UNIVERSITÀ
DI TRENTO

DOCTORAL THESIS

Towards Self-configuring Efficient Neural Architectures for Automatic Speech Recognition

Author:
Abdul HANNAN
241319

Supervisor:
Alessio BRUTTI
Daniele FALAVIGNA

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in

Computer Science
Department of Information Engineering and Computer Science
XVIII Cycle

February 17, 2026

Approval Sheet

The thesis entitled *"Towards Self-configuring Efficient Neural Architectures for Automatic Speech Recognition"*, prepared by Abdul HANNANin fulfillment of the requirements for the degree of Doctor of Philosophy is recommended for the final oral examination.

_____
Supervisor
Alessio BRUTTI

Examination Committee

Name	**Signature**	**Date**
Lorenzo VALERIO (External Member)	_____	_____
Cem SUBAKAN (External Member)	_____	_____

Declaration of Authorship

I, Abdul HANNAN, declare that this thesis titled, "Towards Self-configuring Efficient Neural Architectures for Automatic Speech Recognition" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Indeed, with hardship comes ease."

(Al-Inshirah - 94:6)

"It's your road, and yours alone, others may walk it with you, but no one can walk it for you."

Maulana Rumi

"No one can construct for you the bridge upon which precisely you must cross the stream of life, no one but you yourself alone."

"He who has a why to live can bear almost any how."

Friedrich Nietzsche

"Walk on your broken foot and leave no trace of your hand on anyone's shoulder."

Fyodor Dostoevsky

"Never accept an inferior position to anyone. It is the strongest spirit that wins, not the most expensive sword."

Miyamoto Musashi

"Let nothing bend your will nor break your spirit. What stands firm within cannot be conquered."

Unknown

UNIVERSITY OF TRENTO

Abstract

Department of Information Engineering and Computer Science

Doctor of Philosophy

Towards Self-configuring Efficient Neural Architectures for Automatic Speech Recognition

Recent advancements in developing neural architectures has led to the curation of complex deep neural networks which exhibit promising performance on downstream speech applications, like automatic speech recognition. The superior performance of these models is linked to the immense training data and time, along with significant computational resource requirements. Effectively leveraging the capabilities of such models on varying and resource-constrained edge devices pose considerable challenges. Therefore, it is desirable to optimize these models with minimal performance degradation to render them operational on resource constrained devices. In addition, since the on-device resources are mutually employed for device working, the model should have architectural adaptation capability for seamless operation with respect to the on-device resources.

This thesis proposes novel dynamic compression methods to address these limitation by allowing a model to adapt its architecture on-the-fly. The first contribution is the introduction of a semi-dynamic training framework where a lightweight conformer learns the encoder level representation from a large conformer model leveraging knowledge distillation. As a second step, the decoder is appended on top of the distilled model and finetuning is performed to achieve lightweight models with 3x speed-up as compared to existing approaches. The second contribution is the thorough exploration of various inference-time dropping methods and to quantify performance of conformer model when it is trained with random layer dropping with different sparsity levels. This work laid the foundation to achieve the optimum performance-computation trade-off when the encoder layers are dropped at random, and also served as baseline for future works. The third contribution is the introduction of a novel input conditioned layer dropping approach which was applied on speech foundation models, like wavlm and audio spectrogram transformer, to instill the architectural adaptability in them. Additionally, it allows the deployment of speech foundation models on resource-constrained edge devices by reducing their computational load with minimal performance degradation. Lastly, this thesis employs knowledge distillation to further improve the performance-computation trade-off by minimizing the distance between embeddings. The proposed dynamic compression methods achieve comparable performance with 50% reduced computational complexity of what offered by full model. In addition, the compression limit is extendable to 70% with acceptable performance, enabling the model to operate in varying and constrained resource environments.

Keywords: Automatic speech recognition, dynamic model, efficient network, knowledge distillation, layer drop, layer skip.

Acknowledgements

Praise be to ALLAH Almighty Who guided me through every step and gave me strength to stand firm during hard times. It is only because of His countless blessings that I gathered enough tenacity and will to accomplish the research work.

I would like to express sincere gratitude to my supervisors, especially Dr. Alessio Brutti, for their supervision and support. Their constructive comments and suggestions throughout the studies have contributed to the success of this research.

I would also like to thank Dr. Mubashir Noman and Dr. Shah Nawaz for their support, guidance, and contribution throughout the research work. I truly appreciate the patience and time they devoted to review the writing drafts, and providing their valuable feedback.

Lastly and most importantly, I am extremely grateful to my family, especially my Parents, for their unconditional love, support, and prayers. Thank you for every encouragement, every lesson, and every moment of love that carried me to this point.

