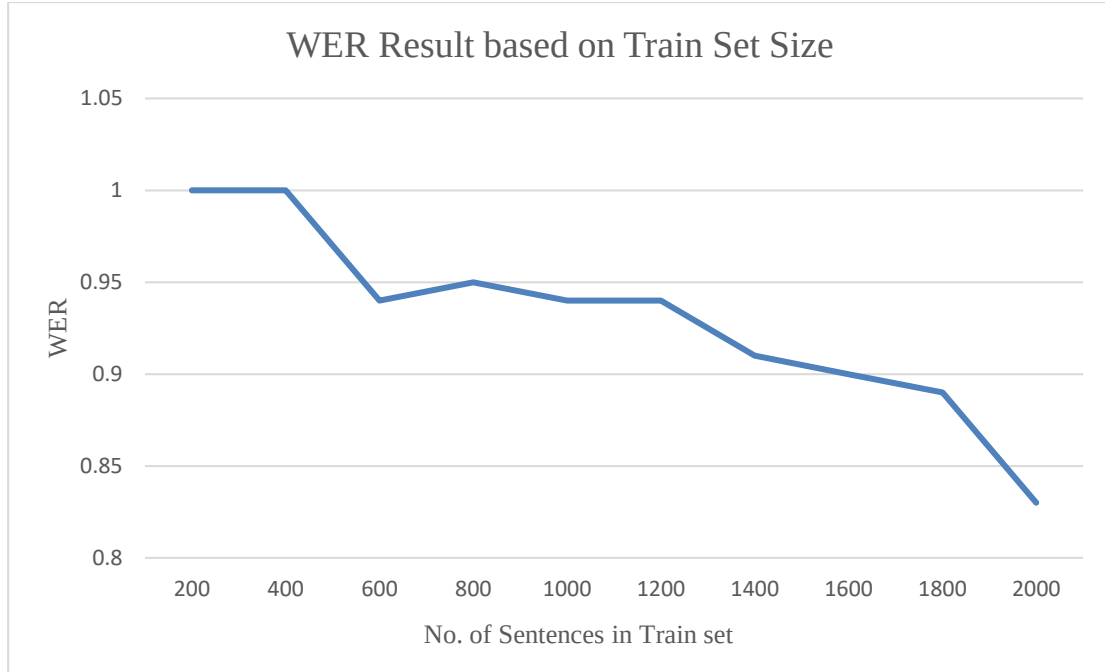


**Figure 4.1:** Results of Experiment I for train set increase

**Table 4.1:** Results of Experiment I for test set increase

| Test set size (sentence) | WER         |
|--------------------------|-------------|
| 100                      | 0.88        |
| 200                      | <b>0.87</b> |
| 300                      | 0.91        |
| 400                      | <b>0.87</b> |

Based on Figure 4.1, we found that the CNN performance, similar to other acoustic modelling technique (Juan, 2015; Santana, Santos, Matos, & Macedo, 2018), improves as

the amount of resource (size of train set) increases. However, the difference between CNN ASR model to other acoustic modelling technique is that even without the integration of core component such as the language model and pronunciation dictionary, it was still able to experience an increase in performance. Using its very basic architecture, we were able to achieve a WER of 0.83 at the final stage of increment with 2K lines of sentences in the train set, which is a reduction of  $\sim 20\%$  error rate from the initial 100%.

As the number of lines of sentence in the test set was increased up to 400, Table 4.1 shows that our model produces somewhat similar result as obtained in the train set experiment, with WER of 0.87 being the lowest WER. This brings us to the conclusion that the model performance was not affected by the amount of testing lines given to transcribe. Although being given a bigger scope of task for transcription, the model performance stays consistent with the amount of training that it has previously received.

#### **4.2.2 Experiment II results**

The experiment II of our preliminary experiment focuses on performing sensitivity analysis on a hyperparameter of our model architecture. This experiment studies their effects and impact towards the WER produced by our model as we change their values by increasing it over several increments. The experiment conducted in this stage focuses only on the epoch hyperparameter as it is still a preliminary experiment. Figure 4.2 shows the results of the experiment.

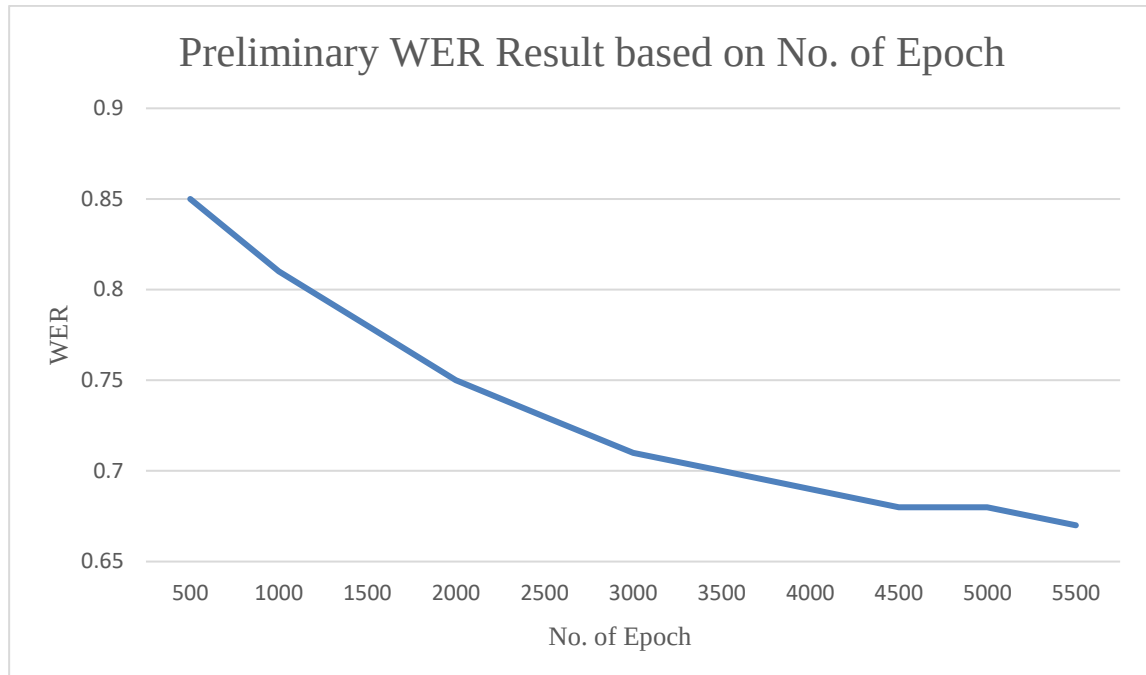


**Figure 4.2:** Line graph of the preliminary experiment II results

Based on Figure 4.2, as we increase the number of epochs, the model experiences an increase in performance. The lowest WER achieved from this experiment is at the final increment of epoch 5500 with a WER of 0.67. From epoch 500 to 5500, the model has experienced a total decrease of 0.18 WER. The increase in the number of epochs helps to improve the performance of our model, however, it takes a very huge increase between increment in the epoch in order to get a significant improvement in the model's performance. Thus, from this preliminary experiment, we observed that a hyperparameter such as the epoch provided positive impacts on the CNN performance. This outcome suggests that there is potential to further improve the performance of our CNN model by optimizing the values of hyperparameters.

## Comparison of Transcription

A qualitative observation to compare the hypotheses produced by our model in the preliminary experiment and the original transcript was conducted. Table 4.2 shows the output differences between the two transcripts.

**Table 4.2:** Comparison of output with the original transcript at epoch 5500

| Epoch    | Transcription                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | WER  |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| Original | <ol style="list-style-type: none"><li>1. pukul sepuluh malam (ibf_001_001)</li><li>2. selamat malam diatu kami naburka berita ari radio malaysia sarawak waifm (ibf_001_002)</li><li>3. menteri besai mutuska penemu deka begulai enggau orang di sarawak ngerami gawai saritu bejalai sereta berandau enggau mensia mayuh madahka iya menteri besai ke semua rayat lalu perintah endangibuhka semua rayat malaysia (ibf_001_003)</li><li>4. datuk seri najib tun razak madahka iya patut ngereja utai nitihka peneka iya ti nyadi menteri besai ke semua orang lalu madahka nya ukai semina jaku politik enggau jaku kelangkung iya aja (ibf_001_004)</li></ol>                                                                        | -    |
| 5500     | <ol style="list-style-type: none"><li>1. eku se pelamula (ibf_001_001)</li><li>2. a malanm litu <b>kami</b> nabur ka berit <b>ari radio</b> misia <b>sarawak</b> waim (ibf_001_002)</li><li>3. <b>menteri</b> besi motuska penembu <b>deka</b> peulai <b>enggau orang di sarawak</b> ngeramiegawai sarito pejalai seka berendal amensia <b>mayuh madahka iya menteri</b> besaika sema rakyat lalau perintah dang ibohka <b>semua</b> raya <b>malaysia</b> (ibf_001_003)</li><li>4. <b>datuk seri najib</b> tugrraza madahk i patul nerea tai <b>nitihka peneka iya</b> tinya di <b>menteri</b> besaika <b>semua orang lalu</b> madahki nya ksmina <b>jaku</b> politi <b>enggau jaku</b> plan kunndia <b>aja</b> (ibf_001_004)</li></ol> | 0.67 |

The hypothesis produced by our model in Table 4.2 holds a large number of errors in terms of words predicted which explained the high WER produced. However, the model was able to predict certain words perfectly such as “*menteri*”, “*kami*”, and “*Sarawak*”. Note

that this transcription is without the integration of pronunciation dictionary and language model.

As a conclusion, from the preliminary experiment we were able to complete our first ASR model development using the Nvidia Nemo and the toolkit was able to provide us with consistent output. From the experiment, we observed that the CNN acoustic model also increases its performance as the training data increases, which is similar to other types of acoustic model technique. In other words, the implementation of CNN acoustic model in an ASR model does not limit it from the need of extra resources to further improve its performance. However, we were able to further increase the performance of our model via increasing the value of the hyperparameter, epoch, and our model was still able to predict certain words from the audio input correctly and perfectly even without the help of pronunciation dictionary and language model. This shows that through the optimization of hyperparameters in the CNN, transcribing accuracy of the model can be increased. Finally, from the preliminary experiment, we were able to record our baseline performance. We chose 0.67 as our baseline WER which was achieved by the model trained with 2000 training lines for an epoch of 5500.

All procedures and results conducted in the preliminary experiment was written in a paper and has been successfully published in a journal. The paper was published in the Polish Journal of Telecommunication and Information Technology (JTIT) in the year 2022 (Michael, Juan, & Mit, 2022).

#### **4.3 Hyperparameter Optimization of Block, Sub-block, and Epoch.**

This section presents the result obtained from the experiment of the first proposed technique in the model improvement stage that has already been presented in section 3.7 of Chapter 3. This includes the result obtained from the no. of block, sub-block, and epoch

optimization experiment. Impact comparison towards the CNN model performance provided by each of the three hyperparameter will also be conducted in this section of chapter 4.

#### 4.3.1 Block

Section 3.7.1 of Chapter 3 presents the setup of our Block optimization experiment. Measuring the impact of the hyperparameter, Block, Figure 4.3 illustrates the dynamic performances of our CNN models when we increase the no. of blocks from a value of 1 to 20 while keeping the no. of sub-blocks to be consistent at 1 for an epoch of 100.

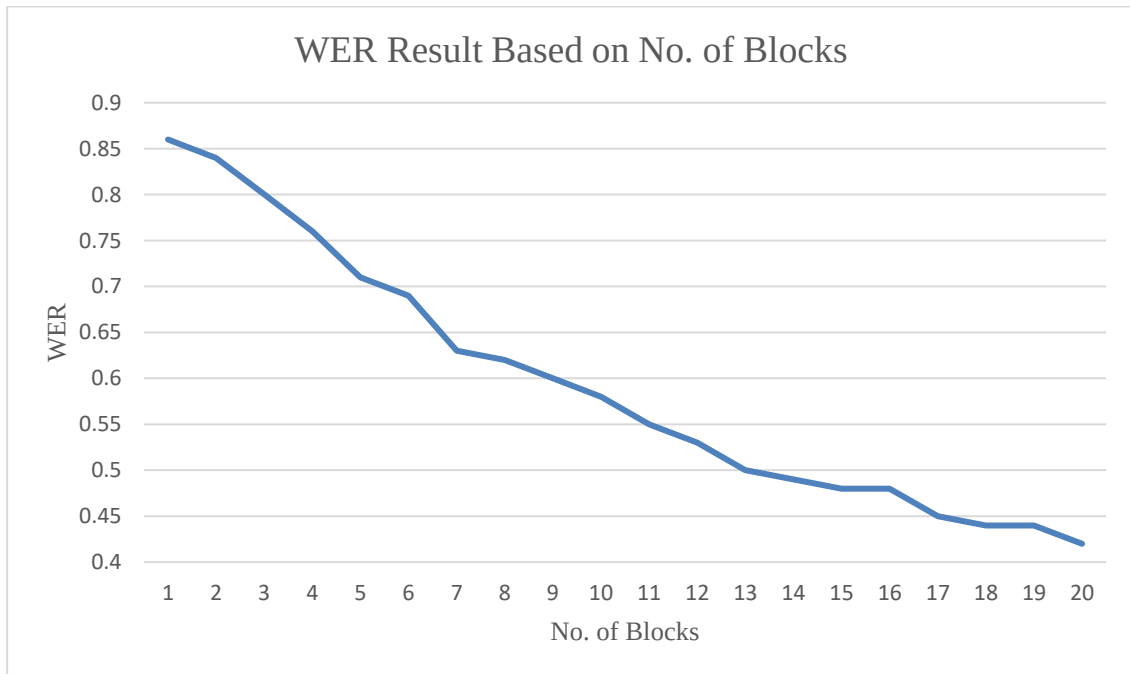


**Figure 4.3:** Dynamics of model performance in the Block experiment

As observed from Figure 4.3, increasing the number of blocks from 1 to 20 helps to lower the WER of our model, increasing its transcribing accuracy and model performance. The experiment result shows that the point where the WER slows down its rate of reduction was at block = 7, the reduction however continues as the number of block increases but at a slower rate. The lowest WER our model was able to achieve was at 20 blocks with a WER

of 0.42. Increasing the number of sub-blocks from 1 to 20 enhanced the performance of our model, reducing its WER from 0.86 to 0.42 WER, a 0.44 WER reduction.

### 4.3.2 Sub-block

Section 3.7.2 of Chapter 3 describes the setup of our Sub-block optimization experiment. Measuring the impact of the hyperparameter, Sub-block, Figure 4.4 illustrates the dynamic of our model performance when we increase the no. of sub-blocks from a value of 1 to 20 while keeping the no. of block to be equal to 1 for an epoch of 100.

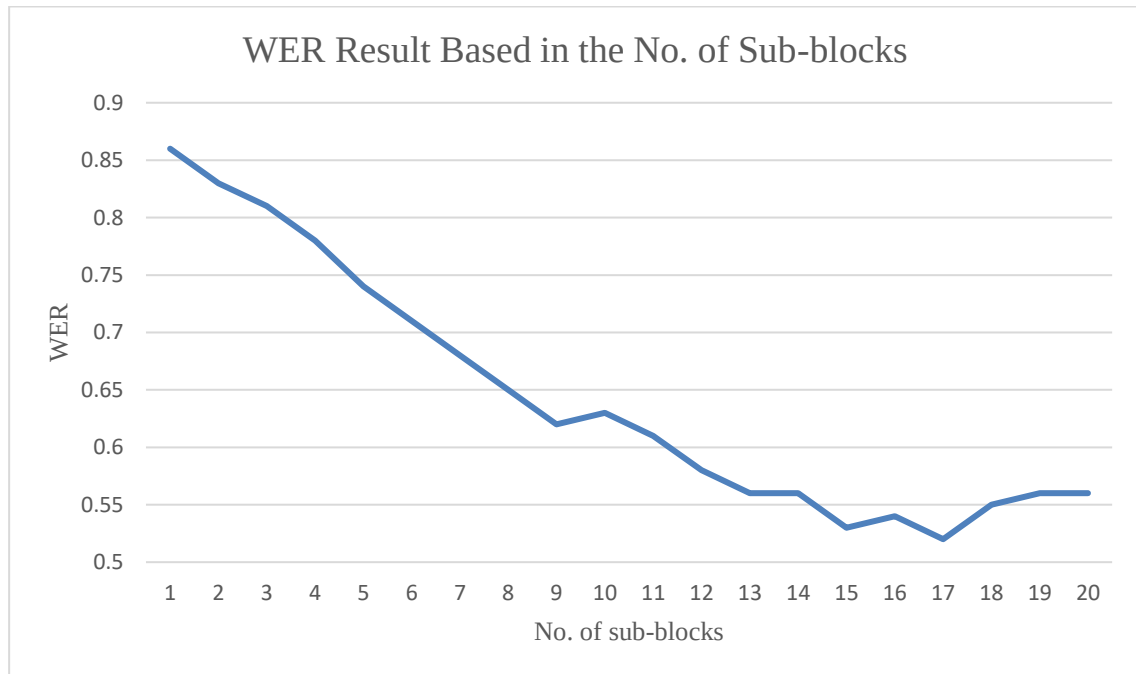


**Figure 4.4:** Iban CNN model performances in the Sub-block experiment

As observed from Figure 4.4, increasing the number of sub-blocks also helps to lower the WER of our model. Although, the dynamic of the WER decrease experienced by our model via the increase of the number of sub-blocks differs from the pattern of decrease obtained from the block experiment, in which the increase of sub-block also contributed a slight increase in WER when we increased it from 9 to 10 and 17 to 20 sub-blocks. The lowest WER recorded during this experiment was 0.52.

From the graph, it shows that with 20 stages of increase in sub-blocks, the reduction of WER stops at sub-block = 17, where the next increase of sub-blocks increases the WER instead. The rapid reduction of WER stops at sub-block = 9 before it continues to experience reduction but at a slower rate. There was also a minimal increase in WER at sub-block = 10, although, the WER continues to decrease at sub-block = 11. Increasing the number of sub-blocks from 1 to 20 enhanced the performance of our model, reducing its WER from 0.86 to 0.56 WER, a 0.30 WER reduction. However, considering that the WER increases instead when the number of sub-blocks exceeds 18, we conclude that our model was able to reduce its WER from 0.86 to 0.52, a 0.34 WER reduction. This is when we exclude the performance decrease experience after sub-block = 17.

### 4.3.3 Epoch

As for the Epoch optimization experiment, section 3.7.3 of Chapter 3 presents the setup details of our Epoch experiment. Two different models of specified configuration; a basic Quartznet  $1 \times 1$  and an optimized  $B \times R$  model configuration, Quartznet  $20 \times 1$ , obtained from the optimization of Block  $\times$  Sub-block experiment, has been used to run the experiment. Figure 4.5 shows the results obtained from the Quartznet  $1 \times 1$  model's experiment while Figure 4.6 illustrates the dynamic of our model performance when we increase the no. of epoch from a value of 100 to 1000 for the Quartznet  $20 \times 1$  model's experiment.

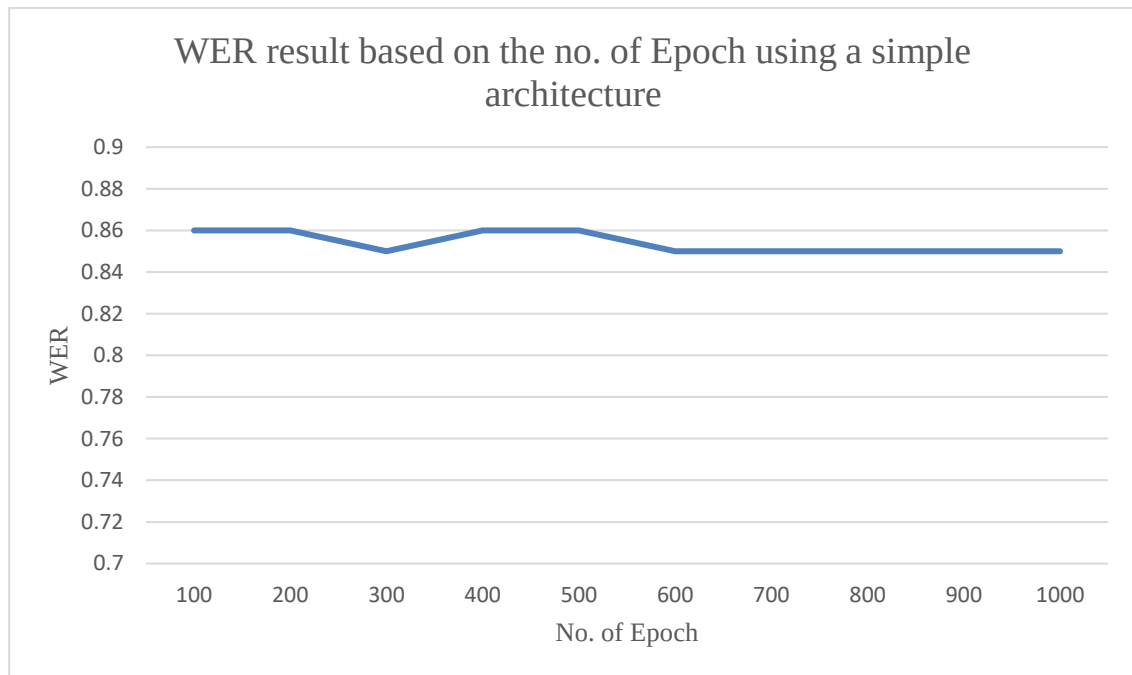


**Figure 4.5:** Graph result of the Epoch experiment for the Quartznet 1×1 model

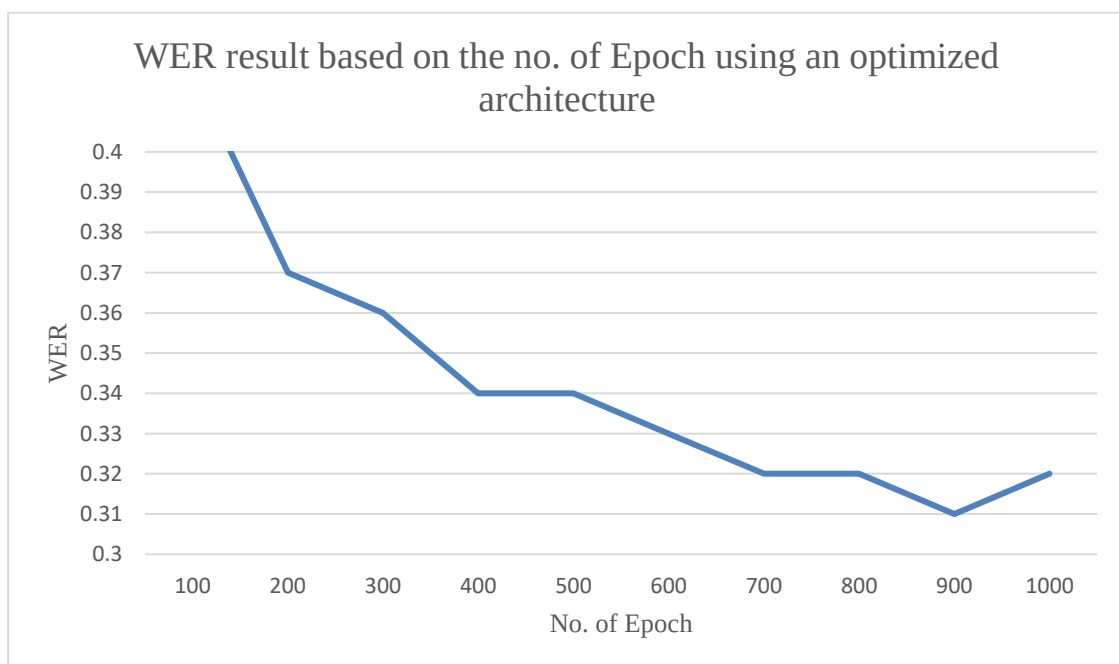


**Figure 4.6:** Graph result of the Epoch experiment for the Quartznet 20×1 model

Based on Figure 4.5, using the Quartznet 1×1 model architecture, it seems that increasing epoch only and without the optimizations of other parameters does not really help

in further improving the performance of our model. The model achieved a WER of around 0.86 to 0.85 and remained constant as the epoch was increased from 100 to 1000. As for our optimized model, the point of rapid reduction for the epoch hyperparameter stops at 200 before it continues to be reduced at a slower rate, but the model's lowest WER was achieved at epoch 900, before it increases again at epoch 1000. Using our optimized model, increasing the training epoch from 100 to 1000 helps to reduce its WER from 0.42 to 0.32, which is a decrease of 0.10 WER as observed in Figure 4.6. However, considering that the WER increases instead when the number of epochs exceeds 900, we conclude that our model was able to reduce its WER from 0.42 to 0.31, a 0.11 WER reduction. This is when we exclude the performance decrease experienced after epoch = 1000.

#### **4.3.4 Comparison between Epoch, Block, and Sub-block**

Based on the results obtained from the optimization of epoch, block, and sub-block experiments (refer to section 3.7.1-3.7.3), we ranked the rate of improvement of the three hyperparameters from the highest to the lowest in this manner: Block (0.44 WER) > Sub-block (0.34 WER) > Epoch (0.10 WER). The rank was made based on the reduction of WER provided by each hyperparameters after a total of 9 increments of value optimization was made. A single increment depicts a single increase in value to each hyperparameters, e.g: an increase of a single block, an increase of a single sub-block, and an increase of a hundred epochs to the model architecture or training. Table 4.3 tabulates the data of the WER reduction that was provided by the three hyperparameters for 9 increments.

**Table 4.3:** Rate of WER reduction produced after applying the different hyperparameter values (increment mode)

| Increment | Block (WER)  | Sub-block (WER) | Epoch (WER)  |
|-----------|--------------|-----------------|--------------|
| 0         | 0.00         | 0.00            | 0.00         |
| 1         | -0.02        | -0.03           | <b>-0.05</b> |
| 2         | -0.06        | -0.05           | -0.06        |
| 3         | -0.10        | -0.08           | -0.08        |
| 4         | <b>-0.15</b> | <b>-0.12</b>    | <b>-0.08</b> |
| 5         | -0.17        | -0.15           | -0.09        |
| 6         | -0.23        | -0.18           | -0.09        |
| 7         | -0.24        | -0.21           | -0.09        |
| 8         | -0.26        | -0.24           | -0.11        |
| 9         | <b>-0.28</b> | <b>-0.23</b>    | <b>-0.10</b> |

Based on Table 4.3, at the first increment, the hyperparameter epoch provided the highest WER reduction with  $-0.05$  WER. However, at the 4th increment, hyperparameter block provided the highest WER reduction with  $-0.15$  WER, followed by sub-block with  $-0.12$  WER, and epoch came at last with  $-0.08$  WER. This displays that the hyperparameter epoch provides its highest rate of improvement to the model only at the early stage of optimization, which also demonstrate to us that the later rounds of optimizations should be focused on other hyperparameters instead of just increasing the epoch. Figure 4.7 visualises the rate of improvement that was provided by the three hyperparameters for 9 increments in a graph.

