UEC2722 --- SPEECH TECHNOLOGY

LOW-RANK ADAPTATION ON WHISPER MODEL FOR TAMIL ASR

PROJECT REPORT

Submitted by

I SYED AZIM
Roll No: 3122 21 3002 110
ECE-B

Department of Electronics and Communication Engineering

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

NOVEMBER 2024

ABSTRACT

Speech processing is a field that involves the study and manipulation of speech signals using computer processes in a digital representation.
This field merges concepts from acoustics, linguistics, signal processing, and computer science to analyze and interpret human speech. Techniques such as Fourier transforms, filtering, machine learning algorithms and neural networks are commonly used to process and analyze speech signals

Speech processing achieves a variety of applications such as

1. Automatic Speech Recognition (ASR),
2. Speaker identification,
3. Language identification
4. Classification

Advances in speech processing have led to significant improvements in technologies like virtual assistants, automated transcription services, and real-time translation tools. This work mainly focuses on extracting learnable features, enhancing accuracy, and reducing the computational complexity of Tamil ASR.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbols	Abbreviation
ML	Machine Learning
MFCC	Mel-Frequency Cepstral Coefficients
CNN	Convolutional Neural Network
LoRA	Low-Rank Adaptation
PEFT	Parameter Efficient fine-tuning
FFT	Full Fine Tunning

CHAPTER 1

INTRODUCTION

Speech processing focuses on processing speech by modeling the human hearing process on a computer and interpreting the audio signal to the computer in a way we understand it. Then we make the system learn the important features of the audio by designing neural network layers to use it for various applications. The process involves several key steps:

1. **Data Collection**: Gathering a diverse set of speech samples that represent various emotional states. This is the most challenging part of any speech-related work. This work is predominantly based on Common Voice Tamil datasets.

2. **Feature Extraction**: Analyzing the speech signal to extract acoustic features such as pitch, energy, formant frequencies, and Mel-frequency cepstral coefficients (MFCCs), which indicate different characteristics of speech.

3. **Feature Selection/Dimensionality Reduction:** Selecting the most relevant features and reducing the feature space's dimensionality to improve the recognition system's efficiency and performance.

4. **Deep Learning**: We design a neural network consisting of an input encoder, hidden layers, and output decoders. We train the model with the features and use it for the desired application or use it as an adaptation for fine-tuning large models

5. **Fine-Tuning:** Fine-tuning is the process of taking a pre-trained machine learning model and further training it on a smaller, targeted data set. Fine-tuning aims to maintain the original capabilities of a pre-trained model while adapting it to suit more specialized use cases. This approach is beneficial when computational resources are limited or data is scarce. The performance of a fine-tuned model can surpass that of the original pre-trained model on the specific tasks for which it was fine-tuned.

   Fine-tuning begins with an existing model which in our case is the Whisper model that has already been trained on a large, diverse speech data set, learning a wide range of underlying features and patterns. After acquiring and preprocessing this additional child speech data, we fine-tune the pre-trained model. The early layers of the neural network, which capture basic features, typically remain unchanged, or "frozen." Later layers, are adjusted as in our case (LoRA) or added as in adapters to capture the new child speech data and better match the task of child speech transcription. This process aims to balance retaining the model's valuable foundational knowledge of speech with improving its performance on fine-tuning child speech.

   To this end, we set a lower learning rate -- a hyperparameter that describes how much a model's weights are adjusted during training and helps prevent drastic changes to the already learned weights, ensuring the model preserves its existing knowledge.

   Benefits of fine-tuning include cost and resource efficiency, better

performance on narrow use cases, and democratization of machine learning capabilities. The risks associated are overfitting, balancing new and previously learned knowledge, and reliance on pre-trained models.

6. **Evaluation**: Assessing the performance of the ASR system using metrics such as accuracy, word error rate (WER), normalized WER, character error rate (CER), and number of deletion, insertion and substitution. The values are noted by varying the models, methods use, dataset used and number of utterances, Using the values trends are plotted.