List of Publications

- Abdul Hannan, Alessio Brutti, and Daniele Falavigna. “LDASR: An Experimental Study on Layer Drop using Conformer-based Architecture”. In: EUSIPCO. 2024.
- Abdul Hannan, Muhammad Arslan Manzoor, Shah Nawaz, Muhammad Irzam Liaqat, Markus Schedl, Mubashir Noman. “An Effective Training Framework for Light-Weight Automatic Speech Recognition Models”. In: Interspeech 2025. 2025, pp. 3613–3617. DOI: 10.21437/Interspeech.2025-1704.
- Abdul Hannan, Daniele Falavigna, and Alessio Brutti. “Input Conditioned Layer Dropping in Speech Foundation Models”. In: 2025 IEEE 35th International Workshop on Machine Learning for Signal Processing (MLSP). IEEE. 2025, pp. 1–6.
- Abdul Hannan, Daniele Falavigna, Shah Nawaz, Mubashir Noman, Markus Schedl, Alessio Brutti. "Distillation-based Layer Dropping (DLD): Effective End-to-end Framework for Dynamic Speech Networks", In ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2026.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
AST	Audio Spectrogram Transformer
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classification
DNN	Deep Neural Network
EE	Early Exit
IDLD	Input-Driven Layer Dropping
KD	Knowledge Distillation
$\mathcal{LD}$	Layer Dropping
ML	Machine Learning
RD	Random Layer Dropping
SLU	Spoken Language Understanding
SOTA	State-of-the-art
WER	Word Error Rate

Dedicated To my Parents and Family

Chapter 1

Introduction

1.1 Background

Speech acts as the primary medium for human communication [31], and serves as a key modality for human-computer interaction (HCI) [17]. Human speech is a complex signal consisting of: phonetic content of the words, prosodic features (word stress, rhythm and intonation) conveying emotions and intent, and unique characteristics of speaker voice (pitch and vocal tract features). This content-rich speech signal allows speech to be extensively used in different applications like automatic speech recognition (ASR), speaker identification and verification, spoken language understanding (SLU), speech emotion recognition (ER), and text-to-speech (TTS). Consequently, speech technology has been widely used in daily-life and commercial speech products, such as interactive virtual assistants like Amazon Alexa and Apple Siri; voice-controlled lighting systems and other smart home appliances; voice-controlled vehicle navigation systems; speech therapy applications; and language learning platforms like Duolingo.

Automatic speech recognition is one of the important area of speech technology that deals with the conversion of spoken language into text, allowing the machine to hear and interpret the spoken language. Figure 1.1 illustrates a general overview of an ASR system that takes the input speech, processes it, and converts it into readable text for humans and machines which can employed for other applications. ASR domain has undergone significant multi-stage evolution from using dynamic time warping [113] for aligning speech patterns to using linear predictive coding (LPC) [3, 55] with fundamental pattern recognition methods [97, 56]. Following that, hidden markov models (HMMs) [61, 63] became the standard for speech modelling [58, 73, 62] followed by utilizing artificial neural networks (ANNs) [84, 77]. The success of AlexNet [69] and the strengths of transformer mechanism [111] lead to the current state-of-the-art (SOTA) architectures like wav2vec 2.0 [4], WavLM [14],