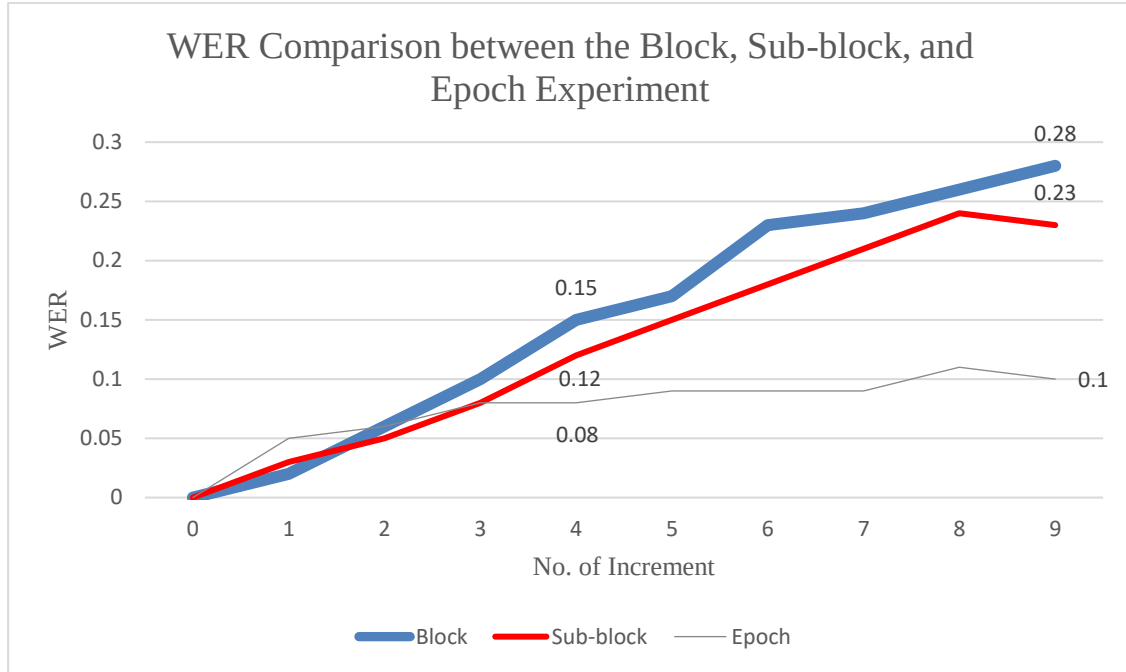


**Figure 4.7:** Graph of Rate of WER reduction between the three hyperparameter in 9 increments.

As observed from Figure 4.7, block has the highest rate of WER reduction while sub-block takes the second place, meanwhile, epoch placed last. It was also seen that at the 8th increment, both sub-block and epoch hyperparameter produce a slight increase in WER, before decreasing again at the 9th increment. The CNN block hyperparameter resulted the highest WER reduction at the 6th increment with  $-0.06$  WER, thus, making it the best hyperparameter to be controlled among the three in the case of optimizing a CNN ASR model. This does not mean that sub-block and epoch are unnecessary, but it is an indication that in the process of hyperparameters optimization of a CNN model training, the sub-block and epoch play less important role compared to the hyperparameter, block, in increasing the model performance.

Regardless of the rate of decrease, all three hyperparameters were able to improve our model performance significantly. Thus, we conclude that it is important to optimize the

three hyperparameters in an attempt to increase the performance of an under-resourced language CNN ASR model.

#### 4.3.5 Technique #1 Comparison of Reference and Hypothesis

We observed the hypothesis produced by the models from each hyperparameter experiment of the first proposed technique. Table 4.4 displays outputs of transcribed text, depicting their accuracies.

**Table 4.4:** Comparison of hypothesis produced by the models from the three hyperparameter experiments

| Hyperparameter<br>/Model                  | Transcription                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | WER<br>(Best) |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| None (Original Transcript)                | <ol style="list-style-type: none"> <li>1. pukul sepuluh malam (female_1)</li> <li>2. selamat malam diatu kami naburka berita ari radio malaysia sarawak waifm (female_2)</li> <li>3. menteri besai mutuska penemu deka begulai enggau orang di sarawak ngerami gawai saritu bejalai sereta berandau enggau mensia mayuh madahka iya menteri besai ke semua rayat lalu perintah endangibuhka semua rayat malaysia (female_3)</li> <li>4. datuk seri najib tun razak madahka iya patut ngereja utai nitihka peneka iya ti nyadi menteri besai ke semua orang lalu madahka nya ukai semina jaku politik enggau jaku kelangkung iya aja (female_4)</li> </ol>                                              | -             |
| Block - Quartznet<br>20×1 at 100<br>Epoch | <ol style="list-style-type: none"> <li>1. ku sepumluh <b>malam</b> (female_1)</li> <li>2. serah <b>malam</b> ditu <b>kami</b> nabur ka <b>berita ari radio</b> mlaysii <b>sarawak</b> waif (female_2)</li> <li>3. <b>menteri besai</b> mutus ka <b>penemu deka</b> begundai <b>enggau orang di sarawak</b> ngrami <b>gawai</b> sari tu <b>bejalai</b> seka <b>berandau enggau mensia mayuh madahka iya menteri besai ke semua</b> rakyat <b>lalu perintah</b> endang ngibuhka <b>semua rayat malaysia</b> (female_3)</li> <li>4. <b>datuk seri najib tun razak madahka iya patut</b> ngrejal ta <b>nitihka</b> meneka iya <b>ti nyadi menteri besai ke semua orang lalu madahka</b> nya uki</li> </ol> | 0.42          |

**Table 4.4** continued

|                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |      |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
|                                                   | smina <b>jaku politik enggau jaku</b> klankung ia <b>aja</b><br>(female_4)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |      |
| Sub-block –<br>Quartznet 1×17<br><br>at 100 Epoch | <ol style="list-style-type: none"> <li>1. kul <b>sepuluh malam</b> (female_1)</li> <li>2. skera alam dideka mi naburt ka <b>berita ari</b> adi mlysi <b>sarawak</b> waiff (female_2)</li> <li>3. <b>menteri besai</b> notieska penebu <b>deka</b> begunai <b>enggau orang di sarawak ngerami gawai saritu</b> bejadtai sreka <b>berandau</b> ngou <b>mensia</b> mayu badahka <b>iya menteri besai</b> ke semuarakyat <b>lalu perintah</b> dang ngibuhka <b>semua rayat malaysia</b> (female_3)</li> <li>4. <b>datuk seri najib tun razak madahka</b> a <b>patut</b> nereja kai nitihka bedneka yao ti unyadi <b>menteri besal ke semua orang</b> laliu <b>madahka nya</b> ukis <b>semina jaku politik enggau jaku</b> glangkon ia <b>aja</b> (female_4)</li> </ol> | 0.52 |
| Epoch –<br>Quartznet 20×1<br><br>at 900 Epoch     | <ol style="list-style-type: none"> <li>1. ku <b>sepuluh malam</b> (female_1)</li> <li>2. seauant <b>malam diatu kami naburka berita ari radio</b> mlasia <b>sarawak waifm</b> (female_2)</li> <li>3. <b>menteri besai mutuska</b> penembu <b>deka begulai enggau orang di sarawak ngerami gawai saritu</b> bejalai srka brandau <b>enggau mensia mayuh madahka iya menteri besai ke semua</b> rakyat <b>lalu perintah</b> dangibuhka <b>semua rayat malaysia</b> (female_3)</li> <li>4. <b>datuk seri najib tun razak madahka iya patut ngereja tai nitihka peneka iya ti nyadi menteri besai ke semua orang lalu madahka nya ukai semina jaku politik enggau jaku kelangkung dia aja</b> (female_4)</li> </ol>                                                    | 0.31 |

The outputs displayed the first four lines of the total number of sentences in the test set used as reference to evaluate the accuracy of the model. All three models were not able to perfectly predict the sentences correctly. However, they were able to form sentences closely similar to the structure of the sentences from the original transcript. At this stage, the model was able to predict more words from the Iban language correctly as compared to the model obtained from the preliminary experiment. This concludes the conduct of our first

proposed technique experiment, the hyperparameter optimizations of block, sub-block, and epoch.

#### 4.4 Spectrogram Augmentation

This section presents the result obtained during the implementation of the second technique proposed in the model improvement stage in our framework. It involves two augmentation methods: rectangle masking and time-frequency masking. The optimized Quartznet 20×1 model was used; we applied the techniques in two separate individual experiments and recorded its WER once the training has been completed. The training ran from an epoch of 100 to 900. The results for both augmentation methods is described below.

##### 4.4.1 Rectangle Masking

Figure 4.8 displays the performances of our Quartznet 20×1 models after a training of 100 to 900 epochs with rectangle masking.

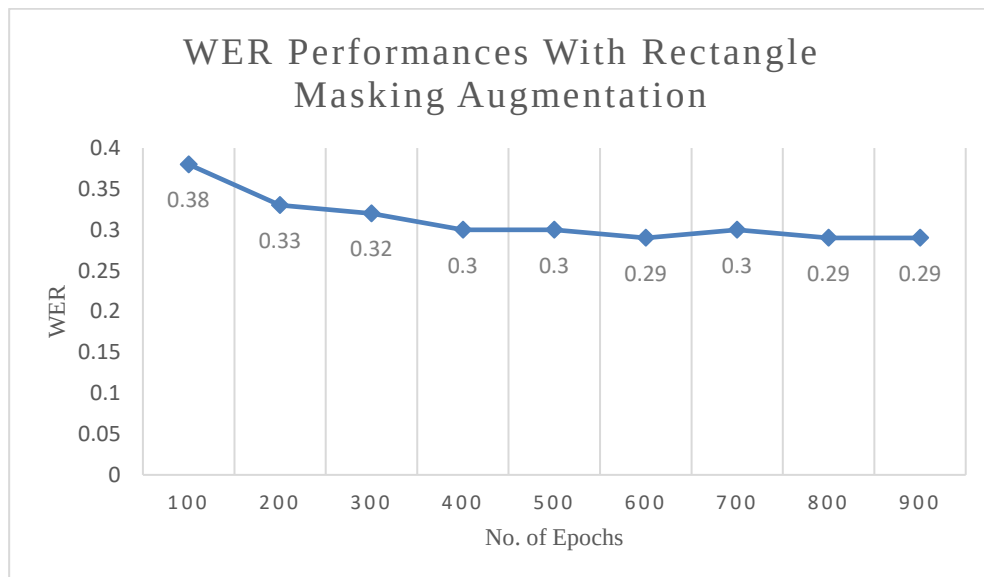


**Figure 4.8:** Graph of rectangle masking experiment results

Based on Figure 4.8, increasing the number of epochs helps to lower the WER of our models. However, the increase was only minimal before it reached to a steady WER values.

The lowest WER achieved during this experiment was 0.29. The augmentation helps to reduce the WER of our model in a range of 0.02 to 0.04 when we compared it with the data obtained from the epoch experiment using the optimized 20×1 model presented in section 4.1.4. At epoch 100, our model with rectangle masking applied scored a WER of 0.38 while the model without the augmentation trained from the epoch experiment scored a WER of 0.42 at epoch 100. Table 4.5 shows the WERs for models with and without applied rectangle masking.

**Table 4.5:** WER differences of model with rectangle masking and model without rectangle masking

| Epoch | Without Rectangle Masking | With Rectangle Masking | WER difference |
|-------|---------------------------|------------------------|----------------|
| 100   | 0.42                      | 0.38                   | 0.04           |
| 200   | 0.37                      | 0.33                   | 0.04           |
| 300   | 0.36                      | 0.32                   | 0.04           |
| 400   | 0.34                      | 0.30                   | 0.04           |
| 500   | 0.34                      | 0.30                   | 0.04           |
| 600   | 0.33                      | 0.29                   | 0.04           |
| 700   | 0.33                      | 0.30                   | 0.03           |
| 800   | 0.33                      | 0.29                   | 0.04           |
| 900   | 0.31                      | 0.29                   | 0.02           |

The rectangle masking augmentation consistently reduce a value of 0.04 WER to the initial performance of our model from epoch 100 to 600. After epoch 600, the reductions are

more varied, however it still results in our model having a lower WER as compared to the model without rectangle masking augmentation. In total, the rectangle masking spectrogram augmentation has helped our model to achieve an extra reduction of 0.33 WER with a training of 900 epochs.

#### 4.4.2 Time-Frequency Masking

Figure 4.9 displays the performances of our Quartznet 20×1 models after a training of 100 to 900 epochs with time-frequency (TF) masking.

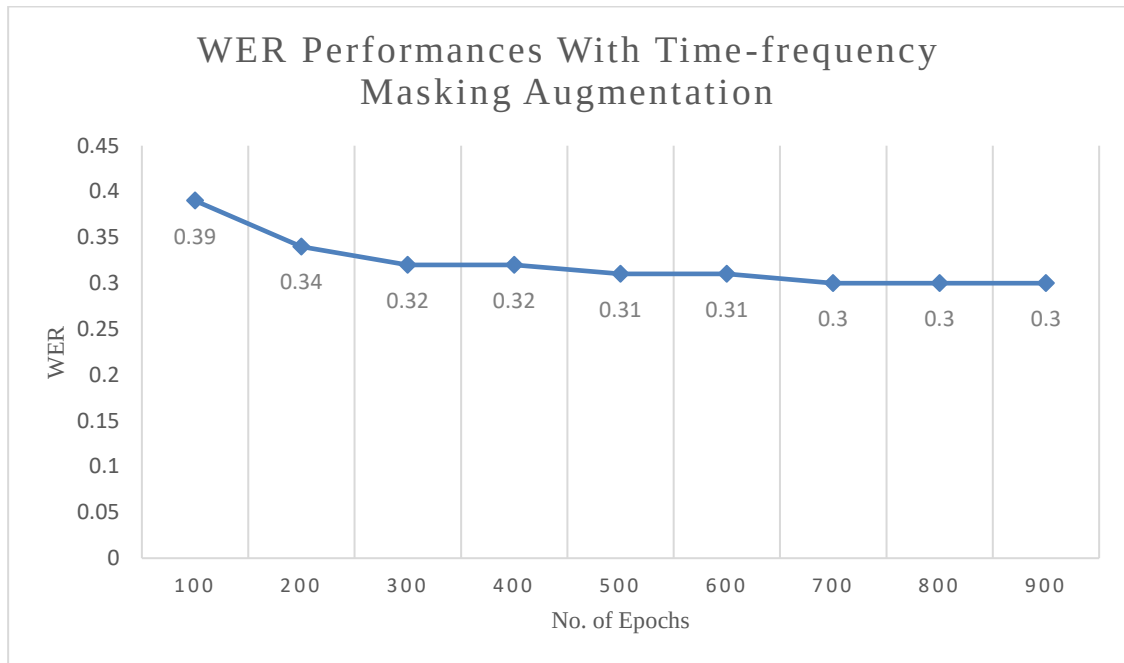


**Figure 4.9:** Graph result of time-frequency masking experiment

Based on Figure 4.9, the time-frequency masking augmentation technique was able to further reduce the WER of our model. The reduction of WER was from 0.39 to 0.30 with 0.30 as the lowest WER achieved for this experiment.

We apply the same method of comparison which uses the optimized 20×1 model from the epoch hyperparameter optimization’s experiment (refer to section 3.7.3) as our benchmark to be compared against the model we obtained during this experiment.

For 900 epochs, this augmentation method reduced our model's WER with values within a range of 0.01 to 0.04 WER per epoch increment. At epoch 300, it further reduced an additional 0.04 WER from the initial performance, achieving a WER of 0.32. Table 4.6 compares the WER obtained by our model after applying time-frequency masking against the optimized model from the epoch experiment.

**Table 4.6:** WERs for Iban CNN models with time-frequency masking and model without time-frequency masking

| Epoch | Without Time-Frequency Masking | With Time-Frequency Masking | WER differences |
|-------|--------------------------------|-----------------------------|-----------------|
| 100   | 0.42                           | 0.39                        | 0.03            |
| 200   | 0.37                           | 0.34                        | 0.03            |
| 300   | 0.36                           | 0.32                        | 0.04            |
| 400   | 0.34                           | 0.32                        | 0.02            |
| 500   | 0.34                           | 0.31                        | 0.03            |
| 600   | 0.33                           | 0.31                        | 0.02            |
| 700   | 0.33                           | 0.30                        | 0.03            |
| 800   | 0.33                           | 0.30                        | 0.03            |
| 900   | 0.31                           | 0.30                        | 0.01            |

The WER reduction produced by the time-frequency masking augmentation seems more varied as compared to the rectangle masking augmentation with values of 0.03 and 0.02 being the common values. However, this augmentation method was still able to improve the performance of our model by further reducing the WER initially obtained by our model.

Thus for 900 epochs, the time-frequency masking augmentation was able to reduce an extra 0.24 WER to the initial performance of our optimized model obtained from the epoch experiment.

#### 4.4.3 Comparison between Rectangle and Time Frequency Masking

Based on the data obtained from this experiment, the rectangle masking augmentation method was able to contribute more WER reduction as compared to the time-frequency masking augmentation, however, with a difference of only 0.01%. Overall, both augmentation methods were able to improve the performance of our model by 1% to 4%. This improvement may not seem significant but considering our case study of developing an ASR model for under-resourced languages, an ASR model with transcribing accuracy of 72% seems much better than an ASR model with 68% accuracy. This advantage is even critical especially when written resources of the language are lacking and/or unavailable to be used for model training and fine-tuning. The use of these two spectrogram augmentations has assisted us in making our data to be more robust for model training. Table 4.7 shows the summary of best WER results produced in the Technique 2 experiments.

**Table 4.7:** Summary of lowest WER for Technique 2

| Augmentations          | Lowest WER | Configurations |
|------------------------|------------|----------------|
| No augmentation        | 0.31       | B20R1 (E900)   |
| Rectangle masking      | 0.29       | B20R1 (E900)   |
| Time-frequency masking | 0.30       | B20R1 (E900)   |

#### 4.4.4 Technique #2 Comparison of Reference and Hypothesis

We also observed the hypothesis produced by the models from each spectrogram augmentation experiment of the second proposed technique. Table 4.8 displays their output of transcribed text, depicting their accuracies.

**Table 4.8:** Comparison of hypothesis produced by the models from the two spectrogram augmentation experiments

| Hyperparameter<br>/Model       | Transcription                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | WER<br>(Best) |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| None (Original Transcript)     | <ol style="list-style-type: none"> <li>1. pukul sepuluh malam (female_1)</li> <li>2. selamat malam diatu kami naburka berita ari radio malaysia sarawak waifm (female_2)</li> <li>3. menteri besai mutuska penemu deka begulai enggau orang di sarawak ngerami gawai saritu bejalai sereta berandau enggau mensia mayuh madahka iya menteri besai ke semua rayat lalu perintah endangibuhka semua rayat malaysia (female_3)</li> <li>4. datuk seri najib tun razak madahka iya patut ngereja utai nitihka peneka iya ti nyadi menteri besai ke semua orang lalu madahka nya ukai semina jaku politik enggau jaku kelangkung iya aja (female_4)</li> </ol>                                              | -             |
| Rect-Masking Model (900 Epoch) | <ol style="list-style-type: none"> <li>1. kul <b>sepuluh malam</b> (female_1)</li> <li>2. <b>selamat malam diatu kami</b> nabur ka <b>berita ari radio malaysia sarawak waifm</b> (female_2)</li> <li>3. <b>menteri besai mutuska penemu deka begulai enggau orang di sarawak ngerami gawai</b> sari tu <b>bejalai</b> saeka brandau <b>enggau mensia mayuh madahka iya menteri besai ke semua rakyat lalu perintah endangibuhka semua rayat malaysia</b> (female_3)</li> <li>4. <b>datuk seri najib tun razak madahka iya patut ngereja tai nitihka peneka iya ti nyadi menteri besai ke semua orang lalu madahka nya ukai semina jaku politik enggau jaku pelangkun ia aja</b> (female_4)</li> </ol> | 0.29          |

**Table 4.8** continued

|                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |      |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| TF-Masking<br><br>Model (900<br><br>Epoch) | <ol style="list-style-type: none"> <li>1. kul <b>sepuluh malam</b> (female_1)</li> <li>2. sellat <b>malam diatu kami</b> nabur ka <b>berita ari radio malaysia sarawak waifm</b> (female_2)</li> <li>3. <b>menteri besai mutuska penemu deka begulai enggau orang di sarawak ngerami gawai saritu bejalai</b> seka brandau <b>enggau mensia mayuh madahka iya menteri besai ke semua rakyat lalu perintah edangibuhka semua rayat malaysia</b> (female_3)</li> <li>4. <b>datuk seri najib tun razak madahka iya patut ngereja utai nitihka peneka iya ti nyadi Menteri besai ke semua orang lalu madahka nya ukai semina jaku politik enggau jaku</b> pelangkun dia <b>aja</b> (female_4)</li> </ol> | 0.30 |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|

The output displays the first four lines of the more than 450 total number of sentences in the test set used as reference to evaluate the accuracy of the model. Both models were able to transcribe the speech input almost perfectly with only several minor errors in each line of sentence. The model from the rectangle masking augmentation, however, was able to perfectly transcribe the output for the second line of sentence. Through the implementation of spectrogram augmentation, our model has gained an increase in its transcribing accuracy. This comparison concludes the conduct of our second proposed technique experiment.

#### 4.5 Cross-lingual Transfer Learning Method

This section presents the results obtained during the implementation of the third technique experiment, the transfer learning (TL) method. As mentioned in Chapter 3, the third technique conducts experiments on two pre-trained models (STTQznet and BM model) where each model conducts transfer learning under three settings, Basic TL, TL with Spectrogram Augmentation (SA), and TL with Optimized Betas. The results of the experiment are presented in the below sections.

#### 4.5.1 Basic TL

Table 4.9 shows the results of both models after carrying out a baseline transfer learning for 100 epochs.

**Table 4.9:** WER results obtained from the Basic TL experiment

| Model    | WER         |
|----------|-------------|
| STTQznet | <b>0.26</b> |
| BM       | 0.47        |

Based on the results, the baseline STTQznet model was able to give a much lower WER result as compared to our BM model. The STTQznet was able to achieve 0.26 WER while our BM model achieved a 0.47 WER. Due to the advantage of speech data abundance, the former model outperformed our BM model by 0.21 WER. The difference could be due to the size of the model as the STTQznet model was previously trained with comparatively more resources and longer hours as compared to our BM model. Comparing these results against the performance of our optimized 20×1 model, Table 4.10 shows its WER differences.

**Table 4.10:** WER differences between TL model against our optimized model

| Model      | WER         | WER difference against Qznet 20×1 |
|------------|-------------|-----------------------------------|
| STTQznet   | <b>0.26</b> | -0.05                             |
| BM         | 0.47        | +0.16                             |
| Qznet 20×1 | 0.31        | 0                                 |

As observed in Table 4.10, only STTQznet model was able to outperform the performance of our optimized model where it contributes a further decrease of 0.05 WER. The results also showed that a model with transfer learning applied outperforms the performance of an optimized low-resource monolingual model.

#### 4.5.2 TL with Spectrogram Augmentation

For the second setting, the result shows that there is a slight increase in performance when we applied spectrogram augmentation (SA) while carrying out TL using both models. Table 4.11 tabulates the results of the STTQznet model for each SA setting after carrying our TL with SA for 100 epochs.

**Table 4.11:** Results of STTQznet model WER for TL with SA setting

| <b>Augmentation</b> | No<br>Augmentation         | Rectangle<br>(Rect) Masking                                                                                                                                                        | Time-<br>Frequency (TF)<br>Masking                                                                                                                                   | All<br>Augmentation                                                                                                                                                                |
|---------------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Parameter</b>    | All<br>parameter: <b>0</b> | Max size of the<br>cut rectangles<br>along the<br>Frequency<br>dimension: <b>50</b> ,<br><br>No. of<br>rectangular<br>masks to be cut<br>from any given<br>spectrogram: <b>5</b> , | No. of<br>frequency<br>segments to be<br>cut: <b>1</b> ,<br><br>Max no. of<br>frequencies to<br>be cut in one<br>segment: <b>27</b> ,<br><br>No. of time<br>segments | Max size of the<br>cut rectangles<br>along the<br>Frequency<br>dimension: <b>50</b> ,<br><br>No. of<br>rectangular<br>masks to be cut<br>from any given<br>spectrogram: <b>5</b> , |

**Table 4.11** continued

|  |  |                                                                          |                                                                                         |                                                                                                                                                                                                                                                                                                                                          |
|--|--|--------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  |  | Max size of cut<br>rectangles<br>along the Time<br>dimension: <b>120</b> | should be cut:<br><b>1</b> ,<br>No. of time<br>segments<br>should be cut:<br><b>100</b> | Max size of cut<br>rectangles<br>along the Time<br>dimension:<br><b>120</b> ,<br>No. of<br>frequency<br>segments to be<br>cut: <b>1</b> ,<br>Max no. of<br>frequencies to<br>be cut in one<br>segment: <b>27</b> ,<br>No. of time<br>segments<br>should be cut:<br><b>1</b> ,<br>No. of time<br>segments<br>should be cut:<br><b>100</b> |
|--|--|--------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Table 4.11** continued

|            |      |      |             |      |
|------------|------|------|-------------|------|
| <b>WER</b> | 0.26 | 0.24 | <b>0.23</b> | 0.23 |
|------------|------|------|-------------|------|

Figure 4.10 displays the WER reduction produced by each spectrogram augmentation settings on the STTQznet model after transfer learning for 100 epochs.

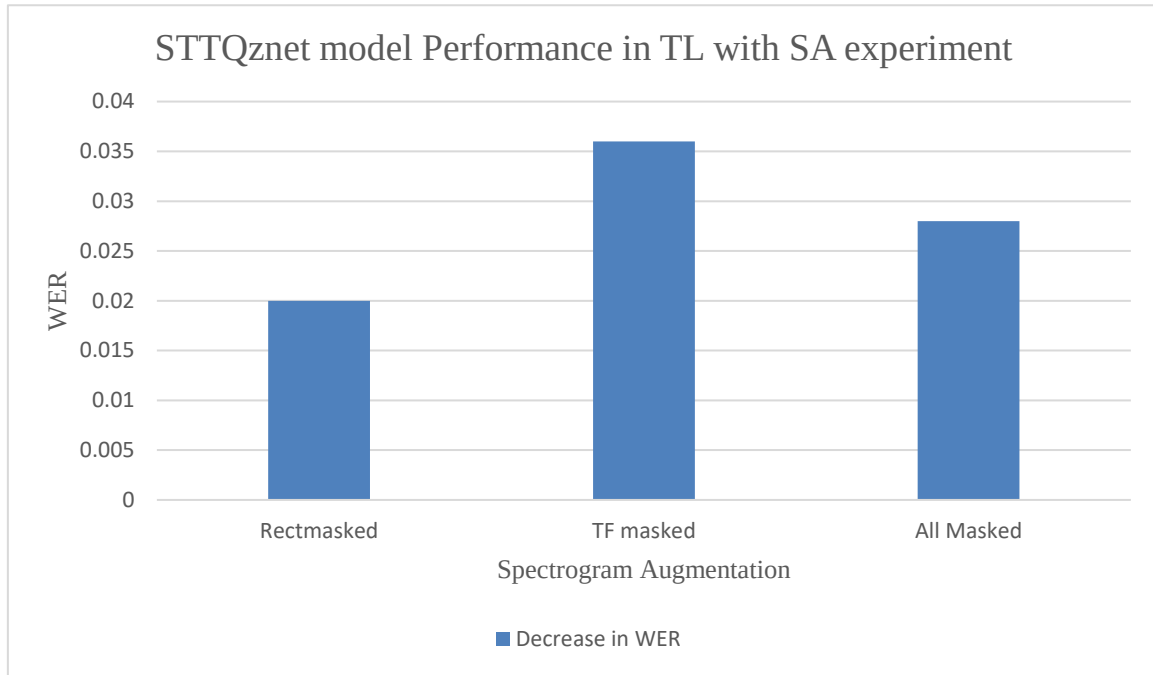


**Figure 4.10:** Graph of WER reduction produced by each spectrogram augmentation settings on the STTQznet model after transfer learning for 100 epochs

For the STTQznet model, using rectangle masking in TL decreased 0.02 WER from the initial performance of the model, while time-frequency masking decreased 0.03 WER. When both masking techniques were applied together, it also decreased 0.03 WER, achieving 0.23 WER which is similar to the performance obtained during TL with time-frequency masking run as shown in Table 4.11. This result shows that the time-frequency masking augmentation helps to further increase the TL performance most as compared to the rectangle masking. Table 4.12 presents the results of the BM model for each SA setting after carrying our TL with SA for 100 epochs.