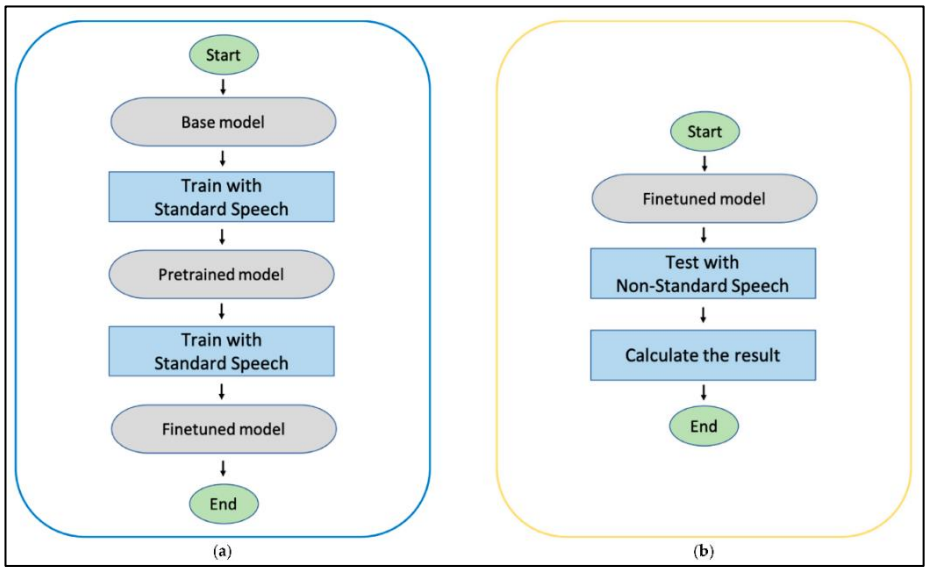


Figure 1.1 Steps for building an ASR System Block Diagram

(Reference: An Effective Learning Method for Automatic Speech Recognition in Korean CI Patients' Speech Jiho Jeong 1 , S. I. M. M. Raton Mondol 1 , Yeon Wook Kim 2 and Sangmin Lee 1,2)

CHAPTER 2

LOW RANK ADAPTATION (LoRA)

2.1 INTRODUCTION

As we pre-train larger models, full fine-tuning, which retrains all model parameters, becomes less feasible. Using Whisper as – to deploy independent instances of fine-tuned models, each with billions of parameters, is prohibitively expensive.

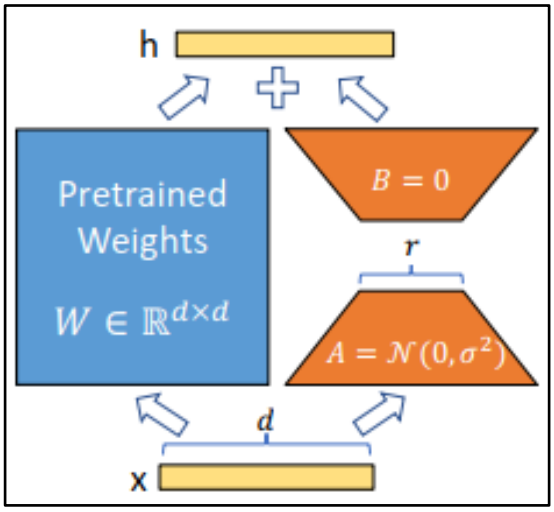


Figure 2.1 LoRA Block diagram

(Reference: Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2022)

Low-**R**ank **A**daptation, or LoRA, freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks. Compared to Whisper fine-tuned model, LoRA can reduce the number of trainable parameters by 100 times and the

GPU memory requirement. LoRA performs on par with finetuning in model quality, despite having fewer trainable parameters, a higher training throughput, and, unlike adapters, no additional inference latency.

2.2 ADVANTAGES

1. A pre-trained model can be shared and used to build many small LoRA modules for different tasks. We can freeze the shared model and efficiently switch tasks by replacing the matrices *A* and *B* in Figure 2.1, reducing the storage requirement and task-switching overhead significantly.
2. LoRA makes training more efficient and lowers the hardware barrier to entry by up to 3 times when using adaptive optimizers since we do not need to calculate the gradients or maintain the optimizer states for most parameters. Instead, only optimize the injected, much smaller low-rank matrices.
3. No Inference Latency*:* Simple linear design allows us to merge the trainable matrices with the frozen weights when deployed, introducing no inference latency compared to a fully fine-tuned model, by construction.
4. LoRA is orthogonal to many prior methods and can be combined with many of them, such as prefix-tuning.

2.3 METHOD

For a pre-trained weight matrix $W_0 \epsilon R_d \times k$, constrain its update by representing the latter with a low-rank decomposition

$$W_0 + \Delta W = W_0 + BA,$$

where $B \epsilon R_d \times r$; $A \epsilon R_r \times k$, and the rank $r \ll \min(d; k)$.