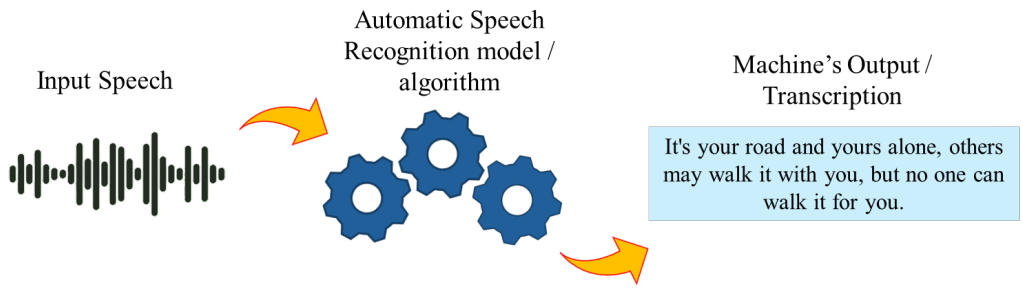

FIGURE 1.1: High level illustration of Automatic Speech Recognition system

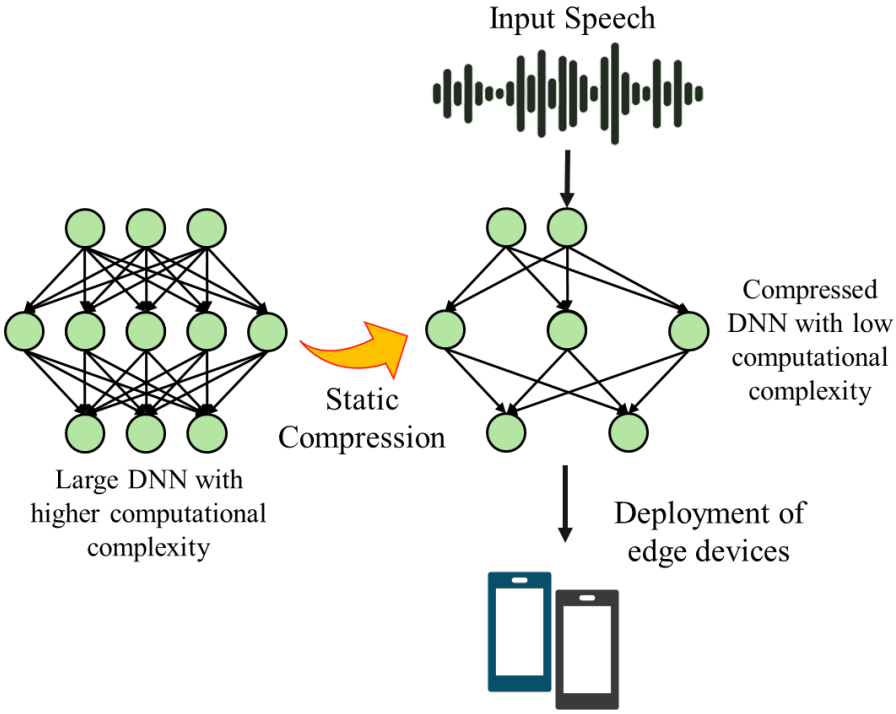

FIGURE 1.2: Static compression flow for feasible deployment of DNNs on edge devices

and conformer [37] demonstrating promising results. This evolution stems from the massive increase in training data, from some words and consonants to vocabulary of $\approx 100 - 1$k words, to corpora comprising of thousands of hours of labeled speech data [87, 48, 116], enabling the development of ASR systems for public and commercial use.

Recent efforts to build ASR systems with superior performance has directed the development of complex DNNs consuming enormous computational and memory resources. Such models provide SOTA performance at the expense of high computational cost and are operable in resource-rich environments, however, these models are inefficient and their deployment to resource-constrained devices is not feasible. In addition, these models drastically suffers in terms of performance when compressed for utilization in resource constrained devices. Furthermore, the computational requirements of these models far exceeds the capacity of edge devices, creating the resource disparity between resource-rich and resource-constrained environments. Hence, to deploy models on edge devices and effectively leverage the on-device processing capabilities in addition with low latency and improved privacy, compression techniques are essential to reduce the computational complexity of complex DNNs. Moreover, it is desired that the compression technique should have minimal performance degradation.

Generally, compression techniques reduce the computational complexity of a DNN resulting in increase in number of errors made during speech to text conversion. Moreover, in case of varying resource budget which is often the case for edge devices, static compression methods (discussed in Chapter 2.1) are ineffective as the model does not have the tendency to adapt its architecture with resource budget. To this end, more sophisticated compression approaches are employed (see Chapter 2.2) enabling a DNN to adapt its architecture and function in any resource budget with acceptable performance.

To bridge these gaps, this thesis introduces dynamic compression of ASR models using effective layer dropping ($\mathcal{LD}$) techniques, which find efficient sub-networks inside a large DNN with optimum performance-computation trade-off. Typically, the DNNs suffer from significant performance degradation when layer dropping is applied which is plausible as some of the weights are zeroed, and the cost function is no longer minimized. Hence, the compression needs to be applied in such a way that the computational complexity of the model is reduced as well as the performance drop is minimal, giving ideal performance-computation trade-off.

1.2 Problem Statement

The successful deployment of DNNs on edge devices relies on reducing their computational and memory footprint, especially for time sensitive applications like Automatic Speech Recognition. Although, model compression offers a potential solution, current methods pose several key challenges:

1. **Performance degradation**: Compression techniques result in significant performance reduction, resulting in infeasible performance-computation trade-off.

2. **Limited compression ratio**: Existing techniques offer low compression ratio which is insufficient for operation on edge devices.

3. **Lack of generalizability**: Most of the compression techniques are downstream task dependent and require architectural modification for their operation.