**Table 4.12:** Result of BM model WER for TL with SA setting

| <b>Augmentation</b> | No<br>Augmentation      | Rectangle<br>(Rect)<br>Masking                                                                                                                                                                                                                                     | Time-<br>Frequency<br>(TF) Masking                                                                                                                                                                                                                                          | All<br>Augmentation                                                                                                                                                                                                                                                                                                                     |
|---------------------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Parameter</b>    | All parameter: <b>0</b> | Max size of the<br>cut rectangles<br>along the<br>Frequency<br>dimension: <b>50</b> ,<br><br>No. of<br>rectangular<br>masks to be cut<br>from any given<br>spectrogram: <b>5</b> ,<br><br>Max size of<br>cut rectangles<br>along the Time<br>dimension: <b>120</b> | No. of<br>frequency<br>segments to be<br>cut: <b>1</b> ,<br><br>Max no. of<br>frequencies to<br>be cut in one<br>segment: <b>27</b> ,<br><br>No. of time<br>segments<br>should be cut:<br><br><b>1</b> ,<br><br>No. of time<br>segments<br>should be cut:<br><br><b>100</b> | Max size of the<br>cut rectangles<br>along the<br>Frequency<br>dimension: <b>50</b> ,<br><br>No. of<br>rectangular<br>masks to be cut<br>from any given<br>spectrogram: <b>5</b> ,<br><br>Max size of<br>cut rectangles<br>along the Time<br>dimension:<br><b>120</b> ,<br><br>No. of<br>frequency<br>segments to be<br>cut: <b>1</b> , |

**Table 4.12** continued

|            |      |      |             |                                                                                                                                                                                                             |
|------------|------|------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|            |      |      |             | Max no. of<br>frequencies to<br>be cut in one<br>segment: <b>27</b> ,<br><br>No. of time<br>segments<br>should be cut:<br><br><b>1</b> ,<br><br>No. of time<br>segments<br>should be cut:<br><br><b>100</b> |
| <b>WER</b> | 0.47 | 0.41 | <b>0.40</b> | 0.40                                                                                                                                                                                                        |

Figure 4.11 displays the WER reduction produced by each spectrogram augmentation settings on the BM model after transfer learning for 100 epochs.

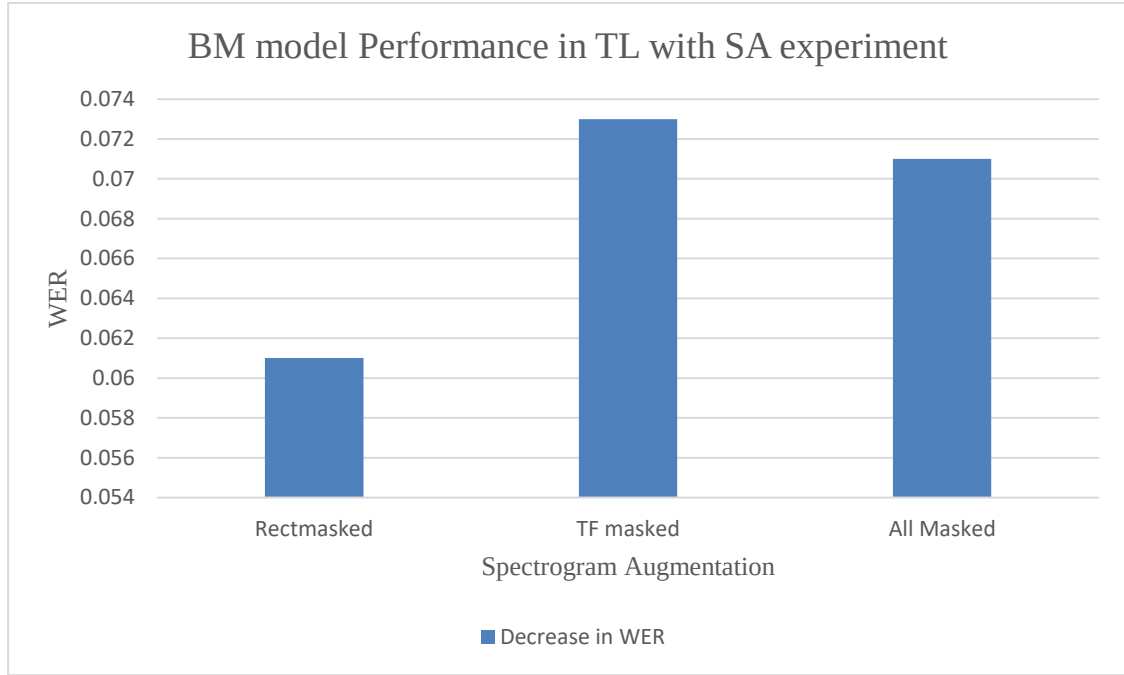


**Figure 4.11:** Graph of WER reduction produced by each spectrogram augmentation settings on the BM model after transfer learning for 100 epochs

For the BM model, using rectangle masking decreased 0.06 WER to the initial performance of the model, whereas applying time-frequency masking decreased 0.07 WER. When both masking techniques were applied together, it also decreased 0.07 WER, achieving 0.40 WER which is similar to the performance obtained during TL with time-frequency masking run as shown in Table 4.12. This result further proves that the time-frequency masking is a better spectrogram augmentation technique compared to rectangle masking when carrying out TL on the Iban corpus using different models.

Combining both time time-frequency and rectangle masking augmentation does not provide an increase in performance, this was shown in both Table 4.11 and Table 4.12. This may be due to the case of the absence of insignificant features left to be masked in our spectrogram after enough features has been masked previously using one of the augmentation techniques as spectrogram augmentation functions to reduce or if possible, eliminate noises found in the input (DeVries & Taylor, 2017; Meng et al., 2021; Park et al.,

2019; Song et al., 2020). Based on our result, time-frequency-masking masks unnecessary features better than rectangle-masking augmentation when conducting TL. Further including rectangle masking method after already applying the time-frequency mask would not contribute any increase in performance as the benefits of masking has already been fully utilized in the previous augmentation method.

Additionally, we found that by applying spectrogram augmentation in transfer learning, it enabled further reduction of a model’s WERs from a range of 0.03-0.07 WER, which leads to a slight improvement in the performance of our models.

### 4.5.3 TL with Optimized Betas

For the third setting experiment, as mentioned previously in Chapter 3, we ran the experiments based on two protocols in conducting TL with optimized betas: the “Min to Max to Min” and the “Median” protocol. The next section first presents the experiment results we have obtained from using the STTQznet model, then followed by the results obtained from using our BM model.

Table 4.13 shows the results obtained during the Min to Max to Min run using STTQznet 15×5 model applied with time-frequency masking with a learning rate of 0.001 for 100 epochs, while Table 4.14 displays the result obtained during the Median run with the same model and settings.

**Table 4.13:** Result of STTQznet model WER for Min to Max to Min run

| Betas<br>( $\beta_1, \beta_2$ ) | WER  |
|---------------------------------|------|
| 0.1, 0.1                        | 0.26 |

**Table 4.13** continued

|          |      |
|----------|------|
| 0.9, 0.9 | 0.23 |
| 0.1, 0.9 | 0.27 |
| 0.9, 0.1 | 0.22 |

**Table 4.14:** Result of STTQznet model WER for Median run

| <b>Betas</b><br><b>(<math>\beta_1</math>, <math>\beta_2</math>)</b> | <b>WER</b> |
|---------------------------------------------------------------------|------------|
| 0.1, 0.5                                                            | 0.27       |
| 0.5, 0.5                                                            | 0.25       |
| 0.9, 0.5                                                            | 0.23       |
| 0.5, 0.1                                                            | 0.24       |

Figure 4.12 displays the chart of model performance during the TL with optimization of betas experiment for STTQznet model.

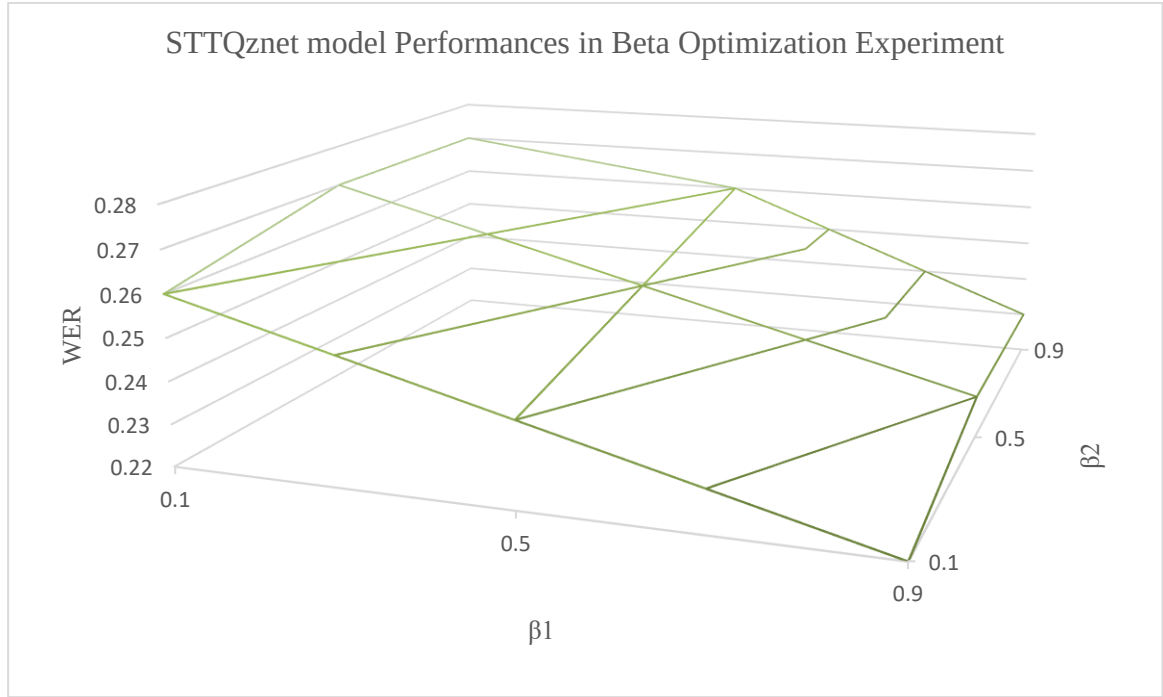


**Figure 4.12:** Chart of model performance (WER) of STTQznet model during TL with beta optimizations

Based on the results obtained, we were able to further increase the performance of our model using the STTQznet for TL applied with time-frequency masking, only when we optimized its beta from (0.8, 0.5) to (0.9, 0.1). Through the optimization of beta, we were able to further decrease 0.01 WER to the initial performance of our model when conducting TL using STTQznet applied and with time-frequency masking, recording a WER of 0.22 as its new performance. The increase in performance was very slight, other attempts to optimize the values of the betas either resulted in an increase of WER or performance stagnancy.

We then proceed to carry out TL with optimized betas for the BM 20×1 model but using only previous beta values that was able to produce the most minimal WERs. Again, the model was trained with time-frequency masking and a learning rate of 0.001 for 100 epochs. Table 4.15 presents the result obtained for the BM model during the experiment.

**Table 4.15:** Result of BM model for TL with optimized betas experiment

| <b>Betas</b><br><b>(<math>\beta_1</math>, <math>\beta_2</math>)</b> | <b>WER</b> |
|---------------------------------------------------------------------|------------|
| 0.8, 0.5                                                            | 0.40       |
| 0.9, 0.1                                                            | 0.36       |
| 0.9, 0.5                                                            | 0.36       |
| 0.9, 0.01                                                           | 0.36       |

When we used the BM model as our source model applied with same SA settings, optimizing the beta also results in an increase in performance. Our model was able to further decrease 0.04 WER from its initial performance after optimizing the first beta,  $\beta_1$  values to 0.9. We tried several other combinations of values, but it did not produce any changes in performance. Only when we optimized its values to (0.9,  $\beta_2$ ),  $\beta_2$  being any number in a range of 0.01-0.9, would the performance of our model improve. At this point, the lowest WER achieved when using the BM model as our source was 0.36.

#### **4.5.4 Findings from the implementation of the three TL settings**

Based on the results obtained in technique 3 experiments, we noticed that models that has most training hours and had been previously trained under high language resource settings would be an ideal choice for the source model of our CNN ASR model when carrying out TLs. We were also able to determine which augmentation method between rectangle and time-frequency masking technique were an ideal augmentation technique to be applied when carrying out TLs. As presented in the spectrogram augmentation experiment results, it shows that the rectangle masking technique provide better improvements for

monolingual models, but for TL and cross lingual operations between models, time-frequency masking augmentation outperforms the former method. We were also able to identify the pattern of increase and decrease of WER when we optimized the beta values of our model training. Optimizing the first beta value,  $\beta_1$ , to hold the maximum value and setting the second beta value,  $\beta_2$ , to hold the minimum value, increases the performance of our model and helps it to achieve the lowest WER possible.

Additionally, through the conduct of TL, we were able to achieve a model performance of 0.22 WER, which is a further decrease of 0.07 WER from the best performance obtained by our monolingual model (0.29 WER) and is recorded as our CNN Iban ASR model's best performance. Table 4.16 displays the summary of the lowest WER obtained by both of our model with its respective hyperparameter that helps to produce it throughout Technique 3 experiments.

**Table 4.16:** Summary of result for the lowest WER obtained from technique 3 experiment.

| Model                        | Basic TL | TL with SA | TL with Optimized Betas |
|------------------------------|----------|------------|-------------------------|
| STTQznet best hyperparameter | -        | TF Masking | Beta (0.9, 0.1)         |
| STTQznet WER                 | 0.26     | 0.23       | <b>0.22</b>             |
| BM best hyperparameter       | -        | TF Masking | Beta (0.9, 0.1)         |
| BM WER                       | 0.47     | 0.40       | <b>0.36</b>             |

Finally, comparing the improvements provided by transfer learning in our work with other research as listed in Table 2.1, we observed that we were also able to achieve almost similar contributions in terms of WER reduction with **0.07 decreased WER** to our best monolingual ASR model performance.

#### 4.5.5 Technique #3 Comparison of Reference and Hypothesis

Similar to technique 1 and 2, we recorded the hypothesis produced by the models from each TL settings experiment. Table 4.17 displays their output of transcribed text, depicting their accuracies.

**Table 4.17:** Comparison of transcription produced by the models from the TL experiments

| Hyperparameter<br><br>/Model                | Transcription                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | WER<br><br>(Best) |
|---------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| None (Original Transcript)                  | <ol style="list-style-type: none"> <li>1. pukul sepuluh malam (female_1)</li> <li>2. selamat malam diatu kami naburka berita ari radio malaysia sarawak waifm (female_2)</li> <li>3. menteri besai mutuska penemu deka begulai enggau orang di sarawak ngerami gawai saritu bejalai sereta berandau enggau mensia mayuh madahka iya menteri besai ke semua rayat lalu perintah endangibuhka semua rayat malaysia (female_3)</li> <li>4. datuk seri najib tun razak madahka iya patut ngereja utai nitihka peneka iya ti nyadi menteri besai ke semua orang lalu madahka nya ukai semina jaku politik enggau jaku kelangkung iya aja (female_4)</li> </ol> | -                 |
| Basic TL -<br>STTQznet model<br>(100 Epoch) | <ol style="list-style-type: none"> <li>1. uku <b>sepuluh malam</b> (female_1)</li> <li>2. selmamt <b>malam diatu kami naburka berita ari radio malaysia sarawak waifm</b> (female_2)</li> <li>3. <b>menteri besai mutuska penemu deka begulai enggau orang di sarawak ngerami gawai sari tu bejalai serta berandau enggau mensia mayuh madahka iya menteri besai ke semua rakyat lalu perintah</b> ending ibuhka <b>semua</b> rakyat</li> </ol>                                                                                                                                                                                                           | 0.26              |

**Table 4.17** continued

|                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |      |
|----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
|                                                                | <p><b>malaysia (female_3)</b></p> <p>4. <b>datuk seri najib tun razak madahka iya patut ngereja utai nitihka peneka iya ti nyadi menteri besai ke semua orang lalu madahka nya ukai semina jaku politik enggau jaku elangkung ia aja (female_4)</b></p>                                                                                                                                                                                                                                                                                                                                                                                                                                             |      |
| Basic TL - BM<br>model (100<br>Epoch)                          | <p>1. ku <b>sepuluh malam (female_1)</b></p> <p>2. slera <b>malam diatu kami naburka berita ari radiorn masiu sarawak waifm (female_2)</b></p> <p>3. <b>menteri besai</b> motus ka pene mbu deka b gundai <b>enggau orang di sarawak</b> nerabi <b>gawai</b> sari tu bejuadai sta brandau engau <b>mensia mayuh</b> madah ka <b>iya menteri</b> besaika <b>semua</b> rakyat <b>lalu</b> perintahdang ibuhka <b>semua rayat malaysia (female_3)</b></p> <p>4. <b>datuk seri najib tu razak</b> madahke iya <b>patut</b> nereja kain <b>nitihka bedeka iya ti dyadi menteri besai ke semua orang lalu madahka nya ukai</b> kemina <b>jaku politik enggau</b> jakub kelang kung ia ajah (female_4)</p> | 0.47 |
| TL with SA (TF-<br>Masking) -<br>STTQznet model<br>(100 Epoch) | <p>1. <b>pukul sepuluh malam (female_1)</b></p> <p>2. <b>selamat malam diatu kami nabur ka berita ari radio malaysia sarawak waim (female_2)</b></p> <p>3. <b>menteri besai mutuska penemu deka begulai enggau orang di sarawak ngerami gawai</b> saharitu <b>bejalai serka berandau enggau mensia mayuh</b> madahka <b>iya menteri besai ke semua</b> rakyat <b>lalu perintah endangibuhka semua rayat malaysia (female_3)</b></p> <p>4. <b>datuk seri najib tun razak madahka iya patut ngereja utai nitihka penekat iya ti nyadi menteri besai ke semua orang lalu madahka nya ukai semina jaku politik enggau jaku gelangkung iya aja (female_4)</b></p>                                        | 0.23 |
| TL with SA (TF-<br>Masking) - BM<br>model (100<br>Epoch)       | <p>1. ku <b>sepuluh malam (female_1)</b></p> <p>2. <b>selamat malam diatu kami naburka berita ari radioamasis sarawak waifm (female_2)</b></p> <p>3. <b>menteri besai</b> motuska penembu deke gulai <b>enggau orang di sarawak ngerami gawai</b> sari tu bejaai sreka branau <b>enggau mensia mayuh</b> madahka <b>iya menteri besai ka semua</b> rakyat <b>lalu perintah dang ibuhke semua rayat malaysia (female_3)</b></p> <p>4. <b>datuk seri najib tun razuk madahka iypatut ngereja tai nitihka penekahl iya ti nyadi menteri besai ke semua orang lalu madahka</b></p>                                                                                                                      | 0.40 |

**Table 4.17** continued

|                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |      |
|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
|                                                                  | <b>nya ukai semina jaku politik enggau jaku</b><br>pelangkun ia <b>aja</b> (female_4)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |      |
| TL with<br>Optimized Betas<br>- STTQznet<br>model (100<br>Epoch) | <ol style="list-style-type: none"> <li>1. <b>pukul sepuluh malam</b> (female_1)</li> <li>2. <b>selamat malam diatu kami</b> nabur ke <b>berita ari</b><br/><b>radio malaysia sarawak waifm</b> (female_2)</li> <li>3. <b>menteri besai mutuska penemu deka begulai</b><br/><b>enggau orang di sarawak ngerami gawai saritu</b><br/><b>bejalai</b> serta beranau <b>enggau mensia mayuh</b><br/><b>madahka iya menteri besai ke semua</b> rakyat<br/><b>lalu perintah endangibuhka semua rayat</b><br/><b>malaysia</b> (female_3)</li> <li>4. <b>datuk seri najib tun razak madahka iya patut</b><br/><b>ngereja utai nitihka peneka iya ti nyadi</b><br/><b>menteri besai ke semua orang lalu madahka</b><br/><b>nya ukai semina jaku politik enggau jaku</b><br/><b>elangkung diya aja</b> (female_4)</li> </ol> | 0.22 |
| TL with<br>Optimized Betas<br>- BM model (100<br>Epoch)          | <ol style="list-style-type: none"> <li>1. kul sepuluh <b>malam</b> (female_1)</li> <li>2. <b>selamat malam diatu kami naburka berita ari</b><br/><b>radioalaysi sarawak waifm</b> (female_2)</li> <li>3. <b>menteri besai</b> motuska penembu <b>deka</b> begeulai<br/><b>enggau orang di sarawak ngerammi gawai seari</b><br/><b>tu bejalai sereka brandau enggau mensia mayuh</b><br/><b>madahka iya menteri besai ke semua</b> rakyat<br/><b>lalu perintah endang ibuhka semua rayat</b><br/><b>malaysia</b> (female_3)</li> <li>4. <b>datuk seri najib tun razak madahka ia patut</b><br/><b>ngereja tai nitihka peneka iya te nyadi menteri</b><br/><b>besai ke semua orang lalu madahka nya ukai</b><br/><b>semina jaku politik enggau jaku pelangkong ia</b><br/><b>aja</b> (female_4)</li> </ol>          | 0.36 |

The output transcription for the third technique includes six models which is obtained from the result produced by both the STTQznet and BM model on each TL settings experiments. The lowest WER which is produced by the STTQznet model from the optimization of betas experiment displays a very high accuracy transcription with two sentences out of the four lines having perfectly transcribed while the rest having only minor errors. Out of the six models, only two were able to transcribe the first sentence of the test set with perfect accuracy, which both came from the transfer learning of the STTQznet model

onto the Iban corpus. The model trained using our BM model displays quite a number of errors in its transcription, however it was able to form the structure of the sentence almost perfectly. It was able to recognize a lot of Iban words after training even without the integration of a language model or pronunciation dictionary but was not able to predict the correct word spoken in the speech input. Based on this observation, we determined that the transfer learning method is beneficial to the CNN and under-resourced language as it significantly lowers the WER of our Iban model even without the increase in language resources for model training. This concludes the conduct of our final proposed technique experiment.

#### **4.5.6 TL with Encoder Freezing**

This section describes our findings when implementing encoder freezing during the transfer learning of a model as an attempt to further lower the WER of our model. The encoder freezing method was not included in our framework as it was not able to produce our targeted outcome but still shows some interesting results. For this additional experiment, we investigated impact of freezing the encoder during the transfer learning of a model on a limited language dataset to observe WER performances. We extracted 2H and 3H or training data from the main Iban corpus to be used as sample for this experiment and carried out transfer learning using the STTQznet model onto the isolated sample dataset while freezing and unfreezing the encoder. Table 4.18 displays the result of our TL with encoder freezing experiment.

**Table 4.18:** Comparison of WER obtained by or model when implementing TL with encoder freezing

| STTQznet model – TL with Encoder<br>Frozen |                           | STTQznet model – TL with Encoder<br>Unfrozen |                           |
|--------------------------------------------|---------------------------|----------------------------------------------|---------------------------|
| 2H Training data<br>(WER)                  | 3H Training data<br>(WER) | 2H Training data<br>(WER)                    | 3H Training data<br>(WER) |
| 0.66                                       | 0.56                      | 0.41                                         | 0.29                      |

From our findings, we could not observe any significant improvements on the WER by the models produced after implementing encoder freezing. As observed in Table 4.18, it shows that the WER produced by our model when carrying out TL with the encoder frozen was outperformed by the WER produced by our model when the encoder was not. When the encoder was frozen, the 2H model achieved a WER of 0.66 while the 3H model achieved a WER of 0.56. Meanwhile, when the encoder was not frozen, the 2H model achieved a WER of 0.41 while the 3H model achieved a WER of 0.29, which is a reduce in 0.25 and 0.27 WER to the result produced when the encoder was frozen respectively. Thus, we conclude that freezing the encoder while carrying out TL in a CNN model for an under-resourced language does not contribute to any improvement in terms of its performance.

#### 4.6 Results after Implementing the Framework

At this stage of the research, we have undergone several methods to improve the performance of our Iban CNN-ASR model. Using our framework, we have managed to reduce the WER of our model from **0.67** to **0.22**, which is a reduction of **0.45 WER** and a 32.8% increase in performance. Our best **monolingual model** was produced by using the **Quartznet 20×1** end-to-end CNN architecture trained with **rectangle masking**

**spectrogram augmentation** for **900 epoch** with a WER of 0.29 while our best **multilingual model** was produced by carrying out **transfer learning** using the **English Quartznet 15×5 pre-trained model** and finetuned to the Iban corpus applied with **spectrogram augmentation** using **time-frequency masking for 100 epoch** with a WER of 0.22. Table 4.19 displays the details of the models that produced the best performance in each technique implementation following the proposed framework.

**Table 4.19:** Improvement of WER throughout the implementation of techniques following our proposed framework

| Technique | Technique Name                                       | Model name                 | Size of Model (MB) | Total # of Parameters | WER  |
|-----------|------------------------------------------------------|----------------------------|--------------------|-----------------------|------|
| #1        | Optimization of Block, Sub-block, and Epoch          | E900-B20R1-local.nemo      | 14.91              | 3.7M                  | 0.31 |
| #2        | Application of Spectrogram Augmentation              | E900-B20R1-rectmasked.nemo | 14.91              | 3.7M                  | 0.29 |
| #3        | Transfer Learning                                    | Tl-SA-noAug-E100.nemo      | 74.54              | 18.9M                 | 0.26 |
| #3        | Transfer Learning with Spectrogram Augmentation (SA) | Tl-SA-FTMasking-E100.nemo  | 74.54              | 18.9M                 | 0.23 |

**Table 4.19** continued

|    |                                                     |                                    |       |       |             |
|----|-----------------------------------------------------|------------------------------------|-------|-------|-------------|
| #3 | Transfer Learning<br>with SA and<br>Optimized Betas | Tl-STTQznet-<br>LRBetas-<br>8.nemo | 74.54 | 18.9M | <b>0.22</b> |
|----|-----------------------------------------------------|------------------------------------|-------|-------|-------------|

As observed from Table 4.19, we saw that the size of our cross lingual model is relatively much bigger than our monolingual model. Although in general, the size of all the models did not cost too much and can be easily stored in any average computing machines. The same case also applies to the total number of parameters where our cross lingual models consist of much higher total number of parameters with 18.9M as compared to our monolingual model that only totals a 3.7M parameters. Based on these data, we believe that the size of a model and its total amount of parameters increases as the complexity of the model architecture increases. Furthermore, as the model size and number of parameters increases, the performance of the model also increases. However, in the case of same sized models, the model with an optimized hyperparameters would produce better performance and scores lower WER than the one that has not. Thus, this information further emphasises the importance of hyperparameter optimizations when improving the performance of an ASR model especially when using the CNN architecture.

#### **4.7 Comparison with previous Iban ASR model reported in literature.**

We compared our model with the works from Juan (2015) which also involves the development of Iban ASR but using different techniques and approaches. Their work was the first in many efforts in an attempt to develop an optimized ASR for the Iban language as an under-resourced language.

#### **4.7.1 Comparison of Monolingual model**

Juan (2015) were able to produce several monolingual models but their best WER recorded was using a conventional DNN ASR model that is integrated with a self-developed hybrid Iban-Malay language grapheme to phoneme (G2P) pronunciation dictionary and Iban language model which was trained on the same 7H Iban corpus that we are using as dataset, their model scored 0.18 in WER. Our best monolingual model implements end-to-end CNN ASR model with CTC decoder applied with block and sub-block optimizations and spectrogram augmentation (rectangle masking), our model scored 0.29 in WER.

Juan's (2015) ASR model was developed using DNN for the emission of probability with Hidden Markov Model (HMM) implemented in the DNN's final layer. An HMM-DNN architecture takes advantage of the DNN's strong learning power and HMM's sequential modelling activity. Since DNN only accepts fixed sized inputs, it requires HMM to handle the dynamic characteristics of speech signal that has varying length time variables (Saikia, 2017). This necessitate HMM to be integrated into any DNN-type speech recognition systems. Figure 4.13 depicts the architecture used by Juan's (2015) HMM-DNN model architecture.