During training, W_0 is frozen and does not receive gradient updates, while A and B contain trainable parameters. Both W_0 and $\Delta W = BA$ are multiplied with the same input, and their respective output vectors are summed coordinate-wise. For $h = W_0 x$, our modified forward pass yields:

$$h = W_0 x + \Delta W x = W0x + BAx$$

A random Gaussian initialization for A and zero for B, so that $\Delta W = BA$ is zero at the beginning of training. Scale $\Delta W x$ by αr , where α is a constant in r. When optimizing with Adam, tuning α is roughly the same as tuning the learning rate if we scale the initialization appropriately. As a result, simply set α to the first r we try and do not tune it. This scaling helps to reduce the need to retune hyperparameters when we vary r.

2.4 APPLYING LORA TO TRANSFORMER

In principle, we can apply LoRA to any subset of weight matrices in a neural network to reduce the number of trainable parameters. In the Transformer architecture, there are four weight matrices in the self-attention module (*Wq; Wk; Wv; Wo*) and two in the MLP module.

We treat *Wq* (or *Wk, Wv,* W_{out}) as a single matrix of dimension *dmodel* ×*dmodel,* even though the output dimension is usually sliced into attention heads. We limit our study to only adapting the attention weights for downstream tasks and freeze the MLP modules (not trained in downstream tasks) for simplicity and parameter efficiency. Further we study the effect on adapting different types of attention weight matrices in a Transformer.

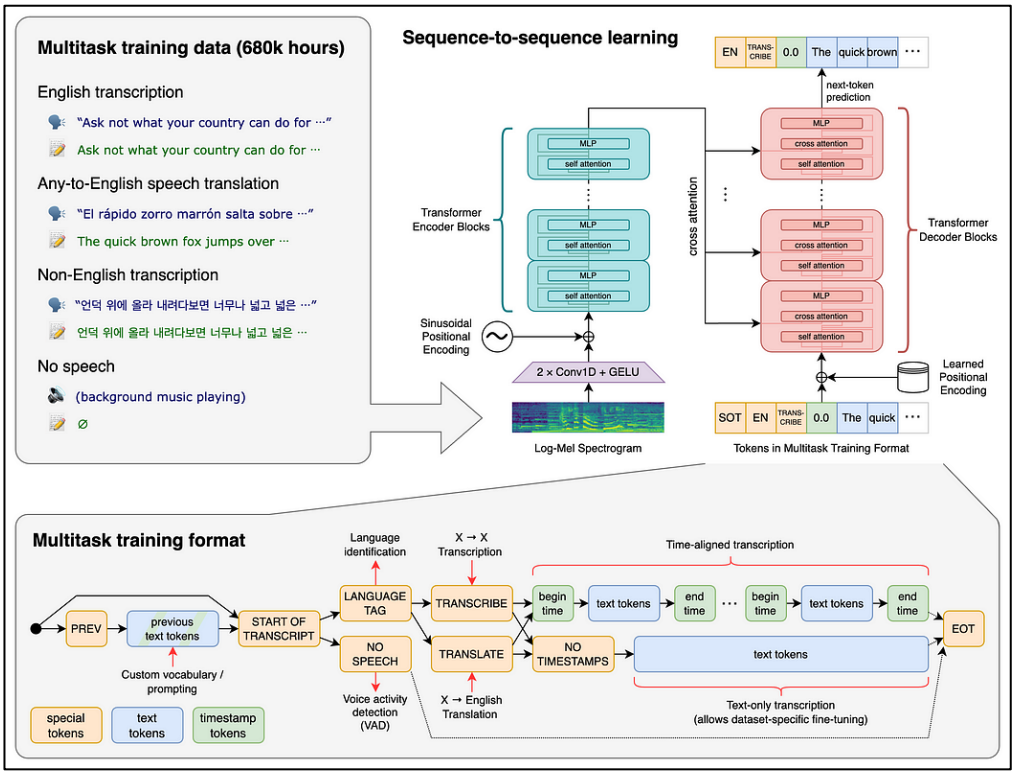


Figure 2.2 Whisper architecture diagram

(Reference: Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2022)

2.5 CONCLUSION

Low-Rank adaptation was applied on whisper model using children speech dataset. The key and output projection were only trained which comprises of about 1% of model parameters. The rank was fixed to 32 and learning rate to 64. The modeled is saved an reloaded for evaluation using test set. The word error rate of the model after applying LoRA is significantly less than the baseline.