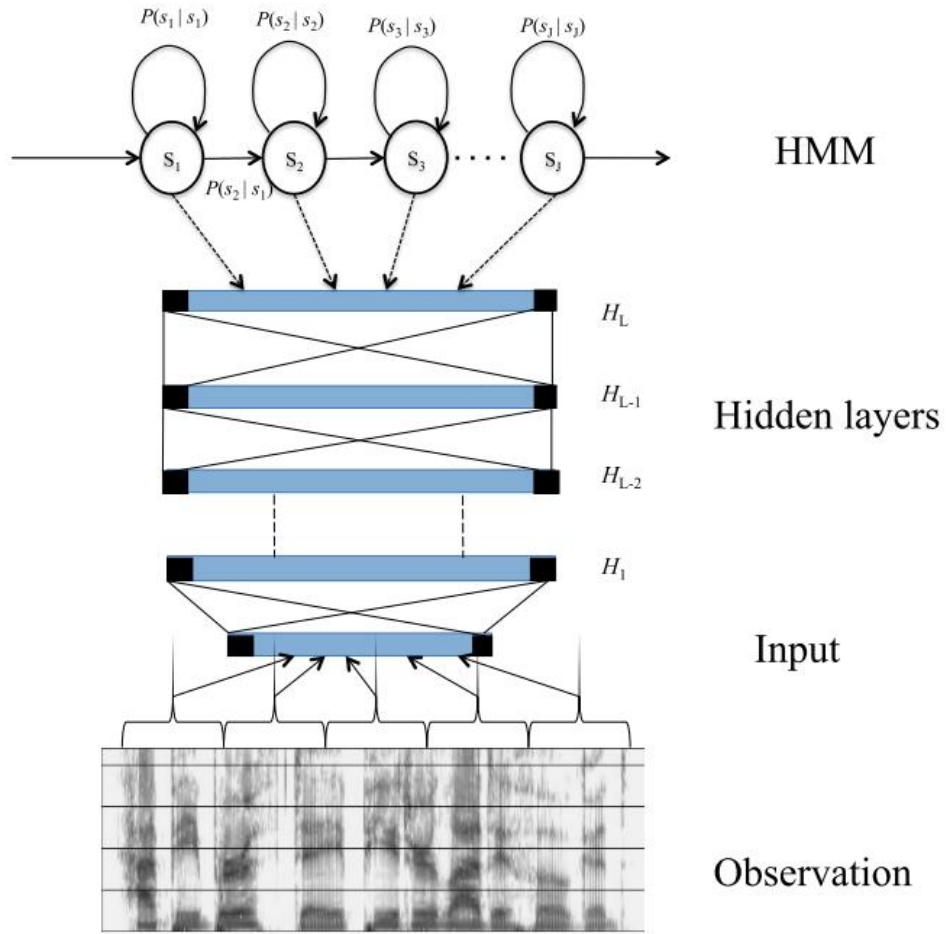


**Figure 4.13:** HMM-DNN model architecture as taken from the paper "Exploiting resources from closely-related languages for automatic speech recognition in low-resource languages from Malaysia" by Juan (2015)

As observed in Figure 4.13,  $P(X|W)$  corresponds to the acoustic probability where,

- $X$  is the sequence of observations.
- $S$  is set of emitting states;  $S_n$  indicates the  $n^{\text{th}}$  state of the set.
- $H$  is hidden layer;  $H_n$  indicates the  $n^{\text{th}}$  layer of the total hidden layers and  $H_L$  represents the final hidden layer.

Juan's (2015) speech recognition model learns via feedforward propagation whereby weights that helps to further distinguish features of speech input were distributed among the hidden layers. Each different layers were assigned with different specific weights that classifies

specific features of the speech input. Furthermore, as mentioned earlier Juan's (2015) monolingual ASR model also incorporates Iban-Malay G2P pronunciation dictionary with 7h data-trained Iban language model.

Meanwhile, our CNN ASR model exhibits end-to-end architecture whereby the whole speech recognition system was interconnected to each other with convolutional layers that helps to extract features from the speech input. Our monolingual model consists of only two sections which are the encoder section where it comprises all of the interconnected layers, and the decoder section. Our CNN model takes raw speech data as input and its features was then learned via the convolutions of filters in the convolutional layer as explained in section 1.2 of CHAPTER 1. The CNN's pooling layer which is commonly found after the final convolutional layer and before any dense layers within the CNN reduces each channel of a convolutional layer to a single value in a way that is not dependent on the resolution of the channel. This feature excludes us the requirement to implement HMM in our model as it allows the CNN to be capable of receiving variable spatial-sized inputs.

Although the result produced by our monolingual model scores a higher WER than Juan's (2015) monolingual ASR model (+0.11 WER), we find that the simple yet efficient architecture of the CNN when compared with the multiple-components integrated system of Juan's (2015) HMM-DNN ASR model, together with its capability to receive raw variable sized input, greatly eases the process of building a speech recognition system all the while achieving WER performance close to that of the existing system. This advantage serves as a motivation for us to further research the potential of CNN as acoustic model in ASR systems, which leads us to the next step, the implementation of cross lingual technique for CNN ASR model.

#### 4.7.2 Comparison of Cross-lingual model (Transfer Learning)

As for cross-lingual model, Juan's (2015) best model produced was using a pretrained Malay DNN with HMM/GMM triphone model where the triphone was employed with 39 MFCC with deltas and deltas deltas and also applied linear discriminant analysis (LDA) transformation, maximum likelihood linear transform (MLLT), and feature-space maximum likelihood linear regression (fMLLR) for speaker adaptation. To conduct transfer learning, they removed the last DNN layer of their Malay DNN model and trained it onto the 7H Iban dataset with a new Iban triphone model that was trained on new feature vectors by using the same feature transformation methods described above (Juan, 2015). Their cross-lingual model scores 0.15 in WER.

Our best cross-lingual model was obtained through the transfer learning of a pretrained NeMo end-to-end English CNN Quartznet model which we labelled as “STTQznet” model that was previously trained on several English corpus which includes the Librispeech and Wall Street Journal corpus. The pretrained model exhibits 15 block and 5 sub-blocks architecture and was used to trained on the same 7H Iban dataset while being applied with spectrogram augmentation (time-frequency masking) and optimized beta parameter values. Our best end-to-end CNN Iban ASR model scored 0.22 in WER.

Based on the results obtained from our cross lingual transfer learning experiment, our cross lingual CNN model was not able to outperform the existing HMM-DNN **cross lingual (outperformed by -0.07 WER) and monolingual model (outperformed by -0.04 WER)**. Our cross lingual CNN model however was able to produce higher WER decrease when implementing cross lingual transfer learning with **0.07 WER decrease (from 0.29 WER to 0.22 WER)** as compared to the existing model best performance of **0.03 WER decrease (from 0.18 WER to 0.15 WER)**. This suggests that cross lingual approach works

well with end-to-end architecture such as our CNN model, although, Juan's (2015) HMM-DNN monolingual model was able to learn and distinguish more features as compared to our monolingual CNN ASR model, giving their existing model the lead in terms of performance. Since the existing model has already performed excellently in the monolingual approach, our model was still not able to outperform it even when our model experienced a higher increase of performance using cross lingual method.

Another observation worth mentioning from the cross lingual experiment is how simple it is to conduct transfer learning using our end-to-end CNN model. Juan's (2015) had to train multiple triphone models using different types of data and retrain each one of them using the same feature transforms in order to carry out a successful transfer learning, whereas our end-to-end system requires only a pretrained model to conduct transfer learning onto the targeted language. Since our CNN model exhibits end-to-end architecture, each convolutional layers are interconnected with each other in the encoder, preserving the knowledge learned previously even after changing the targeted training set. This simplifies the process of transfer learning as the only thing that was left for us to do was to set a new path that points to the targeted language training dataset. Furthermore, this advantage also means that our CNN model requires less hyperparameters to be optimized (ignoring the parameters needed to be configured outside of our encoder, i.e.: parameters in triphone model, feature vectors and feature transform trainings) in order to achieve a WER score that is close to the existing DNN model's accuracy.

#### **4.8 Discussion on using English and Malay language as source language for Transfer Learning**

From our experiments, it is observed that the English language helped to decrease higher WER as compared to the Malay language when used as source language for transfer

learning. We have conducted several other experiments to investigate the viability of using English as source language for our Iban ASR model. As mentioned in Chapter 3, we know that the Malay language is closely related to the Iban language. However, from our experiment, it shows that the English language was able to produce better model performance (lower WER) as compared to the Malay language.

#### **4.8.1 Performance of the English language when used as Iban ASR Source Language in our framework.**

We conducted transfer learning experiments using the Iban, Malay, and English language to see whether we can further improve the WER of our current best CNN ASR. In our experiment, we first used the same imported Speech-to-text English pre-trained model to be trained on our imported Malay corpus. For this experiment, we used a reduced version of our Malay corpus due to GPU limit restriction as target corpus to investigate how effective the English language is as a source language on the Malay dataset. We use 8.30 GB of the original 31.2GB of Malay dataset which corresponds to 47.7 hours of speech data. We used 45.7 hours of training set which comprises of 24316 lines of sentences, while the other 1.99 hours that comprises 1000 line of sentences is used as test set. The result of the experiment is as shown in Table 4.20.

**Table 4.20:** WER of English  $\times$  Malay TL model on the Malay test set

| Size of Malay set      |       |           | WER on Reduced Malay test set |
|------------------------|-------|-----------|-------------------------------|
| # of line of sentences | Hours | Size (GB) |                               |
| 2700                   | 5.33  | 0.88      | 0.18                          |
| 5402                   | 10.68 | 1.76      | 0.18                          |
| 8104                   | 16.03 | 2.65      | 0.19                          |
| 10806                  | 21.36 | 3.52      | 0.18                          |

**Table 4.20** continued

|              |              |             |             |
|--------------|--------------|-------------|-------------|
| 13508        | 26.72        | 4.40        | 0.18        |
| 16210        | 32.10        | 5.29        | 0.17        |
| 18912        | 37.44        | 6.17        | 0.18        |
| <b>21614</b> | <b>42.88</b> | <b>7.07</b> | <b>0.16</b> |
| 24316        | 45.68        | 7.97        | 0.17        |

The best performance achieved in this experiment is when using the imported English model to be trained on the Malay dataset with 42.88 hours of train set where the model scores a 0.16 WER on the reduced Malay test set. Then, using the English-Malay transfer learning model that we have obtained, we conduct another training onto our Iban dataset. The size of the Iban corpus used on this experiment is with its original size as described in section 3.3.1. However, due to time constraint and GPU limit restriction, this operation is executable only using the transfer learning model that was previously trained on the 5.33 hours Malay train set as well as the one with 10.68 hours. We carried out further transfer learning operation using the two model on the Iban dataset with a train set of 2659 line of sentences (~7H) and tested on a test set of 473 line of sentences (~1H) while following the settings that was specified in our framework (section 0 to section 3.7.5.5). The result of the experiment is as shown in Table 4.21.

**Table 4.21:** WER of English  $\times$  Malay  $\times$  Iban TL model on the Iban test set

| Size of Malay set      |       |           | WER on Iban test set |
|------------------------|-------|-----------|----------------------|
| # of line of sentences | Hours | Size (GB) |                      |
| 2700                   | 5.33  | 0.88      | 0.24                 |

**Table 4.21** continued

|      |       |      |      |
|------|-------|------|------|
| 5402 | 10.68 | 1.76 | 0.24 |
|------|-------|------|------|

Based on the result of the experiments, when trained on the Malay dataset, the English pretrained model was able to produce a  $0.16 \approx 0.18$  WER Malay ASR model. It shows that the English model seems also viable in improving Malay-based ASR, however, in a rate that is not significant ( $\approx \pm 0.01$  WER). Furthermore, using the English  $\times$  Malay transfer learning model to carry out further transfer learning onto the Iban dataset, it also shows similar result pattern where our model was able to achieve 0.24 WER. It achieved a performance that exceeds just a mere 0.02 WER as compared to the best model obtained from our framework. On the event where we are able to work with optimal and sufficient tools (with increased or unlimited GPU resources), we hypothesize that experimenting with English and Malay as our source language in transfer learning operation may help us to produce a CNN ASR model with better WER and that is much more efficient (using less Iban speech data but with larger sized transfer learning model) as compared to our best model. However, due to time constraints and GPU limitations, we are able to investigate this study gap only up until this point.

#### **4.9 Discussion on CNN as an acoustic model for an end-to-end ASR for under-resourced language**

In this section, we discuss about the capabilities of the Convolutional Neural Network (CNN) in being the acoustic model for an ASR system. We discuss whether it helps to eliminate the requirement of speech data abundance in speech modelling, its ability to ease the development of ASR for low-resourced languages such as Iban, and other related topics regarding to its strength and weaknesses.

#### **4.9.1 Advantages of the CNN in modelling Iban ASR**

- i. The CNN is able to receive variable sized input.

The existence of pooling layer in the CNN has provided it with an advantage in being a speech recognition model. As we know, speech signals/input often comes in various length of size, this includes the input from our Iban dataset. However, using CNN means that the input of our Iban dataset is processed in a form of image which is formed by patterns of pixels. In CNNs, a ‘channel’ represents a colour, and each pixel of an image input consists of three channels: red, green, and blue. The pooling layer helps to reduce each channel to a single value in a resolution-independent method. The method typically involves taking either the average or the maximum of all of the values in the channel, leaving just a single layer of neurons, one for each channel. This dimension reduction process of the pooling layer allows CNN the advantage of being able to receive dynamic sized input, which is needed precisely in a speech recognition model and also to process our dynamic sized Iban input, excluding the need for the integration of HMM as what is needed by ANNs such as the DNN for dynamic input-size handling.

- ii. The CNN reduces the requirement for pre-processing.

The CNN’s is the only neural network that has the convolutional layer whereby each convolutional layer consists of components called ‘filter’ that is responsible to transform speech input (in a form of soundwave images/spectrogram) which was fed into a speech model to become an extractable data (matrixes) or features through the process of convolution. This helps to reduce or possibly eliminate the need for pre-processing that is commonly required by neural networks to undergo feature transform, saving computational resources.

iii. The CNN does not require human supervision in identifying important features.

All of our experiment was conducted solely using the CNN end-to-end architecture. The process of learning new features and training were done using only the CNN components and functions. Without the use of pronunciation dictionary and/or language model, the CNN was still able to capture important features that would form words and sentences to be produced as output. This displays how remarkably powerful the CNN image processing is in identifying features which is also beneficial to speech recognition task. This strength of the CNN would not only help to increase the efficiency in feature detection and word classification for under-resourced language dataset, but also allows the exclude of human supervision when training the model to recognize new patterns or features from the input.

iv. Weight sharing in an end-to-end architecture increase accuracy in identical feature detection.

CNN same weight sharing across all input samples (pixels) helps the model to detect common 'local' features across an image. This feature provides the CNN the ability to recognize any identical feature that it has previously learned at any layers in the later stage of model training. This feature enhances the speed of similar feature detection as well as model accuracy.

v. End-to-end architecture simplifies training and transfer learning processes.

The CNN model end-to-end architecture interconnects all of the components in the neural network, which means that learning process occurs all around the model and requires only a single model to map speech data. This simplifies the process of developing an ASR model as compared to the traditional ones (which commonly requires trainings of triphones,

isolated speaker adaptive system, etc.). Furthermore, as explained in section 4.7.2, the end-to-end architecture of CNN also simplifies the procedure to conduct model transfer learning.

#### **4.9.2 Disadvantages of the CNN in modelling an Iban ASR**

- i. CNN model training duration increases as parameter value increases.

It was also observed from our experiment that as we increase the complexity of our CNN architecture, the model training duration also takes a longer time to complete especially when the size of data is large. An increase in the value of parameter such as no. of blocks and no. of sub-blocks will also increase the number of layers that exist in the CNN architecture, i.e.: pooling layers, RELU, convolutional layers. The time taken for a model training to complete increases as data has to be passed through each of the layers. It would cost even longer time when the computing machine used does not possess a GPU (up to weeks or even months of training duration). A high number of epochs in training would also result with lengthy training duration.

- ii. CNN has high computational requirements.

CNNs are computationally expensive to train and run. This is because on each round of a training session, they need to update all parameters that involved in the training such as the number of filters. Relating to the previous mentioned disadvantage, since CNN typically consist of many layers and is taking lengthy duration to finish training, this also means that computational resources is being utilized for long amount of duration. A computing machine which does not has the capacity to provide CNN the computational requirements it needed may end up interrupting the model training and causes progress loss due to insufficient computational resources.

- iii. Convolutional neural networks recognize input as clusters of pixels which are arranged in distinct patterns.

CNN learns via feature extraction only and does not gain any knowledge from previous correctly predicted words. As such, it does not understand any words that it has predicted correctly and could not understand the relationship between words in a sentence. An example case of this weakness is shown in Figure 4.14 which was taken from the output of our best model.

```
id: (male_255)
Scores: (#C #S #D #I) 6 3 1 1
REF:  iya madahka ENTARA BULAN          satu ENGGAU  bulan dua **** HARIBULAN enam
HYP:  iya madahka ***** ANTARABULAN satu ENGAGAI  bulan dua HARI BULAN      enam
Eval:                                D      S                      S                      I      S
```

**Figure 4.14:** Example of an error output from our best model

The original audio speech input as shown in Figure 4.14 actually reads “iya madahka entara bulan satu enggau bulan dua haribulan enam”, but our model was not able to accurately predict the input and instead predict it into “iya madahka antarabulan satu engagai bulan dua hari bulan enam”. The phrase “entara bulan” as taken from the input which means “either in the month of”, is a quite straightforward-meaning Iban phrase where “entara” is the iban language for “either” while “bulan” is “month”. However, our CNN ASR model was not able to accurately predict this phrase as it did not understand any of the words predicted even after a 100 epoch runs and more than 2K sentences of training data. The next phrase of the sentence which goes “bulan satu enggau bulan dua” means “the first month or second” is also another quite straightforward Iban phrase that our CNN ASR model was not able to accurately predict. This shows that the CNN would only recognize features as pixels and without any knowledge of understanding it.

#### 4.10 Chapter Summary

In this chapter, we have described the results that we have obtained from our experiments. We have presented our WERs obtained from the preliminary experiment and from the three techniques proposed: hyperparameter optimization, spectrogram augmentation, and cross-lingual transfer learning method in the form of tables and graphs. Furthermore, we have also described the output transcription of our model on each stage's experiments and its increase in accuracy on each implementation of the three techniques. A table that summarizes the resulting WER for each technique proposed as well as their model details has also been included. Moreover, we presented an analysis of result for the implementation of encoder freezing onto our model which was not included in our proposed framework, where we explained why we did not propose the encoder freezing method as our technique to improve the CNN model. As additional analysis, we also conducted a performance and method comparison of our model against a previously developed Iban ASR model. Finally, a discussion on CNN as an acoustic model for ASR model which talks about its strength and weakness has been conducted in this chapter. The next chapter will talk about our conclusion, contribution, and limitation of this research as well as future work.

## **CHAPTER 5**

### **CONCLUSION AND FUTURE WORKS**

#### **5.1 Introduction**

In this chapter, we discuss the work that has been done to conclude this research. We summarize each chapter in the thesis as we relate it to our research questions and objectives. We conclude whether our research and our proposed framework has able to solve all of our research questions and achieve each of our research objectives. We also discuss about our research contributions as well as our limitations. Finally, we conclude the whole research by presenting potential improvements and future works that can be done to our framework as well as for the development of the Iban ASR model.

#### **5.2 Summary of Research**

The aim of our research is to investigate whether CNN-based architecture could produce an ASR that is able to transcribe under-resourced language accurately without the need of abundant resource, specifically for the Iban language. Thus, we designed and proposed a framework that would help us to analyse the capability of CNN as well as develop a CNN ASR model. Five chapters have been written in this thesis to realize the aim of our research.

In Chapter 1, we presented our case study and the reason behind the conduct of our research. We presented the study gap that exist in the ASR development field for the Iban language as well as presented our problem statement. We listed out two research questions and three research objectives in correlation to the solving of our problem statement. We specified our scope of research and its significance towards speech technology and we concluded the chapter by presenting the outline of our overall research.

Chapter 2 presented our literature review where we discussed existing works and general knowledge about ASR and under-resourced languages. We studied about the traditional method of developing an ASR model as we investigated its basic architecture which requires the integration of three critical component, we found out how its pipeline was not effective when being implemented on under-resourced languages. In the same chapter, we introduced the background of our targeted language and its origin as well as previous scientific work that has been done in relation to the development of ASR model using our targeted language. Furthermore, we also studied the architecture of the CNN as acoustic model to fulfil our first research objective **(RO1)** as well as its implementation in the field of speech recognition technology. We found out about the strength of CNN in handling the issue of data scarcity faced by under-resourced languages in the development of ASR model by exhibiting end-to-end architecture **(RO1)**. We explored works that involves the CNN in speech modelling and trending techniques that is being implemented in model development that helps to further optimize the architecture of neural network-based ASR models. At the end of chapter 2, we presented currently available tools that can be used for modelling speech recognition system.

We described our methodology in Chapter 3, where we present systematic steps to implement CNN acoustic model in ASR development by proposing a framework to develop an optimized end-to-end CNN ASR model for the Iban language, fulfilling research question I and research objective II **(RQ1 and RO2)**. We began by introducing our dataset that was being used in the research experiment which includes our target language dataset, followed by the introduction of our framework. We described our framework which implements three techniques and two approaches: monolingual and cross-lingual **(RO2)**, then, we introduced the tools that we have used to conduct our experiments. We also presented the evaluation

metrics, the WER, that was being used to evaluate the result of our experiments (**RO3**). After introducing our framework and our experimental setup, we described the procedure involved in each of the steps being implemented in our framework. The three techniques that was proposed in our framework to improve the performance of our baseline model and optimize the CNN architecture (**RQ1**) were described comprehensively in this chapter. Finally, we included YAML files that was created for each model architecture that was being used in our research experiment at the end of Chapter 3.

Our Chapter 4 presents the results obtained from all of conducted experiments described in the previous chapter. This includes the baseline results obtained from our preliminary experiment, the results obtained from our model improvement strategy experiments which implements three techniques: hyperparameter optimization experiments (technique #1), spectrogram augmentation experiments (technique #2), and cross-lingual transfer learning experiment (technique #3), fulfilling our third research objective and research question II (**RQ2 and RO3**). From this chapter, we found out how to obtain an optimized CNN ASR model based on the results of our experiments (**RQ1**) and identify the hyperparameters that exist in the architecture of the CNN which influences model performances (**RQ2**). We observed how the three techniques being implemented contributes to the further improvement of our Iban speech recognition model which leads to the development of ASR model with a WER of 0.22 (**RO3**). Additionally, we also conducted comparison between our best model performance produced from this research and existing Iban ASR model which was previously developed by Juan (2015) in this chapter. We analysed the differences between the model based on the performance they achieved when implementing monolingual and cross-lingual approaches improvement techniques. Finally,

we conduct analysis between the advantages and disadvantages of using CNN as acoustic model in the development of Iban ASR model at the end of Chapter 4.

### **5.3 Research Contribution**

From our research, several contributions have been made. This includes:

#### **5.3.1 CNN ASR framework for Iban and under-resourced languages**

We have successfully designed a framework to develop an end-to-end CNN ASR model, specifically for the Iban language, which is suitable for the case of under-resourced languages.

#### **5.3.2 A CNN Iban ASR performance benchmark**

Aside from acting as speech modelling reference, the proposed framework also contributes to the setting up of performance benchmark reference for future Iban ASR model development that is using the CNN.

#### **5.3.3 Framework to optimize block, sub-block, and epoch of Quartznet CNN**

Our framework covers the studies of hyperparameter optimization in the CNN architecture which presents the most optimized architecture to follow when developing a CNN-based ASR model using Quartznet end-to-end CNN variant. We identified the priority ranking of optimization between the hyperparameter number of blocks, sub-blocks, and epoch, in obtaining optimized CNN models with the best WER performance and investigated the best combination of values between the three hyperparameters.

#### **5.3.4 Studies on how Spectrogram Augmentation benefits the CNN in modelling an Iban ASR.**

From our research, we also cover the studies on the effect of spectrogram augmentation towards speech input and its effects on the performance of CNN ASR model.

We investigated how noise masking techniques in spectrogram augmentation helped in further increasing the performance of under-resourced CNN ASR model and is viable to help reduce the necessity for abundant speech data in speech modelling. We have also discovered the benefits of implementing spectrogram augmentation technique during the conduct of transfer learning, which by masking unnecessary noises in the input, helps to increase efficiency of our source model in feature extraction when trained using our targeted language dataset. To our knowledge, the spectrogram augmentation technique has never been applied in any previous Iban ASR model development. Thus, the model performance results obtained during the spectrogram augmentation experiments which serves as a benchmark reference for future development also counted as a contribution from our research.

#### **5.3.5 Studies on Transfer Learning in modelling a CNN Iban ASR**

Furthermore, our framework covers the studies of transfer learning implementation on the end-to-end CNN architecture where we studied the effects of transfer learning on end-to-end CNN ASR model for the under-resourced Iban language. Our research presents the very first data analysis on the performance of the conduct of transfer learning using end-to-end CNN architecture on the Iban language. Our research framework is also the first to present the studies on the effect of beta values in determining the learning curve of the model during transfer learning process in producing Iban CNN ASR system.

#### **5.3.6 Production of the first CNN Iban ASR model**

Finally, our research has helped to produce the very first optimized Iban end-to-end Quartznet CNN ASR model which scores a model performance of 0.22 WER.

## 5.4 Research Limitation

Although we have successfully proposed our research framework, it has several weaknesses and limitations that we did not cover in its implementation.

Our framework included the hyperparameter optimization technique which studies the effect of value combinations of block, sub-block, and epoch hyperparameter. However, during its implementation, the optimization for each hyperparameter values was not automated but was manually investigated one by one, combinations by combinations and hyperparameter by hyperparameter. This leads to longer time consumption when implementing Technique #1 as it involves manual experimentation steps of many different combinations of hyperparameter values followed by the observation of result as we did not cover the studies on automated techniques for hyperparameter optimization. Since we have decided to solely study the capability of CNN overcoming data scarcity in under-resourced language, we have also not studied the performance of our Iban CNN ASR model when integrated with language model. Although there are existing works on CNN that includes the integration of language model to further improve the accuracy of an ASR model, doing this however changes the end-to-end architecture of CNN into a more traditional design and would demand speech data abundancy for our proposed framework. Another limitation of our framework is that it requires the use of graphical processing unit (GPU) resources. Without GPU, the machine implementing our framework would have to take a very long time to finish one technique. The computation time taken for the training of our optimized 20×1 Quartznet CNN ASR model was around 4 hours using GPU resources. Without GPU, the training might take longer time, possibly a day or two just to finish Technique #1. Furthermore, in the implementation of Technique #3, we studied the effect of optimizing beta values to better learn its impact towards improving the model learning curve. The study

however investigated only the beta hyperparameter values and did not include the learning rate (LR) hyperparameter. The study did not include the LR hyperparameter because doing so costs us more time which would exceed the limit of this research period as it would broaden up the scope of optimization for the beta hyperparameter. This, however, can be considered as a task for future work. Finally, another limitation of our framework is it has not investigated a lot of other recent spectrogram augmentation techniques that would be very beneficial for CNN in the implementation of Technique #2 aside from the two mentioned in the methodology (see section 3.7.4). In Chapter 2, MixUp, MixSpeech (Meng et al., 2021) and SpecSwap (Song et al., 2020) was mentioned to be implemented in CNNs alongside the SpecAugment (Park et al., 2019) technique in spectrogram augmentation. These few recent techniques however were not included into the framework due to time constraints and GPU resource limitations. This limitation can also be considered as task to be done in future work. Several tasks have been considered for future work; the next section discuss the potential improvements that can be done to further improve our framework.

## **5.5 Future Work**

As mentioned in the previous section, several future works have been identified as an effort to further improve our proposed framework for the building of an optimized Iban CNN ASR model. This includes the conduct of studies on automated hyperparameter optimization techniques which could improve the efficiency in implementing Technique #1 of our framework. Also, the optimization of more hyperparameters such as the learning rate (LR) of the CNN model when conducting transfer learning in Technique #3. As mentioned earlier, the implementation of MixUp, MixSpeech (Meng et al., 2021) and SpecSwap (Song et al., 2020) as recent spectrogram augmentation techniques in Technique #2 would also be a good study for future work.

In a technical perspective, viewing our framework as a module also help us to determine potential improvements that can be done in the future for the design of a better Iban ASR model development framework, for example, the implementation of CNN in wav2vec architectures as described in many works (Liu et al., 2023; Rouhe et al., 2022; Sadhu et al., 2021). Using Wav2vec end-to-end model is also one of the trending approaches currently being implemented in the development of supervised and unsupervised speech recognition system. Investigating the potential of our CNN framework to be integrated as a module into a wav2vec framework that utilizes the CNN, resulting with the production of a bigger scale framework consisting of many different levels of development, would be an interesting study to be conducted.

Finally, investigating other different end-to-end architectures that may help under-resourced languages to exclude the necessity for abundant speech resources in the development of ASR model such as the non-autoregressive speech recognition model that utilizes CNN convolutional layers (Qian et al., 2022), as future work would also help us to design a better framework architecture in building a better Iban ASR model.

## REFERENCES

- Afaq, S., & Rao, S. (2020). Significance Of Epochs On Training A Neural Network. *International Journal of Scientific and Technology Research*, 19(6), 485–488.  
Retrieved from [www.ijstr.org](http://www.ijstr.org)
- Ali, A., & Renals, S. (2020). Word error rate estimation without asr output: E-WER2. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2020-Octob*, 616–620.
- Alsayadi, H. A., Abdelhamid, A. A., Hegazy, I., & Fayed, Z. T. (2021). Arabic speech recognition using end-to-end deep learning. *IET Signal Processing*, 15(8), 521–534.
- Aman, R., Nur Latif, M., & Hamid, S. A. (2019). Linguistik Bandingan Bahasa Iban. *Jurnal Melayu Bil. 18 (2) 2019*, 18(2), 315–320.
- Andonie, R., & Florea, A. C. (2020). Weighted random search for CNN hyperparameter optimization. *International Journal of Computers, Communications and Control*, 15(2), 1–11.
- Arnel Fajardo, F. R. J. (2020). Convolutional Neural Network for Automatic Speech Recognition of Filipino Language. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(1.1 S I), 34–40.
- Aszemi, N. M., & Dominic, P. D. D. (2019). Hyperparameter optimization in convolutional neural network using genetic algorithms. *International Journal of Advanced Computer Science and Applications*.
- Bhatt, S., Jain, A., & Dev, A. (2020). Acoustic Modeling in Speech Recognition: A Systematic Review. *International Journal of Advanced Computer Science and*

*Applications*, 11(4), 397–412.

- Brownlee, J. (2020, August 20). A Gentle Introduction to the Rectified Linear Unit (ReLU). Retrieved April 12, 2021, from <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>
- Carneiro, T., Medeiros Da Nobrega, R. V., Nepomuceno, T., Bian, G.-B., De Albuquerque, V. H. C., & Filho, P. P. R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, 6, 61677–61685.
- Das, N., Bodapati, S., Sunkara, M., Srinivasan, S., & Chau, D. H. (2021). Best of Both Worlds: Robust Accented Speech Recognition with Adversarial Transfer Learning. In *Interspeech 2021* (Vol. 6, pp. 1314–1318). ISCA: ISCA.
- DeVries, T., & Taylor, G. W. (2017). Improved Regularization of Convolutional Neural Networks with Cutout. Retrieved from <http://arxiv.org/abs/1708.04552>
- Fohr, D., & Illina, I. (2021). BERT-Based Semantic Model for Rescoring N-Best Speech Recognition List. In *Interspeech 2021* (Vol. 2, pp. 1867–1871). ISCA: ISCA.
- Grossi, E., & Buscema, M. (2007). Introduction to artificial neural networks. *European Journal of Gastroenterology & Hepatology*, 19(12), 1046–1054.
- Guliani, D., Beaufays, F., & Motta, G. (2021). Training Speech Recognition Models with Federated Learning: A Quality/Cost Framework. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Vol. 2021-June, pp. 3080–3084). IEEE.
- Hassan, M. A., Rehmat, A., Ghani Khan, M. U., & Yousaf, M. H. (2022). Improvement in Automatic Speech Recognition of South Asian Accent Using Transfer Learning of DeepSpeech2. *Mathematical Problems in Engineering*, 2022(July 2018), 1–12.

- He, D., Lim, B. P., Yang, X., Hasegawa-Johnson, M., & Chen, D. (2018). Improved ASR for Under-Resourced Languages Through Multi-Task Learning with Acoustic Landmarks. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2018-Septe*, 2618–2622.
- Huang, J., Kuchaiev, O., O'Neill, P., Lavrukhin, V., Li, J., Flores, A., ... Ginsburg, B. (2020). Cross-Language Transfer Learning, Continuous Learning, and Domain Adaptation for End-to-End Automatic Speech Recognition. *Proceedings - IEEE International Conference on Multimedia and Expo*.
- Iwama, F., & Fukuda, T. (2019). Automated Testing of Basic Recognition Capability for Speech Recognition Systems. In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)* (pp. 13–24). IEEE.
- Jain, A., Rouhe, A., Grönroos, S.-A., & Kurimo, M. (2020). Finnish ASR with Deep Transformer Models. In *Interspeech 2020* (Vol. 2020-Octob, pp. 3630–3634). ISCA: ISCA.
- Juan, S. S. (2015). Exploiting resources from closely-related languages for automatic speech recognition in low-resource languages from Malaysia, (July 2015).
- Juan, S. S., Besacier, L., Lecouteux, B., & Dyab, M. (2015). Using resources from a closely-related language to develop ASR for a very under-resourced language: A case study for iban. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2015-Janua*, 1270–1274.
- Kaldi: About the Kaldi project. (n.d.). Retrieved August 12, 2021, from <https://kaldi-asr.org/doc/about.html>
- Kriman, S., Beliaev, S., Ginsburg, B., Huang, J., Kuchaiev, O., Lavrukhin, V., ... Zhang, Y.

- (2020). Quartznet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Vol. 2020-May, pp. 6124–6128). IEEE.
- Krishna, D. N., Wang, P., & Bozza, B. (2021). Using large self-supervised models for low-resource speech recognition. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 1*, 371–375.
- Kuchaiev, O., Li, J., Nguyen, H., Hrinchuk, O., Leary, R., Ginsburg, B., ... Cohen, J. M. (2019). NeMo: a toolkit for building AI applications using Neural Modules, 1–8. Retrieved from <http://arxiv.org/abs/1909.09577>
- Kurata, G., Saon, G., Kingsbury, B., Haws, D., & Ee, E. (2021). Improving Customization of Neural Transducers by Mitigating Acoustic Mismatch of Synthesized Audio Abstract Customization of automatic speech recognition ( ASR ) models using text data from a target domain is essential to deploying the added mapping netwo, 2027–2031.
- Lekshmi, K. R., & Sherly, E. (2021). An acoustic model and linguistic analysis for Malayalam disyllabic words: a low resource language. *International Journal of Speech Technology, 24*(2), 483–495.
- Li, J., Lavrukhin, V., Ginsburg, B., Leary, R., Kuchaiev, O., Cohen, J. M., ... Gadde, R. T. (2019). Jasper: An End-to-End Convolutional Neural Acoustic Model. In *Interspeech 2019* (Vol. 2019-Sept, pp. 71–75). ISCA: ISCA.
- Liu, A. H., Hsu, W.-N., Auli, M., & Baevski, A. (2023). Towards End-to-End Unsupervised Speech Recognition. In *2022 IEEE Spoken Language Technology Workshop (SLT)* (pp. 221–228). IEEE.

- Lukeš, D., Kopřivová, M., Komrsková, Z., & Poukarová, P. (2019). Pronunciation variants and ASR of colloquial speech: A case study on Czech. *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, 2704–2709.
- Luo, J., Wang, J., Cheng, N., Zheng, Z., & Xiao, J. (2022). Adaptive Activation Network for Low Resource Multilingual Speech Recognition. *Proceedings of the International Joint Conference on Neural Networks, 2022-July*.
- Mamyrbayev, O., Mekebayev, N., Turdalyuly, M., Oshanova, N., Ihsan Medeni, T., & Yessentay, A. (2020). Voice Identification Using Classification Algorithms. In *Intelligent System and Computing*.
- Meng, L., Xu, J., Tan, X., Wang, J., Qin, T., & Xu, B. (2021). MixSpeech: Data augmentation for low-resource automatic speech recognition. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2021-June(2017)*, 7008–7012.
- Michael, S. O., Juan, S. S., & Mit, E. (2022). Preliminary Evaluation of Convolutional Neural Network Acoustic Model for Iban Language Using NVIDIA NeMo. *Journal of Telecommunications and Information Technology*, 1(2022), 43–53.
- Mon, A. Y. E. N. (2019). MYANMAR LANGUAGE CONTINUOUS SPEECH RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK □ CNN □ AYE NYEIN MON UNIVERSITY OF COMPUTER STUDIES , YANGON Myanmar Language Continuous Speech Recognition Using Convolutional Neural Network □ CNN □ Aye Nyein Mon Univers.
- Mou, J., & Li, J. (2020). Effects of Number of Filters of Convolutional Layers on Speech Recognition Model Accuracy. *Proceedings - 19th IEEE International Conference on*

*Machine Learning and Applications, ICMLA 2020*, 971–978.

Overview of the CMUSphinx toolkit – CMUSphinx Open Source Speech Recognition.

(n.d.). Retrieved August 13, 2021, from <https://cmusphinx.github.io/wiki/tutorialoverview/>

Palaz, D., Collobert, R., & Magimai-Doss, M. (2013). Estimating phoneme class conditional probabilities from raw speech signal using convolutional neural networks. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 1766–1770.

Palaz, D., Magimai.-Doss, M., & Collobert, R. (2015). Convolutional Neural Networks-based continuous speech recognition using raw speech signal. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, 2015-April*, 4295–4299.

Panda, S. P. (2017). Automated Speech Recognition System in, (Iccmc), 302–306.

Park, D. S., Chan, W., Zhang, Y., Chiu, C. C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). SpecAugment: A simple data augmentation method for automatic speech recognition. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2019-Septe*, 2613–2617.

Parry, J., Palaz, D., Clarke, G., Lecomte, P., Mead, R., Berger, M., & Hofer, G. (2019). Analysis of deep learning architectures for cross-corpus speech emotion recognition. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2019-Septe*, 1656–1660.

Pulugundla, B., Baskar, M. K., Kesiraju, S., Egorova, E., Karafiát, M., Burget, L., & Cernocký, J. (2018). BUT system for low resource Indian language ASR. *Proceedings*

- of the Annual Conference of the International Speech Communication Association, *INTERSPEECH*, 2018-Septe(September), 3182–3186.
- Qian, Y., Zhuang, X., Zhang, Z., Zhou, L., Lin, X., & Wang, M. (2022). Non-Autoregressive Speech Recognition with Error Correction Module. In *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (pp. 1103–1108). IEEE.
- Qin, S., Wang, L., Li, S., Dang, J., & Pan, L. (2022). Improving low-resource Tibetan end-to-end ASR by multilingual and multilevel unit modeling. *Eurasip Journal on Audio, Speech, and Music Processing*, 2022(1), 2.
- Rohani, M. (2003). Perkaitan Bahasa Melayu dan Bahasa Iban: Satu Tinjauan Ringkas. *Jurnal Bahasa*, 457–475. Retrieved from [http://scholar.google.co.id/scholar?q=jurnal+bahasa&hl=id&as\\_sdt=0,5#8](http://scholar.google.co.id/scholar?q=jurnal+bahasa&hl=id&as_sdt=0,5#8)
- Rouhe, A., Virkkunen, A., Leinonen, J., & Kurimo, M. (2022). Low Resource Comparison of Attention-based and Hybrid ASR Exploiting wav2vec 2.0. In *Interspeech 2022* (pp. 3543–3547). ISCA: ISCA.
- S., A. C., & Ramakrishnan, A. G. (2021). CTC-Based End-To-End ASR for the Low Resource Sanskrit Language with Spectrogram Augmentation. In *2021 National Conference on Communications (NCC)* (pp. 1–6). IEEE.
- S, S. K., & S H, B. (2023). Improving the Performance of Speech Recognition feature selection using Northern Goshawk Optimization. In *2022 International Interdisciplinary Humanitarian Conference for Sustainability (IIHC)* (pp. 556–559). IEEE.
- Sadhu, S., He, D., Huang, C.-W., Mallidi, S. H., Wu, M., Rastrow, A., ... Maas, R. (2021).

- wav2vec-C: A Self-Supervised Model for Speech Representation Learning. In *Interspeech 2021* (Vol. 3, pp. 711–715). ISCA: ISCA.
- Saha, S. (2018, December 16). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science. Retrieved April 12, 2021, from <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- Saikia, P. (2017). Hmm-Dnn Speech Recognition Techniques: a Review. *International Journal of Development Research*, 7(7), 14068–14072. Retrieved from <http://www.journalijdr.com>
- Santana, L. M. Q. De, Santos, R. M., Matos, L. N., & Macedo, H. T. (2018). Deep Neural Networks for Acoustic Modeling in the Presence of Noise. *IEEE Latin America Transactions*, 16(3), 918–925.
- Soe, W., & Theins, Y. (2015). Syllable-based Myanmar language model for speech recognition. In *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)* (pp. 291–296). IEEE.
- Song, X., Wu, Z., Huang, Y., Su, D., & Meng, H. (2020). SpecSwap: A simple data augmentation method for end-to-end speech recognition. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2020-Octob*, 581–585.
- STT En Quartznet15x5 | NVIDIA NGC. (n.d.). Retrieved December 21, 2022, from [https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt\\_en\\_quartznet15x5](https://catalog.ngc.nvidia.com/orgs/nvidia/teams/nemo/models/stt_en_quartznet15x5)
- Tan, T.-P., Besacier, L., & Lecouteux, B. (2014). Acoustic model merging using acoustic models from multilingual speakers for automatic speech recognition. In *2014*

*International Conference on Asian Language Processing (IALP)* (pp. 42–45). IEEE.

Thai, B., Jimerson, R., Ptucha, R., & Prud’hommeaux, E. (2020). Fully Convolutional {ASR} for Less-Resourced Endangered Languages. *Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL)*, (May), 126–130. Retrieved from <https://www.aclweb.org/anthology/2020.sltu-1.17>

Ujjwalkarn. (2016, August 11). An Intuitive Explanation of Convolutional Neural Networks – the data science blog. Retrieved April 12, 2021, from <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

van der Westhuizen, E., Kamper, H., Menon, R., Quinn, J., & Niesler, T. (2021). Feature learning for efficient ASR-free keyword spotting in low-resource languages. *Computer Speech and Language*, 71(July 2020), 101275.

Wei, X., Cucchiarini, C., van Hout, R., & Strik, H. (2022). Automatic Speech Recognition and Pronunciation Error Detection of Dutch Non-native Speech: cumulating speech resources in a pluricentric language. *Speech Communication*, 144(May), 1–9.

Xie, L., & Yuille, A. (2017). Xie\_Genetic\_CNN\_ICCV\_2017\_paper.pdf, 1379–1388.

Xu, F., Dan, Y., Yan, K., Ma, Y., & Wang, M. (2022). Low-Resource Language Discrimination toward Chinese Dialects with Transfer Learning and Data Augmentation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 21(2), 1–21.

Yi, C., Zhou, S., & Xu, B. (2021). Efficiently Fusing Pretrained Acoustic and Linguistic Encoders for Low-Resource Speech Recognition. *IEEE Signal Processing Letters*, 28, 788–792.

- Yu, C., Chen, Y., Li, Y., Kang, M., Xu, S., & Liu, X. (2019). Cross-Language End-to-End Speech Recognition Research Based on Transfer Learning for the Low-Resource Tujia Language. *Symmetry*, 11(2), 179.
- Zhan, Q., Xie, X., Hu, C., Zuluaga-Gomez, J., Wang, J., & Cheng, H. (2021). Domain-Adversarial Based Model with Phonological Knowledge for Cross-Lingual Speech Recognition. *Electronics*, 10(24), 3172.
- Zhang, N., Wang, J., Wei, W., Qu, X., Cheng, N., & Xiao, J. (2021). CACnet: Cube Attentional CNN for Automatic Speech Recognition. In *2021 International Joint Conference on Neural Networks (IJCNN)* (Vol. 2021-July, pp. 1–7). IEEE.

## APPENDICES

APPENDIX B: YAML file of the model architecture specified during the conduct of the preliminary experiment.

---

```
1  name: &name "QuartzNet15x5"
2  sample_rate: &sample_rate 16000
3  repeat: &repeat 5
4  dropout: &dropout 0.0
5  separable: &separable true
6  labels: &labels [" ", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
7    "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", ""]
8
9  model:
10   train_ds:
11     manifest_filepath: ???
12     sample_rate: 16000
13     labels: *labels
14     batch_size: 32
15     trim_silence: True
16     max_duration: 16.7
17     shuffle: True
18     is_tarred: False
19     tarred_audio_filepaths: null
20
21   validation_ds:
22     manifest_filepath: ???
23     sample_rate: 16000
24     labels: *labels
25     batch_size: 32
26     shuffle: False
27
28   preprocessor:
29     cls: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
30     params:
31       normalise: "per_feature"
32       window_size: 0.02
33       sample_rate: *sample_rate
34       window_stride: 0.01
35       window: "hann"
36       features: &n_mels 64
37       n_fft: 512
38       frame_splicing: 1
39       dither: 0.00001
40       stft_conv: false
41
42   spec_augment:
43     cls: nemo.collections.asr.modules.SpectrogramAugmentation
44     params:
45       rect_freq: 50
46       rect_masks: 5
47       rect_time: 120
48
49   encoder:
50     cls: nemo.collections.asr.modules.ConvASREncoder
51     params:
52       feat_in: *n_mels
53       activation: relu
54       conv_mask: true
55
56     jasper:
57       - filters: 128
58         repeat: 1
59         kernel: [11]
60         stride: [1]
61         dilation: [1]
62         dropout: *dropout
63         residual: true
64         separable: *separable
65         se: true
66         se_context_size: -1
67
68       - filters: 256
69         repeat: *repeat
```

```

70         kernel: [13]
71         stride: [1]
72         dilation: [1]
73         dropout: *dropout
74         residual: true
75         separable: *separable
76         se: true
77         se_context_size: -1
78
79     - filters: 256
80       repeat: *repeat
81       kernel: [15]
82       stride: [1]
83       dilation: [1]
84       dropout: *dropout
85       residual: true
86       separable: *separable
87       se: true
88       se_context_size: -1
89
90     - filters: 256
91       repeat: *repeat
92       kernel: [17]
93       stride: [1]
94       dilation: [1]
95       dropout: *dropout
96       residual: true
97       separable: *separable
98       se: true
99       se_context_size: -1
100
101     - filters: 256
102       repeat: *repeat
103       kernel: [19]
104       stride: [1]
105       dilation: [1]
106       dropout: *dropout
107       residual: true
108       separable: *separable
109       se: true
110       se_context_size: -1
111
112     - filters: 256
113       repeat: *repeat
114       kernel: [21]
115       stride: [1]
116       dilation: [1]
117       dropout: *dropout
118       residual: true
119       separable: *separable
120       se: true
121       se_context_size: -1
122
123     - filters: 256
124       repeat: *repeat
125       kernel: [23]
126       stride: [1]
127       dilation: [1]
128       dropout: *dropout
129       residual: true
130       separable: *separable
131       se: true
132       se_context_size: -1
133
134     - filters: 256
135       repeat: 1
136       kernel: [25]
137       stride: [1]
138       dilation: [1]

```

```

139         dropout: 0.0
140         residual: false
141         separable: *separable
142         se: true
143         se_context_size: -1
144
145     - filters: &enc_feat_out 1024
146       repeat: 1
147       kernel: [1]
148       stride: [1]
149       dilation: [1]
150       dropout: 0.0
151       residual: false
152       separable: *separable
153       se: true
154       se_context_size: -1
155
156     decoder:
157       cls: nemo.collections.asr.modules.ConvASRDecoder
158       params:
159         feat_in: 1024
160         num_classes: 28
161         vocabulary: *labels
162
163     optim:
164       name: novograd
165       # cls: nemo.core.optim.optimisers.Novograd
166       lr: .01
167       # optimiser arguments
168       betas: [0.8, 0.5]
169       weight_decay: 0.001
170
171       # scheduler setup
172       sched:
173         name: CosineAnnealing
174
175       # pytorch lightning args
176       monitor: val_loss
177       reduce_on_plateau: false
178
179       # Scheduler params
180       warmup_steps: null
181       warmup_ratio: null
182       min_lr: 0.0
183       last_epoch: -1
184
185     trainer:
186       gpus: 0 # number of gpus
187       max_epochs: 5
188       max_steps: null # computed at runtime if not set
189       num_nodes: 1
190       accelerator: ddp
191       accumulate_grad_batches: 1
192       checkpoint_callback: False # Provided by exp manager
193       logger: False # Provided by exp_manager
194       log_every_n_steps: 1 # Interval of logging.
195       val_check_interval: 1.0 # check once per epoch .25 for 4 times per epoch
196
197     exp_manager:
198       exp_dir: null
199       name: *name
200       create_tensorboard_logger: True
201       create_checkpoint_callback: True
202
203     hydra:
204       run:
205         dir: .
206       job_logging:
207         root:
208
209     handlers: null

```

APPENDIX C: YAML file of the model architecture specified during the conduct of the Block optimization experiment.

---

```

1  name: $name "B20R1"
2  sample_rate: $sample_rate 16000
3  repeat: $repeat 1
4  dropout: $dropout 0.0
5  separable: $separable true
6  labels: $labels [" ", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
7              "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", ""]
8
9  model:
10   train_ds:
11     manifest_filepath: ???
12     sample_rate: 16000
13     labels: *labels
14     batch_size: 5
15     trim_silence: True
16     max_duration: 16.7
17     shuffle: True
18     is_tarred: False
19     tarred_audio_filepaths: null
20     tarred_shard_strategy: "scatter"
21
22   validation_ds:
23     manifest_filepath: ???
24     sample_rate: 16000
25     labels: *labels
26     batch_size: 5
27     shuffle: False
28
29   preprocessor:
30     _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
31     normalise: "per_feature"
32     window_size: 0.02
33     sample_rate: *sample_rate
34     window_stride: 0.01
35     window: "hann"
36     features: $n_mels 64
37     n_fft: 512
38     frame_splicing: 1
39     dither: 0.00001
40     stft_conv: false
41
42   $ spec_augment:
43     $ target : nemo.collections.asr.modules.SpectrogramAugmentation
44     $ rect freq: 50
45     $ rect masks: 5
46     $ rect time: 120
47
48   encoder:
49     target : nemo.collections.asr.modules.ConvASREncoder
50     feat_in: *n_mels
51     activation: relu
52     conv_mask: true
53
54   jasper:
55     - filters: 128
56       repeat: 1
57       kernel: [11]
58       stride: [1]
59       dilation: [1]
60       dropout: *dropout
61       residual: true
62       separable: *separable
63       se: true
64       se_context_size: -1
65
66     - filters: 256
67       repeat: *repeat
68       kernel: [13]
69       stride: [1]

```

```

70         dilation: [1]
71         dropout: *dropout
72         residual: true
73         separable: *separable
74         se: true
75         se_context_size: -1
76
77     - filters: 256
78       repeat: *repeat
79       kernel: [15]
80       stride: [1]
81       dilation: [1]
82       dropout: *dropout
83       residual: true
84       separable: *separable
85       se: true
86       se_context_size: -1
87
88     - filters: 256
89       repeat: *repeat
90       kernel: [17]
91       stride: [1]
92       dilation: [1]
93       dropout: *dropout
94       residual: true
95       separable: *separable
96       se: true
97       se_context_size: -1
98
99     - filters: 256
100      repeat: *repeat
101      kernel: [19]
102      stride: [1]
103      dilation: [1]
104      dropout: *dropout
105      residual: true
106      separable: *separable
107      se: true
108      se_context_size: -1
109
110     - filters: 256
111       repeat: *repeat
112       kernel: [21]
113       stride: [1]
114       dilation: [1]
115       dropout: *dropout
116       residual: true
117       separable: *separable
118       se: true
119       se_context_size: -1
120
121     - filters: 256
122       repeat: *repeat
123       kernel: [23]
124       stride: [1]
125       dilation: [1]
126       dropout: *dropout
127       residual: true
128       separable: *separable
129       se: true
130       se_context_size: -1
131
132     - filters: 256
133       repeat: *repeat
134       kernel: [25]
135       stride: [1]
136       dilation: [1]
137       dropout: *dropout
138       residual: true

```

```

139         separable: *separable
140         se: true
141         se context size: -1
142
143     - filters: 256
144       repeat: *repeat
145       kernel: [27]
146       stride: [1]
147       dilation: [1]
148       dropout: *dropout
149       residual: true
150       separable: *separable
151       se: true
152       se_context_size: -1
153
154     - filters: 256
155       repeat: *repeat
156       kernel: [29]
157       stride: [1]
158       dilation: [1]
159       dropout: *dropout
160       residual: true
161       separable: *separable
162       se: true
163       se_context_size: -1
164
165     - filters: 256
166       repeat: *repeat
167       kernel: [31]
168       stride: [1]
169       dilation: [1]
170       dropout: *dropout
171       residual: true
172       separable: *separable
173       se: true
174       se_context_size: -1
175
176     - filters: 256
177       repeat: *repeat
178       kernel: [33]
179       stride: [1]
180       dilation: [1]
181       dropout: *dropout
182       residual: true
183       separable: *separable
184       se: true
185       se_context_size: -1
186
187     - filters: 256
188       repeat: *repeat
189       kernel: [35]
190       stride: [1]
191       dilation: [1]
192       dropout: *dropout
193       residual: true
194       separable: *separable
195       se: true
196       se_context_size: -1
197
198     - filters: 256
199       repeat: *repeat
200       kernel: [37]
201       stride: [1]
202       dilation: [1]
203       dropout: *dropout
204       residual: true
205       separable: *separable
206       se: true
207       se_context_size: -1

```

```

208
209 - filters: 256
210   repeat: *repeat
211   kernel: [39]
212   stride: [1]
213   dilation: [1]
214   dropout: *dropout
215   residual: true
216   separable: *separable
217   se: true
218   se_context_size: -1
219
220 - filters: 256
221   repeat: *repeat
222   kernel: [41]
223   stride: [1]
224   dilation: [1]
225   dropout: *dropout
226   residual: true
227   separable: *separable
228   se: true
229   se_context_size: -1
230
231 - filters: 256
232   repeat: *repeat
233   kernel: [43]
234   stride: [1]
235   dilation: [1]
236   dropout: *dropout
237   residual: true
238   separable: *separable
239   se: true
240   se_context_size: -1
241
242 - filters: 256
243   repeat: *repeat
244   kernel: [45]
245   stride: [1]
246   dilation: [1]
247   dropout: *dropout
248   residual: true
249   separable: *separable
250   se: true
251   se_context_size: -1
252
253 - filters: 256
254   repeat: *repeat
255   kernel: [47]
256   stride: [1]
257   dilation: [1]
258   dropout: *dropout
259   residual: true
260   separable: *separable
261   se: true
262   se_context_size: -1
263
264 - filters: 256
265   repeat: *repeat
266   kernel: [49]
267   stride: [1]
268   dilation: [1]
269   dropout: *dropout
270   residual: true
271   separable: *separable
272   se: true
273   se_context_size: -1
274
275 - filters: 256
276   repeat: *repeat

```

```

277         kernel: [51]
278         stride: [1]
279         dilation: [1]
280         dropout: *dropout
281         residual: true
282         separable: *separable
283         se: true
284         se_context_size: -1
285
286     - filters: 256
287       repeat: 1
288       kernel: [53]
289       stride: [1]
290       dilation: [1]
291       dropout: 0.0
292       residual: false
293       separable: *separable
294       se: true
295       se_context_size: -1
296
297     - filters: &enc feat out 1024
298       repeat: 1
299       kernel: [1]
300       stride: [1]
301       dilation: [1]
302       dropout: 0.0
303       residual: false
304       separable: *separable
305       se: true
306       se_context_size: -1
307
308   decoder:
309     _target_: nemo.collections.asr.modules.ConvASRDecoder
310     feat_in: 1024
311     num_classes: 28
312     vocabulary: *labels
313
314   optim:
315     name: novograd
316     # _target_: nemo.core.optim.optimizers.Novograd
317     lr: .01
318     # optimiser arguments
319     betas: [0.8, 0.5]
320     weight_decay: 0.001
321
322     # scheduler setup
323     sched:
324       name: CosineAnnealing
325
326     # pytorch lightning args
327     monitor: val loss
328     reduce_on_plateau: false
329
330     # Scheduler params
331     warmup_steps: null
332     warmup_ratio: null
333     min_lr: 0.0
334     last_epoch: -1
335
336   trainer:
337     gpus: 0 # number of gpus
338     max_epochs: 5
339     max_steps: null # computed at runtime if not set
340     num_nodes: 1
341     accelerator: ddp
342     accumulate_grad_batches: 1
343     checkpoint_callback: False # Provided by exp_manager
344     logger: False # Provided by exp manager
345     log_every_n_steps: 1 # Interval of logging.
346
347
348   val check interval: 1.0 # check once per epoch .25 for 4 times per epoch
349
350   exp manager:
351     exp_dir: null
352     name: *name
353     create_tensorboard_logger: True
354     create_checkpoint_callback: True
355
356   hydra:
357     run:
358       dir: .
359     job_logging:
360       root:
361         handlers: null

```

APPENDIX D: YAML file of the model architecture specified during the conduct of the Sub-block optimization experiment.