CHAPTER 3

EXPERIMENTAL RESULTS OF APPLYING LORA TO FINETUNE WHISPER MODEL

3.1 INTRODUCTION

The low-rank adaptation technique discussed in the previous chapter was implemented by finetuning the Whisper Small Tamil model using publicly available Common Voice Tamil dataset 13.

```
DatasetDict({
    train: Dataset({
        features: ['audio', 'sentence'],
        num_rows: 43350
    })
    test: Dataset({
        features: ['audio', 'sentence'],
        num_rows: 11973
    })
})
```

Example of data:

{'audio':

{'path':'/root/.cache/huggingface/datasets/downloads/extracted/26d01089476eefe8f1950b403fe01fb35c17249845931bb35227afa2fe442bdd/ta_train_0/common_voice_ta_26650298.mp3',

'array': array([0., 0., 0., ..., 0., 0., 0.]),

'sampling_rate': 16000},

'sentence': 'அவரைப் பொதுமக்கள் விடாமல் பின்னாலேயே துரத்திக் கொண்டே ஓடினார்கள்.'}

3.2 EXPERIMENT

LoRA is implemented on key and output projections, for fine-tuning Whisper Small Tamil model with 3M parameters.

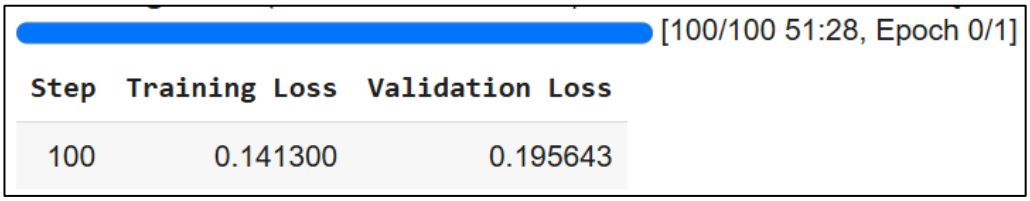
```
trainable params: 3,538,944 || all params: 245,273,856 || trainable%: 1.4429
```

The publicly available Common Voice 13 Tamil speech datasets were used for finetuning the whisper model for Tamil transcription task. Implementation is based on the publicly available *Huggingface Transformers*3 code base. All the experiments are conducted on free available NVIDIA T4 GPUs in the Collab platform.

```
Sun Nov 10 06:33:06 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   75C    P0              31W /  70W |    447MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
+---------------------------------------------------------------------------------------+
```

Typically, α is set as 64 and rank as 32. Besides, in Algorithm 1, we prune singular values every ΔT step (e.g., $\Delta T = 100$) such that the pruned triplets can still get updated within these intervals and possibly reactivated in future iterations. The number of trainable parameters is controlled by the rank r and the number of adapted weight matrices n. The dropout rate is fixed as 0.05 for all experiments



Training Parameters:

TrainOutput(

global_step=100,

training_loss=0.1413337230682373,

metrics={'train_runtime': 3350.6723,

'train_samples_per_second': 0.239,

'train_steps_per_second': 0.03,

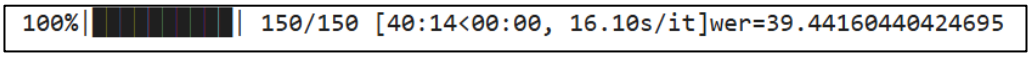
'total_flos': 2.34945183744e+17,

'train_loss': 0.1413337230682373,

'epoch': 0.018453589223103892})

Evaluation Results:

The Word Error Rate by testing the finetuned model with the Common Voice 13 Tamil Test Data is obtained as **39.44160**



3.3 CONCLUSION

Low-rank adaptation was applied to the Whisper Small Tamil Model and a Word Error Rate of 39.44160 was achieved.

CHAPTER 6

CONCLUSION

Reviewed multiple papers to understand the fundamentals of speech processing, full fine-tuning of large models, parameter effective ways of fine-tuning such as adapters (adapter layers inserted between the neural network), and different low-rank adaptation techniques.

I implemented the following:

1. Fine-tuning Whisper Small Tamil model using LoRA

The decision criterion to evaluate the model is Word Error Rate

REFERENCES

[1] Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2022