---

```
1  name: $name "B1R17"
2  sample_rate: $sample_rate 16000
3  repeat: $repeat 17
4  dropout: $dropout 0.0
5  separable: $separable true
6  labels: $labels [" ", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
7              "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", ""]
8
9  model:
10   train_ds:
11     manifest_filepath: ???
12     sample_rate: 16000
13     labels: *labels
14     batch_size: 5
15     trim_silence: True
16     max_duration: 16.7
17     shuffle: True
18     is_tarred: False
19     tarred_audio_filepaths: null
20     tarred_shard_strategy: "scatter"
21
22   validation_ds:
23     manifest_filepath: ???
24     sample_rate: 16000
25     labels: *labels
26     batch_size: 5
27     shuffle: False
28
29   preprocessor:
30     _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
31     normalise: "per feature"
32     window_size: 0.02
33     sample_rate: *sample_rate
34     window_stride: 0.01
35     window: "hann"
36     features: $n_mels 64
37     n_fft: 512
38     frame_splicing: 1
39     dither: 0.00001
40     stft_conv: false
41
42   $ spec_augment:
43     $ target: nemo.collections.asr.modules.SpectrogramAugmentation
44     $ rect_freq: 50
45     $ rect_masks: 5
46     $ rect_time: 120
47
48   encoder:
49     _target_: nemo.collections.asr.modules.ConvASREncoder
50     feat_in: *n_mels
51     activation: relu
52     conv_mask: true
53
54   jasper:
55     - filters: 128
56       repeat: 1
57       kernel: [11]
58       stride: [1]
59       dilation: [1]
60       dropout: *dropout
61       residual: true
62       separable: *separable
63       se: true
64       se_context_size: -1
65
66     - filters: 256
67       repeat: *repeat
68       kernel: [13]
69       stride: [1]
```

```

70         dilation: [1]
71         dropout: *dropout
72         residual: true
73         separable: *separable
74         se: true
75         se_context_size: -1
76
77
78     - filters: 256
79       repeat: 1
80       kernel: [21]
81       stride: [1]
82       dilation: [1]
83       dropout: 0.0
84       residual: false
85       separable: *separable
86       se: true
87       se_context_size: -1
88
89     - filters: 6enc feat out 1024
90       repeat: 1
91       kernel: [1]
92       stride: [1]
93       dilation: [1]
94       dropout: 0.0
95       residual: false
96       separable: *separable
97       se: true
98       se_context_size: -1
99
100 decoder:
101   _target_: nemo.collections.asr.modules.ConvASRDecoder
102   feat_in: 1024
103   num_classes: 28
104   vocabulary: *labels
105
106 optim:
107   name: novograd
108   # _target_: nemo.core.optim.optimizers.Novograd
109   lr: .01
110   # optimiser arguments
111   betas: [0.8, 0.5]
112   weight_decay: 0.001
113
114   # scheduler setup
115   sched:
116     name: CosineAnnealing
117
118   # pytorch lightning args
119   monitor: val loss
120   reduce_on_plateau: false
121
122   # Scheduler params
123   warmup_steps: null
124   warmup_ratio: null
125   min_lr: 0.0
126   last_epoch: -1
127
128 trainer:
129   gpus: 0 # number of gpus
130   max_epochs: 5
131   max_steps: null # computed at runtime if not set
132   num_nodes: 1
133   accelerator: ddp
134   accumulate_grad_batches: 1
135   checkpoint_callback: False # Provided by exp_manager
136   logger: False # Provided by exp_manager
137   log_every_n_steps: 1 # Interval of logging.
138   val_check_interval: 1.0 # check once per epoch .25 for 4 times per epoch
139
140 exp manager:
141   exp_dir: null
142   name: *name
143   create_tensorboard_logger: True
144   create_checkpoint_callback: True
145
146 hydra:
147   run:
148     dir: .
149   job_logging:
150     root:
151       handlers: null
152

```

APPENDIX E: YAML file of the model architecture specified for the Quartznet 1×1 model during the conduct of the Epoch optimization experiment.

---

```

1  name: $name "B1R1"
2  sample_rate: $sample_rate 16000
3  repeat: $repeat 1
4  dropout: $dropout 0.0
5  separable: $separable true
6  labels: $labels [" ", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
7              "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", ""]
8
9  model:
10   train_ds:
11     manifest_filepath: ???
12     sample_rate: 16000
13     labels: *labels
14     batch_size: 5
15     trim_silence: True
16     max_duration: 16.7
17     shuffle: True
18     is_tarred: False
19     tarred_audio_filepaths: null
20     tarred_shard_strategy: "scatter"
21
22   validation_ds:
23     manifest_filepath: ???
24     sample_rate: 16000
25     labels: *labels
26     batch_size: 5
27     shuffle: False
28
29   preprocessor:
30     _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
31     normalise: "per_feature"
32     window_size: 0.02
33     sample_rate: *sample_rate
34     window_stride: 0.01
35     window: "hann"
36     features: $n_mels 64
37     n_fft: 512
38     frame_splicing: 1
39     dither: 0.00001
40     stft_conv: false
41
42   # spec_augment:
43   #   _target_: nemo.collections.asr.modules.SpectrogramAugmentation
44   #   rect_freq: 50
45   #   rect_masks: 5
46   #   rect_time: 120
47
48   encoder:
49     _target_: nemo.collections.asr.modules.ConvASREncoder
50     feat_in: *n_mels
51     activation: relu
52     conv_mask: true
53
54   jasper:
55     - filters: 128
56       repeat: 1
57       kernel: [11]
58       stride: [1]
59       dilation: [1]
60       dropout: *dropout
61       residual: true
62       separable: *separable
63       se: true
64       se_context_size: -1
65
66     - filters: 256
67       repeat: *repeat
68       kernel: [13]
69       stride: [1]

```

```

70         dilation: [1]
71         dropout: *dropout
72         residual: true
73         separable: *separable
74         se: true
75         se_context_size: -1
76
77
78     - filters: 256
79       repeat: 1
80       kernel: [21]
81       stride: [1]
82       dilation: [1]
83       dropout: 0.0
84       residual: false
85       separable: *separable
86       se: true
87       se_context_size: -1
88
89     - filters: 6enc feat out 1024
90       repeat: 1
91       kernel: [1]
92       stride: [1]
93       dilation: [1]
94       dropout: 0.0
95       residual: false
96       separable: *separable
97       se: true
98       se_context_size: -1
99
100 decoder:
101   _target_: nemo.collections.asr.modules.ConvASRDecoder
102   feat_in: 1024
103   num_classes: 28
104   vocabulary: *labels
105
106 optim:
107   name: novograd
108   #_target_: nemo.core.optim.optimisers.Novograd
109   lr: .01
110   # optimiser arguments
111   betas: [0.8, 0.5]
112   weight_decay: 0.001
113
114   # scheduler setup
115   sched:
116     name: CosineAnnealing
117
118   # pytorch lightning args
119   monitor: val_loss
120   reduce_on_plateau: false
121
122   # Scheduler params
123   warmup_steps: null
124   warmup_ratio: null
125   min_lr: 0.0
126   last_epoch: -1
127
128 trainer:
129   gpus: 0 # number of gpus
130   max_epochs: 5
131   max_steps: null # computed at runtime if not set
132   num_nodes: 1
133   accelerator: ddp
134   accumulate_grad_batches: 1
135   checkpoint_callback: False # Provided by exp_manager
136   logger: False # Provided by exp_manager
137   log_every_n_steps: 1 # Interval of logging.
138   val_check_interval: 1.0 # check once per epoch .25 for 4 times per epoch
139
140
141 exp manager:
142   exp_dir: null
143   name: *name
144   create_tensorboard_logger: True
145   create_checkpoint_callback: True
146
147 hydra:
148   run:
149     dir: .
150   job_logging:
151     root:
152       handlers: null

```

APPENDIX F: YAML file of the model architecture specified for the Quartznet 20×1 model during the conduct of the Epoch optimization experiment.

---

```

1  name: &name "B20R1"
2  sample_rate: &sample_rate 16000
3  repeat: &repeat 1
4  dropout: &dropout 0.0
5  separable: &separable true
6  labels: &labels [" ", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
7              "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", ""]
8
9  model:
10   train_ds:
11     manifest_filepath: ???
12     sample_rate: 16000
13     labels: *labels
14     batch_size: 5
15     trim_silence: True
16     max_duration: 16.7
17     shuffle: True
18     is_tarred: False
19     tarred_audio_filepaths: null
20     tarred_shard_strategy: "scatter"
21
22   validation_ds:
23     manifest_filepath: ???
24     sample_rate: 16000
25     labels: *labels
26     batch_size: 5
27     shuffle: False
28
29   preprocessor:
30     _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
31     normalise: "per_feature"
32     window_size: 0.02
33     sample_rate: *sample_rate
34     window_stride: 0.01
35     window: "hann"
36     features: &n_mels 64
37     n_fft: 512
38     frame_splicing: 1
39     dither: 0.00001
40     stft_conv: false
41
42   # spec_augment:
43   #   _target_: nemo.collections.asr.modules.SpectrogramAugmentation
44   #   rect_freq: 50
45   #   rect_masks: 5
46   #   rect_time: 120
47
48   encoder:
49     _target_: nemo.collections.asr.modules.ConvASREncoder
50     feat_in: *n_mels
51     activation: relu
52     conv_mask: true
53
54   jasper:
55     - filters: 128
56       repeat: 1
57       kernel: [11]
58       stride: [1]
59       dilation: [1]
60       dropout: *dropout
61       residual: true
62       separable: *separable
63       se: true
64       se_context_size: -1
65
66     - filters: 256
67       repeat: *repeat
68       kernel: [13]
69       stride: [1]

```

```

70         dilation: [1]
71         dropout: *dropout
72         residual: true
73         separable: *separable
74         se: true
75         se_context_size: -1
76
77     - filters: 256
78       repeat: *repeat
79       kernel: [15]
80       stride: [1]
81       dilation: [1]
82       dropout: *dropout
83       residual: true
84       separable: *separable
85       se: true
86       se_context_size: -1
87
88     - filters: 256
89       repeat: *repeat
90       kernel: [17]
91       stride: [1]
92       dilation: [1]
93       dropout: *dropout
94       residual: true
95       separable: *separable
96       se: true
97       se_context_size: -1
98
99     - filters: 256
100      repeat: *repeat
101      kernel: [19]
102      stride: [1]
103      dilation: [1]
104      dropout: *dropout
105      residual: true
106      separable: *separable
107      se: true
108      se_context_size: -1
109
110     - filters: 256
111       repeat: *repeat
112       kernel: [21]
113       stride: [1]
114       dilation: [1]
115       dropout: *dropout
116       residual: true
117       separable: *separable
118       se: true
119       se_context_size: -1
120
121     - filters: 256
122       repeat: *repeat
123       kernel: [23]
124       stride: [1]
125       dilation: [1]
126       dropout: *dropout
127       residual: true
128       separable: *separable
129       se: true
130       se_context_size: -1
131
132     - filters: 256
133       repeat: *repeat
134       kernel: [25]
135       stride: [1]
136       dilation: [1]
137       dropout: *dropout
138       residual: true

```

```

139         separable: *separable
140         se: true
141         se context size: -1
142
143     - filters: 256
144       repeat: *repeat
145       kernel: [27]
146       stride: [1]
147       dilation: [1]
148       dropout: *dropout
149       residual: true
150       separable: *separable
151       se: true
152       se_context_size: -1
153
154     - filters: 256
155       repeat: *repeat
156       kernel: [29]
157       stride: [1]
158       dilation: [1]
159       dropout: *dropout
160       residual: true
161       separable: *separable
162       se: true
163       se_context_size: -1
164
165     - filters: 256
166       repeat: *repeat
167       kernel: [31]
168       stride: [1]
169       dilation: [1]
170       dropout: *dropout
171       residual: true
172       separable: *separable
173       se: true
174       se_context_size: -1
175
176     - filters: 256
177       repeat: *repeat
178       kernel: [33]
179       stride: [1]
180       dilation: [1]
181       dropout: *dropout
182       residual: true
183       separable: *separable
184       se: true
185       se_context_size: -1
186
187     - filters: 256
188       repeat: *repeat
189       kernel: [35]
190       stride: [1]
191       dilation: [1]
192       dropout: *dropout
193       residual: true
194       separable: *separable
195       se: true
196       se_context_size: -1
197
198     - filters: 256
199       repeat: *repeat
200       kernel: [37]
201       stride: [1]
202       dilation: [1]
203       dropout: *dropout
204       residual: true
205       separable: *separable
206       se: true
207       se_context_size: -1

```

```

208
209 - filters: 256
210   repeat: *repeat
211   kernel: [39]
212   stride: [1]
213   dilation: [1]
214   dropout: *dropout
215   residual: true
216   separable: *separable
217   se: true
218   se_context_size: -1
219
220 - filters: 256
221   repeat: *repeat
222   kernel: [41]
223   stride: [1]
224   dilation: [1]
225   dropout: *dropout
226   residual: true
227   separable: *separable
228   se: true
229   se_context_size: -1
230
231 - filters: 256
232   repeat: *repeat
233   kernel: [43]
234   stride: [1]
235   dilation: [1]
236   dropout: *dropout
237   residual: true
238   separable: *separable
239   se: true
240   se_context_size: -1
241
242 - filters: 256
243   repeat: *repeat
244   kernel: [45]
245   stride: [1]
246   dilation: [1]
247   dropout: *dropout
248   residual: true
249   separable: *separable
250   se: true
251   se_context_size: -1
252
253 - filters: 256
254   repeat: *repeat
255   kernel: [47]
256   stride: [1]
257   dilation: [1]
258   dropout: *dropout
259   residual: true
260   separable: *separable
261   se: true
262   se_context_size: -1
263
264 - filters: 256
265   repeat: *repeat
266   kernel: [49]
267   stride: [1]
268   dilation: [1]
269   dropout: *dropout
270   residual: true
271   separable: *separable
272   se: true
273   se_context_size: -1
274
275 - filters: 256
276   repeat: *repeat

```

```

277         kernel: [51]
278         stride: [1]
279         dilation: [1]
280         dropout: *dropout
281         residual: true
282         separable: *separable
283         se: true
284         se_context_size: -1
285
286     - filters: 256
287       repeat: 1
288       kernel: [53]
289       stride: [1]
290       dilation: [1]
291       dropout: 0.0
292       residual: false
293       separable: *separable
294       se: true
295       se_context_size: -1
296
297     - filters: &enc feat out 1024
298       repeat: 1
299       kernel: [1]
300       stride: [1]
301       dilation: [1]
302       dropout: 0.0
303       residual: false
304       separable: *separable
305       se: true
306       se_context_size: -1
307
308   decoder:
309     _target_: nemo.collections.asr.modules.ConvASRDecoder
310     feat_in: 1024
311     num_classes: 28
312     vocabulary: *labels
313
314   optim:
315     name: novograd
316     # _target_: nemo.core.optim.optimizers.Novograd
317     lr: .01
318     # optimiser arguments
319     betas: [0.8, 0.5]
320     weight_decay: 0.001
321
322     # scheduler setup
323   sched:
324     name: CosineAnnealing
325
326     # pytorch lightning args
327     monitor: val loss
328     reduce_on_plateau: false
329
330     # Scheduler params
331     warmup_steps: null
332     warmup_ratio: null
333     min_lr: 0.0
334     last_epoch: -1
335
336   trainer:
337     gpus: 0 # number of gpus
338     max_epochs: 5
339     max_steps: null # computed at runtime if not set
340     num_nodes: 1
341     accelerator: ddp
342     accumulate_grad_batches: 1
343     checkpoint_callback: False # Provided by exp_manager
344     logger: False # Provided by exp manager
345     log_every_n_steps: 1 # Interval of logging.
346
347   val check interval: 1.0 # check once per epoch .25 for 4 times per epoch
348
349   exp manager:
350     exp dir: null
351     name: *name
352     create_tensorboard_logger: True
353     create_checkpoint_callback: True
354
355   hydra:
356     run:
357       dir: .
358     job_logging:
359       root:
360         handlers: null

```

APPENDIX G: YAML file of the model architecture specified during the conduct of the Rectangle Masking Spectrogram Augmentation experiment.

---

```

1  name: $name "B20R1-rectmasked"
2  sample_rate: $sample_rate 16000
3  repeat: $repeat 1
4  dropout: $dropout 0.0
5  separable: $separable true
6  labels: $labels [" ", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
7      "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", "'"]
8
9  model:
10   train_ds:
11     manifest_filepath: ???
12     sample_rate: 16000
13     labels: *labels
14     batch_size: 5
15     trim_silence: True
16     max_duration: 16.7
17     shuffle: True
18     is_tarred: False
19     tarred_audio_filepaths: null
20     tarred_shard_strategy: "scatter"
21
22   validation_ds:
23     manifest_filepath: ???
24     sample_rate: 16000
25     labels: *labels
26     batch_size: 5
27     shuffle: False
28
29   preprocessor:
30     _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
31     normalise: "per_feature"
32     window_size: 0.02
33     sample_rate: *sample_rate
34     window_stride: 0.01
35     window: "hann"
36     features: $n_mels 64
37     n_fft: 512
38     frame_splicing: 1
39     dither: 0.00001
40     stft_conv: false
41
42   spec_augment:
43     _target_: nemo.collections.asr.modules.SpectrogramAugmentation
44     rect_freq: 50
45     rect_masks: 5
46     rect_time: 120
47
48   encoder:
49     _target_: nemo.collections.asr.modules.ConvASREncoder
50     feat_in: *n_mels
51     activation: relu
52     conv_mask: true
53
54   jasper:
55     - filters: 128
56       repeat: 1
57       kernel: [11]
58       stride: [1]
59       dilation: [1]
60       dropout: *dropout
61       residual: true
62       separable: *separable
63       se: true
64       se_context_size: -1
65
66     - filters: 256
67       repeat: *repeat
68       kernel: [13]
69       stride: [1]

```

```

70     dilation: [1]
71     dropout: *dropout
72     residual: true
73     separable: *separable
74     se: true
75     se_context_size: -1
76
77 - filters: 256
78   repeat: *repeat
79   kernel: [15]
80   stride: [1]
81   dilation: [1]
82   dropout: *dropout
83   residual: true
84   separable: *separable
85   se: true
86   se_context_size: -1
87
88 - filters: 256
89   repeat: *repeat
90   kernel: [17]
91   stride: [1]
92   dilation: [1]
93   dropout: *dropout
94   residual: true
95   separable: *separable
96   se: true
97   se_context_size: -1
98
99 - filters: 256
100   repeat: *repeat
101   kernel: [19]
102   stride: [1]
103   dilation: [1]
104   dropout: *dropout
105   residual: true
106   separable: *separable
107   se: true
108   se_context_size: -1
109
110 - filters: 256
111   repeat: *repeat
112   kernel: [21]
113   stride: [1]
114   dilation: [1]
115   dropout: *dropout
116   residual: true
117   separable: *separable
118   se: true
119   se_context_size: -1
120
121 - filters: 256
122   repeat: *repeat
123   kernel: [23]
124   stride: [1]
125   dilation: [1]
126   dropout: *dropout
127   residual: true
128   separable: *separable
129   se: true
130   se_context_size: -1
131
132 - filters: 256
133   repeat: *repeat
134   kernel: [25]
135   stride: [1]
136   dilation: [1]
137   dropout: *dropout
138   residual: true

```

```

139         separable: *separable
140         se: true
141         se context size: -1
142
143     - filters: 256
144       repeat: *repeat
145       kernel: [27]
146       stride: [1]
147       dilation: [1]
148       dropout: *dropout
149       residual: true
150       separable: *separable
151       se: true
152       se_context_size: -1
153
154     - filters: 256
155       repeat: *repeat
156       kernel: [29]
157       stride: [1]
158       dilation: [1]
159       dropout: *dropout
160       residual: true
161       separable: *separable
162       se: true
163       se_context_size: -1
164
165     - filters: 256
166       repeat: *repeat
167       kernel: [31]
168       stride: [1]
169       dilation: [1]
170       dropout: *dropout
171       residual: true
172       separable: *separable
173       se: true
174       se_context_size: -1
175
176     - filters: 256
177       repeat: *repeat
178       kernel: [33]
179       stride: [1]
180       dilation: [1]
181       dropout: *dropout
182       residual: true
183       separable: *separable
184       se: true
185       se_context_size: -1
186
187     - filters: 256
188       repeat: *repeat
189       kernel: [35]
190       stride: [1]
191       dilation: [1]
192       dropout: *dropout
193       residual: true
194       separable: *separable
195       se: true
196       se_context_size: -1
197
198     - filters: 256
199       repeat: *repeat
200       kernel: [37]
201       stride: [1]
202       dilation: [1]
203       dropout: *dropout
204       residual: true
205       separable: *separable
206       se: true
207       se_context_size: -1

```

```

208
209 - filters: 256
210   repeat: *repeat
211   kernel: [39]
212   stride: [1]
213   dilation: [1]
214   dropout: *dropout
215   residual: true
216   separable: *separable
217   se: true
218   se_context_size: -1
219
220 - filters: 256
221   repeat: *repeat
222   kernel: [41]
223   stride: [1]
224   dilation: [1]
225   dropout: *dropout
226   residual: true
227   separable: *separable
228   se: true
229   se_context_size: -1
230
231 - filters: 256
232   repeat: *repeat
233   kernel: [43]
234   stride: [1]
235   dilation: [1]
236   dropout: *dropout
237   residual: true
238   separable: *separable
239   se: true
240   se_context_size: -1
241
242 - filters: 256
243   repeat: *repeat
244   kernel: [45]
245   stride: [1]
246   dilation: [1]
247   dropout: *dropout
248   residual: true
249   separable: *separable
250   se: true
251   se_context_size: -1
252
253 - filters: 256
254   repeat: *repeat
255   kernel: [47]
256   stride: [1]
257   dilation: [1]
258   dropout: *dropout
259   residual: true
260   separable: *separable
261   se: true
262   se_context_size: -1
263
264 - filters: 256
265   repeat: *repeat
266   kernel: [49]
267   stride: [1]
268   dilation: [1]
269   dropout: *dropout
270   residual: true
271   separable: *separable
272   se: true
273   se_context_size: -1
274
275 - filters: 256
276   repeat: *repeat

```

```

277         kernel: [51]
278         stride: [1]
279         dilation: [1]
280         dropout: *dropout
281         residual: true
282         separable: *separable
283         se: true
284         se_context_size: -1
285
286     - filters: 256
287       repeat: 1
288       kernel: [53]
289       stride: [1]
290       dilation: [1]
291       dropout: 0.0
292       residual: false
293       separable: *separable
294       se: true
295       se_context_size: -1
296
297     - filters: &enc feat out 1024
298       repeat: 1
299       kernel: [1]
300       stride: [1]
301       dilation: [1]
302       dropout: 0.0
303       residual: false
304       separable: *separable
305       se: true
306       se_context_size: -1
307
308     decoder:
309       _target_: nemo.collections.asr.modules.ConvASRDecoder
310       feat_in: 1024
311       num_classes: 28
312       vocabulary: *labels
313
314     optim:
315       name: novograd
316       # _target_: nemo.core.optim.optimizers.Novograd
317       lr: .01
318       # optimiser arguments
319       betas: [0.8, 0.5]
320       weight_decay: 0.001
321
322       # scheduler setup
323     sched:
324       name: CosineAnnealing
325
326       # pytorch lightning args
327       monitor: val_loss
328       reduce_on_plateau: false
329
330       # Scheduler params
331       warmup_steps: null
332       warmup_ratio: null
333       min_lr: 0.0
334       last_epoch: -1
335
336     trainer:
337       gpus: 0 # number of gpus
338       max_epochs: 5
339       max_steps: null # computed at runtime if not set
340       num_nodes: 1
341       accelerator: ddp
342       accumulate_grad_batches: 1
343       checkpoint_callback: False # Provided by exp_manager
344       logger: False # Provided by exp manager
345       log_every_n_steps: 1 # Interval of logging.
346
347
348     val check interval: 1.0 # check once per epoch .25 for 4 times per epoch
349
350     exp_manager:
351       exp_dir: null
352       name: *name
353       create_tensorboard_logger: True
354       create_checkpoint_callback: True
355
356     hydra:
357       run:
358         dir: .
359       job_logging:
360         root:
361           handlers: null

```

APPENDIX H: YAML file of the model architecture specified during the conduct of the Time-Frequency Masking Spectrogram Augmentation experiment.

---

```
1  name: &name "B20R1-timefreqmasked"
2  sample_rate: &sample_rate 16000
3  repeat: &repeat 1
4  dropout: &dropout 0.0
5  separable: &separable true
6  labels: &labels [" ", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m",
7              "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", "'"]
8
9  model:
10   train_ds:
11     manifest_filepath: ???
12     sample_rate: 16000
13     labels: *labels
14     batch_size: 5
15     trim_silence: True
16     max_duration: 16.7
17     shuffle: True
18     is_tarred: False
19     tarred_audio_filepaths: null
20     tarred_shard_strategy: "scatter"
21
22   validation_ds:
23     manifest_filepath: ???
24     sample_rate: 16000
25     labels: *labels
26     batch_size: 5
27     shuffle: False
28
29   preprocessor:
30     _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
31     normalise: "per_feature"
32     window_size: 0.02
33     sample_rate: *sample_rate
34     window_stride: 0.01
35     window: "hann"
36     features: &n_mels 64
37     n_fft: 512
38     frame_splicing: 1
39     dither: 0.00001
40     stft_conv: false
41
42   spec_augment:
43     _target_: nemo.collections.asr.modules.SpectrogramAugmentation
44     $ rect_freq: 50
45     $ rect_masks: 5
46     $ rect_time: 120
47     freq_masks: 1
48     freq_width: 5
49     time_masks: 1
50     time_width: 5
51
52   encoder:
53     _target_: nemo.collections.asr.modules.ConvASREncoder
54     feat_in: *n_mels
55     activation: relu
56     conv_mask: true
57
58   jasper:
59     - filters: 128
60       repeat: 1
61       kernel: [11]
62       stride: [1]
63       dilation: [1]
64       dropout: *dropout
65       residual: true
66       separable: *separable
67       se: true
68       se_context_size: -1
69
```

```

70     - filters: 256
71       repeat: *repeat
72       kernel: [13]
73       stride: [1]
74       dilation: [1]
75       dropout: *dropout
76       residual: true
77       separable: *separable
78       se: true
79       se_context_size: -1
80
81     - filters: 256
82       repeat: *repeat
83       kernel: [15]
84       stride: [1]
85       dilation: [1]
86       dropout: *dropout
87       residual: true
88       separable: *separable
89       se: true
90       se_context_size: -1
91
92     - filters: 256
93       repeat: *repeat
94       kernel: [17]
95       stride: [1]
96       dilation: [1]
97       dropout: *dropout
98       residual: true
99       separable: *separable
100      se: true
101      se_context_size: -1
102
103     - filters: 256
104       repeat: *repeat
105       kernel: [19]
106       stride: [1]
107       dilation: [1]
108       dropout: *dropout
109       residual: true
110       separable: *separable
111       se: true
112       se_context_size: -1
113
114     - filters: 256
115       repeat: *repeat
116       kernel: [21]
117       stride: [1]
118       dilation: [1]
119       dropout: *dropout
120       residual: true
121       separable: *separable
122       se: true
123       se_context_size: -1
124
125     - filters: 256
126       repeat: *repeat
127       kernel: [23]
128       stride: [1]
129       dilation: [1]
130       dropout: *dropout
131       residual: true
132       separable: *separable
133       se: true
134       se_context_size: -1
135
136     - filters: 256
137       repeat: *repeat
138       kernel: [25]

```

```

139         stride: [1]
140         dilation: [1]
141         dropout: *dropout
142         residual: true
143         separable: *separable
144         se: true
145         se_context_size: -1
146
147     - filters: 256
148       repeat: *repeat
149       kernel: [27]
150       stride: [1]
151       dilation: [1]
152       dropout: *dropout
153       residual: true
154       separable: *separable
155       se: true
156       se_context_size: -1
157
158     - filters: 256
159       repeat: *repeat
160       kernel: [29]
161       stride: [1]
162       dilation: [1]
163       dropout: *dropout
164       residual: true
165       separable: *separable
166       se: true
167       se_context_size: -1
168
169     - filters: 256
170       repeat: *repeat
171       kernel: [31]
172       stride: [1]
173       dilation: [1]
174       dropout: *dropout
175       residual: true
176       separable: *separable
177       se: true
178       se_context_size: -1
179
180     - filters: 256
181       repeat: *repeat
182       kernel: [33]
183       stride: [1]
184       dilation: [1]
185       dropout: *dropout
186       residual: true
187       separable: *separable
188       se: true
189       se_context_size: -1
190
191     - filters: 256
192       repeat: *repeat
193       kernel: [35]
194       stride: [1]
195       dilation: [1]
196       dropout: *dropout
197       residual: true
198       separable: *separable
199       se: true
200       se_context_size: -1
201
202     - filters: 256
203       repeat: *repeat
204       kernel: [37]
205       stride: [1]
206       dilation: [1]
207       dropout: *dropout

```

```

208         residual: true
209         separable: *separable
210         se: true
211         se_context_size: -1
212
213     - filters: 256
214       repeat: *repeat
215       kernel: [39]
216       stride: [1]
217       dilation: [1]
218       dropout: *dropout
219       residual: true
220       separable: *separable
221       se: true
222       se_context_size: -1
223
224     - filters: 256
225       repeat: *repeat
226       kernel: [41]
227       stride: [1]
228       dilation: [1]
229       dropout: *dropout
230       residual: true
231       separable: *separable
232       se: true
233       se_context_size: -1
234
235     - filters: 256
236       repeat: *repeat
237       kernel: [43]
238       stride: [1]
239       dilation: [1]
240       dropout: *dropout
241       residual: true
242       separable: *separable
243       se: true
244       se_context_size: -1
245
246     - filters: 256
247       repeat: *repeat
248       kernel: [45]
249       stride: [1]
250       dilation: [1]
251       dropout: *dropout
252       residual: true
253       separable: *separable
254       se: true
255       se_context_size: -1
256
257     - filters: 256
258       repeat: *repeat
259       kernel: [47]
260       stride: [1]
261       dilation: [1]
262       dropout: *dropout
263       residual: true
264       separable: *separable
265       se: true
266       se_context_size: -1
267
268     - filters: 256
269       repeat: *repeat
270       kernel: [49]
271       stride: [1]
272       dilation: [1]
273       dropout: *dropout
274       residual: true
275       separable: *separable
276       se: true

```

```

277         se_context_size: -1
278
279     - filters: 256
280       repeat: *repeat
281       kernel: [51]
282       stride: [1]
283       dilation: [1]
284       dropout: *dropout
285       residual: true
286       separable: *separable
287       se: true
288       se_context_size: -1
289
290     - filters: 256
291       repeat: 1
292       kernel: [53]
293       stride: [1]
294       dilation: [1]
295       dropout: 0.0
296       residual: false
297       separable: *separable
298       se: true
299       se_context_size: -1
300
301     - filters: 6enc feat out 1024
302       repeat: 1
303       kernel: [1]
304       stride: [1]
305       dilation: [1]
306       dropout: 0.0
307       residual: false
308       separable: *separable
309       se: true
310       se_context_size: -1
311
312   decoder:
313     _target_: nemo.collections.asr.modules.ConvASRDecoder
314     feat_in: 1024
315     num_classes: 28
316     vocabulary: *labels
317
318   optim:
319     name: novograd
320     # target : nemo.core.optim.optimizers.Novograd
321     lr: .01
322     # optimiser arguments
323     betas: [0.8, 0.5]
324     weight_decay: 0.001
325
326     # scheduler setup
327     sched:
328       name: CosineAnnealing
329
330     # pytorch lightning args
331     monitor: val_loss
332     reduce_on_plateau: false
333
334     # Scheduler params
335     warmup_steps: null
336     warmup_ratio: null
337     min_lr: 0.0
338     last_epoch: -1
339
340   trainer:
341     gpus: 0 # number of gpus
342     max_epochs: 5
343     max_steps: null # computed at runtime if not set
344     num_nodes: 1
345     accelerator: ddp

```

```

346     accumulate_grad_batches: 1
347     checkpoint_callback: False # Provided by exp manager
348     logger: False # Provided by exp manager
349     log_every_n_steps: 1 # Interval of logging.
350     val_check_interval: 1.0 # check once per epoch .25 for 4 times per epoch
351
352     exp_manager:
353         exp_dir: null
354         name: *name
355         create_tensorboard_logger: True
356         create_checkpoint_callback: True
357
358     hydra:
359         run:
360             dir: .
361         job_logging:
362             root:
363                 handlers: null
364

```

APPENDIX I: YAML file of the model architecture specified for the Quartznet 15×5 English model during the conduct of the Basic TL setting experiment.

---

```
1  cfg:
2    train ds:
3      manifest filepath: ./iban/train manifest.json
4      sample rate: 16000
5      labels:
6        - ' '
7        - a
8        - b
9        - c
10       - d
11       - e
12       - f
13       - g
14       - h
15       - i
16       - j
17       - k
18       - l
19       - m
20       - 'n'
21       - o
22       - p
23       - q
24       - r
25       - s
26       - t
27       - u
28       - v
29       - w
30       - x
31       - 'y'
32       - z
33     batch_size: 5
34     trim_silence: true
35     max_duration: 16.7
36     shuffle: true
37     is_tarred: false
38     tarred_audio_filepaths: null
39     tarred_shard_strategy: scatter
40     normalise_transcripts: false
41     num_workers: 8
42     pin_memory: true
43   validation ds:
44     manifest filepath: ./iban/test manifest.json
45     sample rate: 16000
46     labels:
47       - ' '
48       - a
49       - b
50       - c
51       - d
52       - e
53       - f
54       - g
55       - h
56       - i
57       - j
58       - k
59       - l
60       - m
61       - 'n'
62       - o
63       - p
64       - q
65       - r
66       - s
67       - t
68       - u
69       - v
```

```

70     - w
71     - x
72     - 'y'
73     - z
74     batch_size: 5
75     shuffle: false
76     normalise_transcripts: false
77     num_workers: 8
78     pin_memory: true
79     trim_silence: true
80     preprocessor:
81         _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
82         normalise: per_feature
83         window_size: 0.02
84         sample_rate: 16000
85         window_stride: 0.01
86         window: hann
87         features: 64
88         n_fft: 512
89         frame_splicing: 1
90         dither: 1.0e-05
91         stft_conv: false
92     spec_augment:
93         target: nemo.collections.asr.modules.SpectrogramAugmentation
94         rect_freq: 0
95         rect_masks: 0
96         rect_time: 0
97     encoder:
98         target: nemo.collections.asr.modules.ConvASREncoder
99         feat_in: 64
100        activation: relu
101        conv_mask: true
102        jasper:
103            - filters: 128
104              repeat: 1
105              kernel:
106                  - 11
107              stride:
108                  - 1
109              dilation:
110                  - 1
111              dropout: 0.0
112              residual: true
113              separable: true
114              se: true
115              se_context_size: -1
116            - filters: 256
117              repeat: 1
118              kernel:
119                  - 13
120              stride:
121                  - 1
122              dilation:
123                  - 1
124              dropout: 0.0
125              residual: true
126              separable: true
127              se: true
128              se_context_size: -1
129            - filters: 256
130              repeat: 1
131              kernel:
132                  - 15
133              stride:
134                  - 1
135              dilation:
136                  - 1
137              dropout: 0.0
138              residual: true

```

```

139     separable: true
140     se: true
141     se context size: -1
142 - filters: 256
143   repeat: 1
144   kernel:
145     - 17
146   stride:
147     - 1
148   dilation:
149     - 1
150   dropout: 0.0
151   residual: true
152   separable: true
153   se: true
154   se context size: -1
155 - filters: 256
156   repeat: 1
157   kernel:
158     - 19
159   stride:
160     - 1
161   dilation:
162     - 1
163   dropout: 0.0
164   residual: true
165   separable: true
166   se: true
167   se context size: -1
168 - filters: 256
169   repeat: 1
170   kernel:
171     - 21
172   stride:
173     - 1
174   dilation:
175     - 1
176   dropout: 0.0
177   residual: true
178   separable: true
179   se: true
180   se context size: -1
181 - filters: 256
182   repeat: 1
183   kernel:
184     - 23
185   stride:
186     - 1
187   dilation:
188     - 1
189   dropout: 0.0
190   residual: true
191   separable: true
192   se: true
193   se context size: -1
194 - filters: 256
195   repeat: 1
196   kernel:
197     - 25
198   stride:
199     - 1
200   dilation:
201     - 1
202   dropout: 0.0
203   residual: true
204   separable: true
205   se: true
206   se context size: -1
207 - filters: 256

```

```

208         repeat: 1
209         kernel:
210         - 27
211         stride:
212         - 1
213         dilation:
214         - 1
215         dropout: 0.0
216         residual: true
217         separable: true
218         se: true
219         se_context_size: -1
220     - filters: 256
221       repeat: 1
222       kernel:
223       - 29
224       stride:
225       - 1
226       dilation:
227       - 1
228       dropout: 0.0
229       residual: true
230       separable: true
231       se: true
232       se_context_size: -1
233   - filters: 256
234     repeat: 1
235     kernel:
236     - 31
237     stride:
238     - 1
239     dilation:
240     - 1
241     dropout: 0.0
242     residual: true
243     separable: true
244     se: true
245     se_context_size: -1
246   - filters: 256
247     repeat: 1
248     kernel:
249     - 33
250     stride:
251     - 1
252     dilation:
253     - 1
254     dropout: 0.0
255     residual: true
256     separable: true
257     se: true
258     se_context_size: -1
259   - filters: 256
260     repeat: 1
261     kernel:
262     - 35
263     stride:
264     - 1
265     dilation:
266     - 1
267     dropout: 0.0
268     residual: true
269     separable: true
270     se: true
271     se_context_size: -1
272   - filters: 256
273     repeat: 1
274     kernel:
275     - 37
276     stride:

```

```

277         - 1
278         dilation:
279         - 1
280         dropout: 0.0
281         residual: true
282         separable: true
283         se: true
284         se_context_size: -1
285     - filters: 256
286       repeat: 1
287       kernel:
288       - 29
289       stride:
290       - 1
291       dilation:
292       - 1
293       dropout: 0.0
294       residual: true
295       separable: true
296       se: true
297       se_context_size: -1
298   - filters: 256
299     repeat: 1
300     kernel:
301     - 41
302     stride:
303     - 1
304     dilation:
305     - 1
306     dropout: 0.0
307     residual: true
308     separable: true
309     se: true
310     se_context_size: -1
311   - filters: 256
312     repeat: 1
313     kernel:
314     - 43
315     stride:
316     - 1
317     dilation:
318     - 1
319     dropout: 0.0
320     residual: true
321     separable: true
322     se: true
323     se_context_size: -1
324   - filters: 256
325     repeat: 1
326     kernel:
327     - 45
328     stride:
329     - 1
330     dilation:
331     - 1
332     dropout: 0.0
333     residual: true
334     separable: true
335     se: true
336     se_context_size: -1
337   - filters: 256
338     repeat: 1
339     kernel:
340     - 47
341     stride:
342     - 1
343     dilation:
344     - 1
345     dropout: 0.0

```

```

346         residual: true
347         separable: true
348         se: true
349         se_context_size: -1
350     - filters: 256
351       repeat: 1
352       kernel:
353         - 49
354       stride:
355         - 1
356       dilation:
357         - 1
358       dropout: 0.0
359       residual: true
360       separable: true
361       se: true
362       se_context_size: -1
363   - filters: 256
364     repeat: 1
365     kernel:
366       - 51
367     stride:
368       - 1
369     dilation:
370       - 1
371     dropout: 0.0
372     residual: true
373     separable: true
374     se: true
375     se_context_size: -1
376   - filters: 256
377     repeat: 1
378     kernel:
379       - 53
380     stride:
381       - 1
382     dilation:
383       - 1
384     dropout: 0.0
385     residual: false
386     separable: true
387     se: true
388     se_context_size: -1
389   - filters: 1024
390     repeat: 1
391     kernel:
392       - 1
393     stride:
394       - 1
395     dilation:
396       - 1
397     dropout: 0.0
398     residual: false
399     separable: true
400     se: true
401     se_context_size: -1
402 decoder:
403   _target_: nemo.collections.asr.modules.ConvASRDecoder
404   feat_in: 1024
405   num_classes: 28
406   vocabulary:
407     - ' '
408     - a
409     - b
410     - c
411     - d
412     - e
413     - f
414     - g

```

```

415     - h
416     - i
417     - j
418     - k
419     - l
420     - m
421     - 'n'
422     - o
423     - p
424     - q
425     - r
426     - s
427     - t
428     - u
429     - v
430     - w
431     - x
432     - 'y'
433     - z
434     - ''''
435 optim:
436     name: novograd
437     lr: 0.001
438     betas:
439         - 0.8
440         - 0.5
441     weight_decay: 0.001
442     sched:
443         name: CosineAnnealing
444         monitor: val_loss
445         reduce_on_plateau: false
446         warmup_steps: null
447         warmup_ratio: 0.05
448         min_lr: 1.0e-05
449         last_epoch: -1
450 target: nemo.collections.asr.models.ctc_models.EncDecCTCModel
451 nemo_version: 1.9.0rc0
452

```

APPENDIX J: YAML file of the model architecture specified for the Quartznet 15×5 English model during the conduct of the TL with TF-SA (Best Model – with Time-Frequency SA) setting experiment.

---

```
1  cfg:
2    sample_rate: 16000
3    repeat: 5
4    dropout: 0.0
5    separable: true
6    labels:
7      - ' '
8      - a
9      - b
10     - c
11     - d
12     - e
13     - f
14     - g
15     - h
16     - i
17     - j
18     - k
19     - l
20     - m
21     - 'n'
22     - o
23     - p
24     - q
25     - r
26     - s
27     - t
28     - u
29     - v
30     - w
31     - x
32     - 'y'
33     - z
34     - ''
35  train_ds:
36    manifest_filepath: ./iban/train_manifest.json
37    sample_rate: 16000
38    labels:
39      - ' '
40      - a
41      - b
42      - c
43      - d
44      - e
45      - f
46      - g
47      - h
48      - i
49      - j
50      - k
51      - l
52      - m
53      - 'n'
54      - o
55      - p
56      - q
57      - r
58      - s
59      - t
60      - u
61      - v
62      - w
63      - x
64      - 'y'
65      - z
66    batch_size: 5
67    trim_silence: true
68    max_duration: 16.7
69    shuffle: true
```

```

70     is tarred: false
71     tarred audio filepaths: null
72     tarred shard strategy: scatter
73     normalise transcripts: false
74     num_workers: 8
75     pin_memory: true
76 validation ds:
77     manifest_filepath: ./iban/test_manifest.json
78     sample_rate: 16000
79     labels:
80     - ' '
81     - a
82     - b
83     - c
84     - d
85     - e
86     - f
87     - g
88     - h
89     - i
90     - j
91     - k
92     - l
93     - m
94     - 'n'
95     - o
96     - p
97     - q
98     - r
99     - s
100    - t
101    - u
102    - v
103    - w
104    - x
105    - 'y'
106    - z
107    batch_size: 5
108    shuffle: false
109    normalise transcripts: false
110    num_workers: 8
111    pin_memory: true
112    trim silence: true
113 preprocessor:
114     target : nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
115     normalise: per feature
116     window size: 0.02
117     sample rate: 16000
118     window stride: 0.01
119     window: hann
120     features: 64
121     n fft: 512
122     frame splicing: 1
123     dither: 1.0e-05
124     stft_conv: false
125 spec_augment:
126     _target_ : nemo.collections.asr.modules.SpectrogramAugmentation
127     rect_freq: 0
128     rect_masks: 0
129     rect_time: 0
130     freq_masks: 1
131     freq_width: 27
132     time_masks: 1
133     time_width: 100
134 encoder:
135     _target_ : nemo.collections.asr.modules.ConvASREncoder
136     feat_in: 64
137     activation: relu
138     conv_mask: true

```

```

139     jasper:
140     - dilation:
141       - 1
142       dropout: 0.0
143       filters: 256
144       kernel:
145       - 33
146       repeat: 1
147       residual: false
148       separable: true
149       stride:
150       - 2
151     - dilation:
152       - 1
153       dropout: 0.0
154       filters: 256
155       kernel:
156       - 33
157       repeat: 5
158       residual: true
159       separable: true
160       stride:
161       - 1
162     - dilation:
163       - 1
164       dropout: 0.0
165       filters: 256
166       kernel:
167       - 33
168       repeat: 5
169       residual: true
170       separable: true
171       stride:
172       - 1
173     - dilation:
174       - 1
175       dropout: 0.0
176       filters: 256
177       kernel:
178       - 33
179       repeat: 5
180       residual: true
181       separable: true
182       stride:
183       - 1
184     - dilation:
185       - 1
186       dropout: 0.0
187       filters: 256
188       kernel:
189       - 39
190       repeat: 5
191       residual: true
192       separable: true
193       stride:
194       - 1
195     - dilation:
196       - 1
197       dropout: 0.0
198       filters: 256
199       kernel:
200       - 39
201       repeat: 5
202       residual: true
203       separable: true
204       stride:
205       - 1
206     - dilation:
207       - 1

```

```

208         dropout: 0.0
209         filters: 256
210         kernel:
211         - 39
212         repeat: 5
213         residual: true
214         separable: true
215         stride:
216         - 1
217     - dilation:
218     - 1
219       dropout: 0.0
220       filters: 512
221       kernel:
222       - 51
223       repeat: 5
224       residual: true
225       separable: true
226       stride:
227       - 1
228     - dilation:
229     - 1
230       dropout: 0.0
231       filters: 512
232       kernel:
233       - 51
234       repeat: 5
235       residual: true
236       separable: true
237       stride:
238       - 1
239     - dilation:
240     - 1
241       dropout: 0.0
242       filters: 512
243       kernel:
244       - 51
245       repeat: 5
246       residual: true
247       separable: true
248       stride:
249       - 1
250     - dilation:
251     - 1
252       dropout: 0.0
253       filters: 512
254       kernel:
255       - 63
256       repeat: 5
257       residual: true
258       separable: true
259       stride:
260       - 1
261     - dilation:
262     - 1
263       dropout: 0.0
264       filters: 512
265       kernel:
266       - 63
267       repeat: 5
268       residual: true
269       separable: true
270       stride:
271       - 1
272     - dilation:
273     - 1
274       dropout: 0.0
275       filters: 512
276       kernel:

```

```

277         - 63
278         repeat: 5
279         residual: true
280         separable: true
281         stride:
282         - 1
283     - dilation:
284     - 1
285     dropout: 0.0
286     filters: 512
287     kernel:
288     - 75
289     repeat: 5
290     residual: true
291     separable: true
292     stride:
293     - 1
294     - dilation:
295     - 1
296     dropout: 0.0
297     filters: 512
298     kernel:
299     - 75
300     repeat: 5
301     residual: true
302     separable: true
303     stride:
304     - 1
305     - dilation:
306     - 1
307     dropout: 0.0
308     filters: 512
309     kernel:
310     - 75
311     repeat: 5
312     residual: true
313     separable: true
314     stride:
315     - 1
316     - dilation:
317     - 2
318     dropout: 0.0
319     filters: 512
320     kernel:
321     - 87
322     repeat: 1
323     residual: false
324     separable: true
325     stride:
326     - 1
327     - dilation:
328     - 1
329     dropout: 0.0
330     filters: 1024
331     kernel:
332     - 1
333     repeat: 1
334     residual: false
335     stride:
336     - 1
337 decoder:
338     _target_: nemo.collections.asr.modules.ConvASRDecoder
339     feat_in: 1024
340     num_classes: 28
341     vocabulary:
342     - ' '
343     - a
344     - b
345     - c

```

```

346     - d
347     - e
348     - f
349     - g
350     - h
351     - i
352     - j
353     - k
354     - l
355     - m
356     - 'n'
357     - o
358     - p
359     - q
360     - r
361     - s
362     - t
363     - u
364     - v
365     - w
366     - x
367     - 'y'
368     - z
369     - ''''
370 optim:
371     name: novograd
372     lr: 0.001
373     betas:
374     - 0.8
375     - 0.5
376     weight_decay: 0.001
377     sched:
378         name: CosineAnnealing
379         warmup_steps: null
380         warmup_ratio: 0.05
381         min_lr: 1.0e-05
382         last_epoch: -1
383 target: nemo.collections.asr.models.ctc_models.EncDecCTCModel
384 nemo_version: 1.9.0rc0
385

```

APPENDIX K: YAML file of the model architecture specified for the Quartznet 15×5 English model during the conduct of the TL with Beta Optimization (Best Model – with (0.9, 0.1)) setting experiment.

---

```
1  cfg:
2    sample_rate: 16000
3    repeat: 5
4    dropout: 0.0
5    separable: true
6    labels:
7      - ' '
8      - a
9      - b
10     - c
11     - d
12     - e
13     - f
14     - g
15     - h
16     - i
17     - j
18     - k
19     - l
20     - m
21     - 'n'
22     - o
23     - p
24     - q
25     - r
26     - s
27     - t
28     - u
29     - v
30     - w
31     - x
32     - 'y'
33     - z
34     - ''
35  train_ds:
36    manifest_filepath: ./iban/train_manifest.json
37    sample_rate: 16000
38    labels:
39      - ' '
40      - a
41      - b
42      - c
43      - d
44      - e
45      - f
46      - g
47      - h
48      - i
49      - j
50      - k
51      - l
52      - m
53      - 'n'
54      - o
55      - p
56      - q
57      - r
58      - s
59      - t
60      - u
61      - v
62      - w
63      - x
64      - 'y'
65      - z
66    batch_size: 5
67    trim_silence: true
68    max_duration: 16.7
69    shuffle: true
```

```

70     is tarred: false
71     tarred audio filepaths: null
72     tarred shard strategy: scatter
73     normalise transcripts: false
74     num_workers: 8
75     pin_memory: true
76 validation ds:
77     manifest_filepath: ./iban/test_manifest.json
78     sample_rate: 16000
79     labels:
80     - ' '
81     - a
82     - b
83     - c
84     - d
85     - e
86     - f
87     - g
88     - h
89     - i
90     - j
91     - k
92     - l
93     - m
94     - 'n'
95     - o
96     - p
97     - q
98     - r
99     - s
100    - t
101    - u
102    - v
103    - w
104    - x
105    - 'y'
106    - z
107    batch_size: 5
108    shuffle: false
109    normalise transcripts: false
110    num_workers: 8
111    pin_memory: true
112    trim silence: true
113 preprocessor:
114     target : nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
115     normalise: per feature
116     window size: 0.02
117     sample rate: 16000
118     window stride: 0.01
119     window: hann
120     features: 64
121     n fft: 512
122     frame splicing: 1
123     dither: 1.0e-05
124     stft_conv: false
125 spec_augment:
126     _target_ : nemo.collections.asr.modules.SpectrogramAugmentation
127     rect_freq: 0
128     rect_masks: 0
129     rect_time: 0
130     freq_masks: 1
131     freq_width: 27
132     time_masks: 1
133     time_width: 100
134 encoder:
135     _target_ : nemo.collections.asr.modules.ConvASREncoder
136     feat_in: 64
137     activation: relu
138     conv_mask: true

```

```

139     jasper:
140     - dilation:
141       - 1
142       dropout: 0.0
143       filters: 256
144       kernel:
145       - 33
146       repeat: 1
147       residual: false
148       separable: true
149       stride:
150       - 2
151     - dilation:
152       - 1
153       dropout: 0.0
154       filters: 256
155       kernel:
156       - 33
157       repeat: 5
158       residual: true
159       separable: true
160       stride:
161       - 1
162     - dilation:
163       - 1
164       dropout: 0.0
165       filters: 256
166       kernel:
167       - 33
168       repeat: 5
169       residual: true
170       separable: true
171       stride:
172       - 1
173     - dilation:
174       - 1
175       dropout: 0.0
176       filters: 256
177       kernel:
178       - 33
179       repeat: 5
180       residual: true
181       separable: true
182       stride:
183       - 1
184     - dilation:
185       - 1
186       dropout: 0.0
187       filters: 256
188       kernel:
189       - 39
190       repeat: 5
191       residual: true
192       separable: true
193       stride:
194       - 1
195     - dilation:
196       - 1
197       dropout: 0.0
198       filters: 256
199       kernel:
200       - 39
201       repeat: 5
202       residual: true
203       separable: true
204       stride:
205       - 1
206     - dilation:
207       - 1

```

```

208 dropout: 0.0
209 filters: 256
210 kernel:
211 - 39
212 repeat: 5
213 residual: true
214 separable: true
215 stride:
216 - 1
217 - dilation:
218 - 1
219 dropout: 0.0
220 filters: 512
221 kernel:
222 - 51
223 repeat: 5
224 residual: true
225 separable: true
226 stride:
227 - 1
228 - dilation:
229 - 1
230 dropout: 0.0
231 filters: 512
232 kernel:
233 - 51
234 repeat: 5
235 residual: true
236 separable: true
237 stride:
238 - 1
239 - dilation:
240 - 1
241 dropout: 0.0
242 filters: 512
243 kernel:
244 - 51
245 repeat: 5
246 residual: true
247 separable: true
248 stride:
249 - 1
250 - dilation:
251 - 1
252 dropout: 0.0
253 filters: 512
254 kernel:
255 - 63
256 repeat: 5
257 residual: true
258 separable: true
259 stride:
260 - 1
261 - dilation:
262 - 1
263 dropout: 0.0
264 filters: 512
265 kernel:
266 - 63
267 repeat: 5
268 residual: true
269 separable: true
270 stride:
271 - 1
272 - dilation:
273 - 1
274 dropout: 0.0
275 filters: 512
276 kernel:

```

```

277         - 63
278         repeat: 5
279         residual: true
280         separable: true
281         stride:
282         - 1
283     - dilation:
284     - 1
285     dropout: 0.0
286     filters: 512
287     kernel:
288     - 75
289     repeat: 5
290     residual: true
291     separable: true
292     stride:
293     - 1
294     - dilation:
295     - 1
296     dropout: 0.0
297     filters: 512
298     kernel:
299     - 75
300     repeat: 5
301     residual: true
302     separable: true
303     stride:
304     - 1
305     - dilation:
306     - 1
307     dropout: 0.0
308     filters: 512
309     kernel:
310     - 75
311     repeat: 5
312     residual: true
313     separable: true
314     stride:
315     - 1
316     - dilation:
317     - 2
318     dropout: 0.0
319     filters: 512
320     kernel:
321     - 87
322     repeat: 1
323     residual: false
324     separable: true
325     stride:
326     - 1
327     - dilation:
328     - 1
329     dropout: 0.0
330     filters: 1024
331     kernel:
332     - 1
333     repeat: 1
334     residual: false
335     stride:
336     - 1
337 decoder:
338     _target_: nemo.collections.asr.modules.ConvASRDecoder
339     feat_in: 1024
340     num_classes: 28
341     vocabulary:
342     - ' '
343     - a
344     - b
345     - c

```

```

346     - d
347     - e
348     - f
349     - g
350     - h
351     - i
352     - j
353     - k
354     - l
355     - m
356     - 'n'
357     - o
358     - p
359     - q
360     - r
361     - s
362     - t
363     - u
364     - v
365     - w
366     - x
367     - 'y'
368     - z
369     - ''
370 optim:
371     name: novograd
372     lr: 0.001
373     betas:
374     - 0.9
375     - 0.1
376     weight_decay: 0.001
377     sched:
378     name: CosineAnnealing
379     warmup_steps: null
380     warmup_ratio: 0.05
381     min_lr: 1.0e-05
382     last_epoch: -1
383 target: nemo.collections.asr.models.ctc_models.EncDecCTCModel
384 nemo_version: 1.9.0rc0
385

```

APPENDIX L: YAML file of the model architecture specified for the Quartznet 20×1 Malay model during the conduct of the Basic TL setting experiment.

---

```
1  cfg:
2    train ds:
3      manifest filepath: ./iban/train manifest.json
4      sample rate: 16000
5      labels:
6        - ' '
7        - a
8        - b
9        - c
10       - d
11       - e
12       - f
13       - g
14       - h
15       - i
16       - j
17       - k
18       - l
19       - m
20       - 'n'
21       - o
22       - p
23       - q
24       - r
25       - s
26       - t
27       - u
28       - v
29       - w
30       - x
31       - 'y'
32       - z
33     batch_size: 5
34     trim_silence: true
35     max_duration: 16.7
36     shuffle: true
37     is_tarred: false
38     tarred_audio_filepaths: null
39     tarred_shard_strategy: scatter
40     normalise_transcripts: false
41     num_workers: 8
42     pin_memory: true
43   validation ds:
44     manifest filepath: ./iban/test manifest.json
45     sample rate: 16000
46     labels:
47       - ' '
48       - a
49       - b
50       - c
51       - d
52       - e
53       - f
54       - g
55       - h
56       - i
57       - j
58       - k
59       - l
60       - m
61       - 'n'
62       - o
63       - p
64       - q
65       - r
66       - s
67       - t
68       - u
69       - v
```

```

70     - w
71     - x
72     - 'y'
73     - z
74     batch_size: 5
75     shuffle: false
76     normalise_transcripts: false
77     num_workers: 8
78     pin_memory: true
79     trim_silence: true
80     preprocessor:
81         _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
82         normalise: per_feature
83         window_size: 0.02
84         sample_rate: 16000
85         window_stride: 0.01
86         window: hann
87         features: 64
88         n_fft: 512
89         frame_splicing: 1
90         dither: 1.0e-05
91         stft_conv: false
92     spec_augment:
93         target: nemo.collections.asr.modules.SpectrogramAugmentation
94         rect_freq: 0
95         rect_masks: 0
96         rect_time: 0
97     encoder:
98         target: nemo.collections.asr.modules.ConvASREncoder
99         feat_in: 64
100        activation: relu
101        conv_mask: true
102        jasper:
103            - filters: 128
104              repeat: 1
105              kernel:
106                  - 11
107              stride:
108                  - 1
109              dilation:
110                  - 1
111              dropout: 0.0
112              residual: true
113              separable: true
114              se: true
115              se_context_size: -1
116            - filters: 256
117              repeat: 1
118              kernel:
119                  - 13
120              stride:
121                  - 1
122              dilation:
123                  - 1
124              dropout: 0.0
125              residual: true
126              separable: true
127              se: true
128              se_context_size: -1
129            - filters: 256
130              repeat: 1
131              kernel:
132                  - 15
133              stride:
134                  - 1
135              dilation:
136                  - 1
137              dropout: 0.0
138              residual: true

```

```

139         separable: true
140         se: true
141         se context size: -1
142     - filters: 256
143       repeat: 1
144       kernel:
145         - 17
146       stride:
147         - 1
148       dilation:
149         - 1
150       dropout: 0.0
151       residual: true
152       separable: true
153       se: true
154       se context size: -1
155   - filters: 256
156     repeat: 1
157     kernel:
158       - 19
159     stride:
160       - 1
161     dilation:
162       - 1
163     dropout: 0.0
164     residual: true
165     separable: true
166     se: true
167     se context size: -1
168   - filters: 256
169     repeat: 1
170     kernel:
171       - 21
172     stride:
173       - 1
174     dilation:
175       - 1
176     dropout: 0.0
177     residual: true
178     separable: true
179     se: true
180     se context size: -1
181   - filters: 256
182     repeat: 1
183     kernel:
184       - 23
185     stride:
186       - 1
187     dilation:
188       - 1
189     dropout: 0.0
190     residual: true
191     separable: true
192     se: true
193     se context size: -1
194   - filters: 256
195     repeat: 1
196     kernel:
197       - 25
198     stride:
199       - 1
200     dilation:
201       - 1
202     dropout: 0.0
203     residual: true
204     separable: true
205     se: true
206     se context size: -1
207   - filters: 256

```

```

208         repeat: 1
209         kernel:
210         - 27
211         stride:
212         - 1
213         dilation:
214         - 1
215         dropout: 0.0
216         residual: true
217         separable: true
218         se: true
219         se_context_size: -1
220     - filters: 256
221       repeat: 1
222       kernel:
223       - 29
224       stride:
225       - 1
226       dilation:
227       - 1
228       dropout: 0.0
229       residual: true
230       separable: true
231       se: true
232       se_context_size: -1
233   - filters: 256
234     repeat: 1
235     kernel:
236     - 31
237     stride:
238     - 1
239     dilation:
240     - 1
241     dropout: 0.0
242     residual: true
243     separable: true
244     se: true
245     se_context_size: -1
246   - filters: 256
247     repeat: 1
248     kernel:
249     - 33
250     stride:
251     - 1
252     dilation:
253     - 1
254     dropout: 0.0
255     residual: true
256     separable: true
257     se: true
258     se_context_size: -1
259   - filters: 256
260     repeat: 1
261     kernel:
262     - 35
263     stride:
264     - 1
265     dilation:
266     - 1
267     dropout: 0.0
268     residual: true
269     separable: true
270     se: true
271     se_context_size: -1
272   - filters: 256
273     repeat: 1
274     kernel:
275     - 37
276     stride:

```

```

277         - 1
278         dilation:
279         - 1
280         dropout: 0.0
281         residual: true
282         separable: true
283         se: true
284         se_context_size: -1
285     - filters: 256
286       repeat: 1
287       kernel:
288       - 39
289       stride:
290       - 1
291       dilation:
292       - 1
293       dropout: 0.0
294       residual: true
295       separable: true
296       se: true
297       se_context_size: -1
298   - filters: 256
299     repeat: 1
300     kernel:
301     - 41
302     stride:
303     - 1
304     dilation:
305     - 1
306     dropout: 0.0
307     residual: true
308     separable: true
309     se: true
310     se_context_size: -1
311   - filters: 256
312     repeat: 1
313     kernel:
314     - 43
315     stride:
316     - 1
317     dilation:
318     - 1
319     dropout: 0.0
320     residual: true
321     separable: true
322     se: true
323     se_context_size: -1
324   - filters: 256
325     repeat: 1
326     kernel:
327     - 45
328     stride:
329     - 1
330     dilation:
331     - 1
332     dropout: 0.0
333     residual: true
334     separable: true
335     se: true
336     se_context_size: -1
337   - filters: 256
338     repeat: 1
339     kernel:
340     - 47
341     stride:
342     - 1
343     dilation:
344     - 1
345     dropout: 0.0

```

```

346         residual: true
347         separable: true
348         se: true
349         se_context_size: -1
350     - filters: 256
351       repeat: 1
352       kernel:
353         - 49
354       stride:
355         - 1
356       dilation:
357         - 1
358       dropout: 0.0
359       residual: true
360       separable: true
361       se: true
362       se_context_size: -1
363   - filters: 256
364     repeat: 1
365     kernel:
366       - 51
367     stride:
368       - 1
369     dilation:
370       - 1
371     dropout: 0.0
372     residual: true
373     separable: true
374     se: true
375     se_context_size: -1
376   - filters: 256
377     repeat: 1
378     kernel:
379       - 53
380     stride:
381       - 1
382     dilation:
383       - 1
384     dropout: 0.0
385     residual: false
386     separable: true
387     se: true
388     se_context_size: -1
389   - filters: 1024
390     repeat: 1
391     kernel:
392       - 1
393     stride:
394       - 1
395     dilation:
396       - 1
397     dropout: 0.0
398     residual: false
399     separable: true
400     se: true
401     se_context_size: -1
402 decoder:
403   _target_: nemo.collections.asr.modules.ConvASRDecoder
404   feat_in: 1024
405   num_classes: 28
406   vocabulary:
407     - ' '
408     - a
409     - b
410     - c
411     - d
412     - e
413     - f
414     - g

```

```

415     - h
416     - i
417     - j
418     - k
419     - l
420     - m
421     - 'n'
422     - o
423     - p
424     - q
425     - r
426     - s
427     - t
428     - u
429     - v
430     - w
431     - x
432     - 'y'
433     - z
434     - ''''
435 optim:
436     name: novograd
437     lr: 0.001
438     betas:
439     - 0.8
440     - 0.5
441     weight_decay: 0.001
442     sched:
443         name: CosineAnnealing
444         monitor: val_loss
445         reduce_on_plateau: false
446         warmup_steps: null
447         warmup_ratio: 0.05
448         min_lr: 1.0e-05
449         last_epoch: -1
450 target: nemo.collections.asr.models.ctc_models.EncDecCTCModel
451 nemo_version: 1.9.0rc0
452

```

APPENDIX M: YAML file of the model architecture specified for the Quartznet 20×1 Malay model during the conduct of the TL with SA (Best Model – with Time-Frequency SA) setting experiment.

---

```
1  cfg:
2    train ds:
3      manifest filepath: ./iban/train manifest.json
4      sample rate: 16000
5      labels:
6        - ' '
7        - a
8        - b
9        - c
10       - d
11       - e
12       - f
13       - g
14       - h
15       - i
16       - j
17       - k
18       - l
19       - m
20       - 'n'
21       - o
22       - p
23       - q
24       - r
25       - s
26       - t
27       - u
28       - v
29       - w
30       - x
31       - 'y'
32       - z
33     batch_size: 5
34     trim_silence: true
35     max_duration: 16.7
36     shuffle: true
37     is_tarred: false
38     tarred_audio_filepaths: null
39     tarred_shard_strategy: scatter
40     normalise_transcripts: false
41     num_workers: 8
42     pin_memory: true
43   validation ds:
44     manifest filepath: ./iban/test manifest.json
45     sample rate: 16000
46     labels:
47       - ' '
48       - a
49       - b
50       - c
51       - d
52       - e
53       - f
54       - g
55       - h
56       - i
57       - j
58       - k
59       - l
60       - m
61       - 'n'
62       - o
63       - p
64       - q
65       - r
66       - s
67       - t
68       - u
69       - v
```

```

70     - w
71     - x
72     - 'y'
73     - z
74     batch_size: 5
75     shuffle: false
76     normalise_transcripts: false
77     num_workers: 8
78     pin_memory: true
79     trim_silence: true
80     preprocessor:
81         _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
82         normalise: per_feature
83         window_size: 0.02
84         sample_rate: 16000
85         window_stride: 0.01
86         window: hann
87         features: 64
88         n_fft: 512
89         frame_splicing: 1
90         dither: 1.0e-05
91         stft_conv: false
92     spec_augment:
93         _target_: nemo.collections.asr.modules.SpectrogramAugmentation
94         rect_freq: 0
95         rect_masks: 0
96         rect_time: 0
97         freq_masks: 1
98         freq_width: 27
99         time_masks: 1
100        time_width: 100
101    encoder:
102        _target_: nemo.collections.asr.modules.ConvASREncoder
103        feat_in: 64
104        activation: relu
105        conv_mask: true
106        jasper:
107            - filters: 128
108              repeat: 1
109              kernel:
110                  - 11
111              stride:
112                  - 1
113              dilation:
114                  - 1
115              dropout: 0.0
116              residual: true
117              separable: true
118              se: true
119              se_context_size: -1
120            - filters: 256
121              repeat: 1
122              kernel:
123                  - 13
124              stride:
125                  - 1
126              dilation:
127                  - 1
128              dropout: 0.0
129              residual: true
130              separable: true
131              se: true
132              se_context_size: -1
133            - filters: 256
134              repeat: 1
135              kernel:
136                  - 15
137              stride:
138                  - 1

```

```

139         dilation:
140             - 1
141         dropout: 0.0
142         residual: true
143         separable: true
144         se: true
145         se_context_size: -1
146     - filters: 256
147       repeat: 1
148     kernel:
149         - 17
150     stride:
151         - 1
152     dilation:
153         - 1
154     dropout: 0.0
155     residual: true
156     separable: true
157     se: true
158     se_context_size: -1
159 - filters: 256
160   repeat: 1
161   kernel:
162       - 19
163   stride:
164       - 1
165   dilation:
166       - 1
167   dropout: 0.0
168   residual: true
169   separable: true
170   se: true
171   se_context_size: -1
172 - filters: 256
173   repeat: 1
174   kernel:
175       - 21
176   stride:
177       - 1
178   dilation:
179       - 1
180   dropout: 0.0
181   residual: true
182   separable: true
183   se: true
184   se_context_size: -1
185 - filters: 256
186   repeat: 1
187   kernel:
188       - 23
189   stride:
190       - 1
191   dilation:
192       - 1
193   dropout: 0.0
194   residual: true
195   separable: true
196   se: true
197   se_context_size: -1
198 - filters: 256
199   repeat: 1
200   kernel:
201       - 25
202   stride:
203       - 1
204   dilation:
205       - 1
206   dropout: 0.0
207   residual: true

```

```

208     separable: true
209     se: true
210     se context size: -1
211 - filters: 256
212   repeat: 1
213   kernel:
214     - 27
215   stride:
216     - 1
217   dilation:
218     - 1
219   dropout: 0.0
220   residual: true
221   separable: true
222   se: true
223   se context size: -1
224 - filters: 256
225   repeat: 1
226   kernel:
227     - 29
228   stride:
229     - 1
230   dilation:
231     - 1
232   dropout: 0.0
233   residual: true
234   separable: true
235   se: true
236   se context size: -1
237 - filters: 256
238   repeat: 1
239   kernel:
240     - 31
241   stride:
242     - 1
243   dilation:
244     - 1
245   dropout: 0.0
246   residual: true
247   separable: true
248   se: true
249   se context size: -1
250 - filters: 256
251   repeat: 1
252   kernel:
253     - 33
254   stride:
255     - 1
256   dilation:
257     - 1
258   dropout: 0.0
259   residual: true
260   separable: true
261   se: true
262   se context size: -1
263 - filters: 256
264   repeat: 1
265   kernel:
266     - 35
267   stride:
268     - 1
269   dilation:
270     - 1
271   dropout: 0.0
272   residual: true
273   separable: true
274   se: true
275   se context size: -1
276 - filters: 256

```

```

277         repeat: 1
278         kernel:
279         - 37
280         stride:
281         - 1
282         dilation:
283         - 1
284         dropout: 0.0
285         residual: true
286         separable: true
287         se: true
288         se_context_size: -1
289     - filters: 256
290       repeat: 1
291       kernel:
292       - 39
293       stride:
294       - 1
295       dilation:
296       - 1
297       dropout: 0.0
298       residual: true
299       separable: true
300       se: true
301       se_context_size: -1
302   - filters: 256
303     repeat: 1
304     kernel:
305     - 41
306     stride:
307     - 1
308     dilation:
309     - 1
310     dropout: 0.0
311     residual: true
312     separable: true
313     se: true
314     se_context_size: -1
315   - filters: 256
316     repeat: 1
317     kernel:
318     - 43
319     stride:
320     - 1
321     dilation:
322     - 1
323     dropout: 0.0
324     residual: true
325     separable: true
326     se: true
327     se_context_size: -1
328   - filters: 256
329     repeat: 1
330     kernel:
331     - 45
332     stride:
333     - 1
334     dilation:
335     - 1
336     dropout: 0.0
337     residual: true
338     separable: true
339     se: true
340     se_context_size: -1
341   - filters: 256
342     repeat: 1
343     kernel:
344     - 47
345     stride:

```

```

346         - 1
347         dilation:
348         - 1
349         dropout: 0.0
350         residual: true
351         separable: true
352         se: true
353         se_context_size: -1
354     - filters: 256
355       repeat: 1
356       kernel:
357       - 49
358       stride:
359       - 1
360       dilation:
361       - 1
362       dropout: 0.0
363       residual: true
364       separable: true
365       se: true
366       se_context_size: -1
367   - filters: 256
368     repeat: 1
369     kernel:
370     - 51
371     stride:
372     - 1
373     dilation:
374     - 1
375     dropout: 0.0
376     residual: true
377     separable: true
378     se: true
379     se_context_size: -1
380   - filters: 256
381     repeat: 1
382     kernel:
383     - 53
384     stride:
385     - 1
386     dilation:
387     - 1
388     dropout: 0.0
389     residual: false
390     separable: true
391     se: true
392     se_context_size: -1
393   - filters: 1024
394     repeat: 1
395     kernel:
396     - 1
397     stride:
398     - 1
399     dilation:
400     - 1
401     dropout: 0.0
402     residual: false
403     separable: true
404     se: true
405     se_context_size: -1
406 decoder:
407   _target_: nemo.collections.asr.modules.ConvASRDecoder
408   feat_in: 1024
409   num_classes: 28
410   vocabulary:
411   - ' '
412   - a
413   - b
414   - c

```

```

415     - d
416     - e
417     - f
418     - g
419     - h
420     - i
421     - j
422     - k
423     - l
424     - m
425     - 'n'
426     - o
427     - p
428     - q
429     - r
430     - s
431     - t
432     - u
433     - v
434     - w
435     - x
436     - 'y'
437     - z
438     - ''''
439     optim:
440         name: novograd
441         lr: 0.001
442         betas:
443             - 0.8
444             - 0.5
445         weight_decay: 0.001
446         sched:
447             name: CosineAnnealing
448             monitor: val_loss
449             reduce_on_plateau: false
450             warmup_steps: null
451             warmup_ratio: 0.05
452             min_lr: 1.0e-05
453             last_epoch: -1
454     target: nemo.collections.asr.models.ctc_models.EncDecCTCModel
455     nemo_version: 1.9.0rc0
456

```

APPENDIX N: YAML file of the model architecture specified for the Quartznet 20×1 Malay model during the conduct of the TL with Beta Optimization (Best Model – with (0.9, 0.5)) setting experiment.

---

```
1  cfg:
2    train ds:
3      manifest filepath: ./iban/train manifest.json
4      sample rate: 16000
5      labels:
6        - ' '
7        - a
8        - b
9        - c
10       - d
11       - e
12       - f
13       - g
14       - h
15       - i
16       - j
17       - k
18       - l
19       - m
20       - 'n'
21       - o
22       - p
23       - q
24       - r
25       - s
26       - t
27       - u
28       - v
29       - w
30       - x
31       - 'y'
32       - z
33     batch_size: 5
34     trim_silence: true
35     max_duration: 16.7
36     shuffle: true
37     is_tarred: false
38     tarred_audio_filepaths: null
39     tarred_shard_strategy: scatter
40     normalise_transcripts: false
41     num_workers: 8
42     pin_memory: true
43   validation ds:
44     manifest filepath: ./iban/test manifest.json
45     sample rate: 16000
46     labels:
47       - ' '
48       - a
49       - b
50       - c
51       - d
52       - e
53       - f
54       - g
55       - h
56       - i
57       - j
58       - k
59       - l
60       - m
61       - 'n'
62       - o
63       - p
64       - q
65       - r
66       - s
67       - t
68       - u
69       - v
```

```

70     - w
71     - x
72     - 'y'
73     - z
74     batch_size: 5
75     shuffle: false
76     normalise_transcripts: false
77     num_workers: 8
78     pin_memory: true
79     trim_silence: true
80     preprocessor:
81         _target_: nemo.collections.asr.modules.AudioToMelSpectrogramPreprocessor
82         normalise: per_feature
83         window_size: 0.02
84         sample_rate: 16000
85         window_stride: 0.01
86         window: hann
87         features: 64
88         n_fft: 512
89         frame_splicing: 1
90         dither: 1.0e-05
91         stft_conv: false
92     spec_augment:
93         target: nemo.collections.asr.modules.SpectrogramAugmentation
94         rect_freq: 0
95         rect_masks: 0
96         rect_time: 0
97         freq_masks: 1
98         freq_width: 27
99         time_masks: 1
100        time_width: 100
101    encoder:
102        _target_: nemo.collections.asr.modules.ConvASREncoder
103        feat_in: 64
104        activation: relu
105        conv_mask: true
106        jasper:
107            - filters: 128
108              repeat: 1
109              kernel:
110                  - 11
111              stride:
112                  - 1
113              dilation:
114                  - 1
115              dropout: 0.0
116              residual: true
117              separable: true
118              se: true
119              se_context_size: -1
120            - filters: 256
121              repeat: 1
122              kernel:
123                  - 13
124              stride:
125                  - 1
126              dilation:
127                  - 1
128              dropout: 0.0
129              residual: true
130              separable: true
131              se: true
132              se_context_size: -1
133            - filters: 256
134              repeat: 1
135              kernel:
136                  - 15
137              stride:
138                  - 1

```

```

139     dilation:
140     - 1
141     dropout: 0.0
142     residual: true
143     separable: true
144     se: true
145     se_context_size: -1
146 - filters: 256
147   repeat: 1
148   kernel:
149   - 17
150   stride:
151   - 1
152   dilation:
153   - 1
154   dropout: 0.0
155   residual: true
156   separable: true
157   se: true
158   se_context_size: -1
159 - filters: 256
160   repeat: 1
161   kernel:
162   - 19
163   stride:
164   - 1
165   dilation:
166   - 1
167   dropout: 0.0
168   residual: true
169   separable: true
170   se: true
171   se_context_size: -1
172 - filters: 256
173   repeat: 1
174   kernel:
175   - 21
176   stride:
177   - 1
178   dilation:
179   - 1
180   dropout: 0.0
181   residual: true
182   separable: true
183   se: true
184   se_context_size: -1
185 - filters: 256
186   repeat: 1
187   kernel:
188   - 23
189   stride:
190   - 1
191   dilation:
192   - 1
193   dropout: 0.0
194   residual: true
195   separable: true
196   se: true
197   se_context_size: -1
198 - filters: 256
199   repeat: 1
200   kernel:
201   - 25
202   stride:
203   - 1
204   dilation:
205   - 1
206   dropout: 0.0
207   residual: true

```

```

208     separable: true
209     se: true
210     se context size: -1
211 - filters: 256
212   repeat: 1
213   kernel:
214     - 27
215   stride:
216     - 1
217   dilation:
218     - 1
219   dropout: 0.0
220   residual: true
221   separable: true
222   se: true
223   se context size: -1
224 - filters: 256
225   repeat: 1
226   kernel:
227     - 29
228   stride:
229     - 1
230   dilation:
231     - 1
232   dropout: 0.0
233   residual: true
234   separable: true
235   se: true
236   se context size: -1
237 - filters: 256
238   repeat: 1
239   kernel:
240     - 31
241   stride:
242     - 1
243   dilation:
244     - 1
245   dropout: 0.0
246   residual: true
247   separable: true
248   se: true
249   se context size: -1
250 - filters: 256
251   repeat: 1
252   kernel:
253     - 33
254   stride:
255     - 1
256   dilation:
257     - 1
258   dropout: 0.0
259   residual: true
260   separable: true
261   se: true
262   se context size: -1
263 - filters: 256
264   repeat: 1
265   kernel:
266     - 35
267   stride:
268     - 1
269   dilation:
270     - 1
271   dropout: 0.0
272   residual: true
273   separable: true
274   se: true
275   se context size: -1
276 - filters: 256

```

```

277         repeat: 1
278         kernel:
279         - 37
280         stride:
281         - 1
282         dilation:
283         - 1
284         dropout: 0.0
285         residual: true
286         separable: true
287         se: true
288         se_context_size: -1
289     - filters: 256
290       repeat: 1
291       kernel:
292       - 39
293       stride:
294       - 1
295       dilation:
296       - 1
297       dropout: 0.0
298       residual: true
299       separable: true
300       se: true
301       se_context_size: -1
302   - filters: 256
303     repeat: 1
304     kernel:
305     - 41
306     stride:
307     - 1
308     dilation:
309     - 1
310     dropout: 0.0
311     residual: true
312     separable: true
313     se: true
314     se_context_size: -1
315   - filters: 256
316     repeat: 1
317     kernel:
318     - 43
319     stride:
320     - 1
321     dilation:
322     - 1
323     dropout: 0.0
324     residual: true
325     separable: true
326     se: true
327     se_context_size: -1
328   - filters: 256
329     repeat: 1
330     kernel:
331     - 45
332     stride:
333     - 1
334     dilation:
335     - 1
336     dropout: 0.0
337     residual: true
338     separable: true
339     se: true
340     se_context_size: -1
341   - filters: 256
342     repeat: 1
343     kernel:
344     - 47
345     stride:

```

```

346         - 1
347         dilation:
348         - 1
349         dropout: 0.0
350         residual: true
351         separable: true
352         se: true
353         se_context_size: -1
354     - filters: 256
355       repeat: 1
356       kernel:
357       - 49
358       stride:
359       - 1
360       dilation:
361       - 1
362       dropout: 0.0
363       residual: true
364       separable: true
365       se: true
366       se_context_size: -1
367     - filters: 256
368       repeat: 1
369       kernel:
370       - 51
371       stride:
372       - 1
373       dilation:
374       - 1
375       dropout: 0.0
376       residual: true
377       separable: true
378       se: true
379       se_context_size: -1
380     - filters: 256
381       repeat: 1
382       kernel:
383       - 53
384       stride:
385       - 1
386       dilation:
387       - 1
388       dropout: 0.0
389       residual: false
390       separable: true
391       se: true
392       se_context_size: -1
393     - filters: 1024
394       repeat: 1
395       kernel:
396       - 1
397       stride:
398       - 1
399       dilation:
400       - 1
401       dropout: 0.0
402       residual: false
403       separable: true
404       se: true
405       se_context_size: -1
406 decoder:
407   _target_: nemo.collections.asr.modules.ConvASRDecoder
408   feat_in: 1024
409   num_classes: 28
410   vocabulary:
411   - ' '
412   - a
413   - b
414   - c

```

```

415     - d
416     - e
417     - f
418     - g
419     - h
420     - i
421     - j
422     - k
423     - l
424     - m
425     - 'n'
426     - o
427     - p
428     - q
429     - r
430     - s
431     - t
432     - u
433     - v
434     - w
435     - x
436     - 'y'
437     - z
438     - ''''
439     optim:
440         name: novograd
441         lr: 0.001
442         betas:
443             - 0.9
444             - 0.5
445         weight_decay: 0.001
446         sched:
447             name: CosineAnnealing
448             monitor: val_loss
449             reduce_on_plateau: false
450             warmup_steps: null
451             warmup_ratio: 0.05
452             min_lr: 1.0e-05
453             last_epoch: -1
454     target: nemo.collections.asr.models.ctc_models.EncDecCTCModel
455     nemo_version: 1.9.0rc0
456

```