These issues highlight the need for compression strategy that should preserve the model's performance while offering better performance-compression trade-off, and should be extended to any downstream task. Furthermore, the compression technique should enable a DNN to adapt its architecture with respect to device resources. For instance, if the compressed model is executed on a medium resourced device, the dynamic model uses most of the layers, whereas when it is executed over a resource-constrained edge device, the model selects the optimal layers to shrink its size and uses a compressed version to produce the final output without compromising final performance. Figure 1.3 depicts a dynamic architecture that has been compressed to perform in varying resource environment.

1.3 Contributions

To address the issues mentioned in section 1.2, this thesis summarizes following novel contributions:

1. We proposed an effective two-step framework capable of producing multiple lightweight and small models. The proposed semi-dynamic approach enables us to train static models of different depths 3x faster than normal training from scratch. The various sized lightweight models are conditioned for different edge devices, and easily deployable on such devices. Moreover, the approach is extendable to several downstream speech applications, and is explained in Chapter 4.

2. To explore the optimum performance-computation trade-off and quantify the effect of layer dropping on conformer based DNN, we performed exhaustive

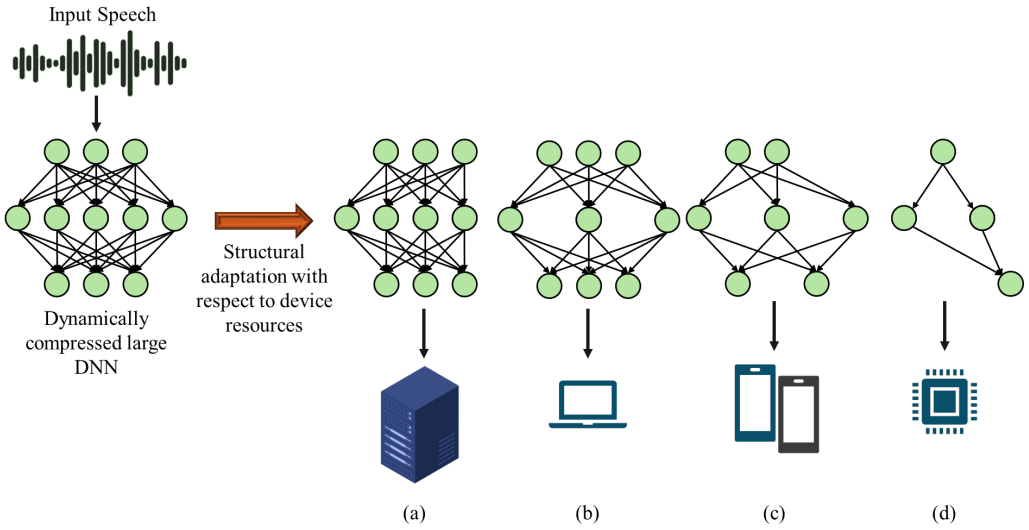

FIGURE 1.3: An example of neural architecture that underwent dynamic compression, and is capable of adapting the architecture as per available resource budget. (a), (b), (c), and (d) represent devices with different computational budget, the dynamic model adapts its architecture depending on level of available resources for each device.

experimental analysis using different layer dropping rates during training, and evaluated DNN's performance for different level of layer dropping as well as compared its performance against early exit, and equivalent sized static models trained from scratch. The method is explained in Chapter 5.

3. We extended the layer dropping to pre-trained speech foundation models [14, 32], and directed our work to alleviate the disadvantages of random layer dropping. To this end, we proposed an input driven layer dropping strategy that significantly improves the performance-computation trade-off on several speech and audio applications, including automatic speech recognition. We succeeded in reducing 50% of computational complexity with minimal affect on final performance, and can be further extended till 70% with acceptable performance. The method is discussed in Chapter 6.

4. To further improve the performance-computation trade-off and compression beyond 50%, we utilized knowledge distillation in conjunction with layer dropping to distill the expert knowledge of full model to the dynamic student model capable of working in varying resource environment. The method is detailed in Chapter 7.

1.4 Thesis Organization

The rest of the thesis is organized as follows: Chapter 2 summarizes the commonly used compression methods in literature with their limitations. Chapter 3 gives an overview of network architectures, datasets and metrics used in this thesis. Chapter 4 presents a two-step framework to efficiently train multiple lightweight models using less computational resources. Chapter 5 provides the detailed experimental

analysis of using layer dropping technique on conformer architecture. Chapter 6 introduces an input-driven layer dropping mechanism that selects the optimum subnetwork inside a DNN mainly for ASR, and can be utilized with other audio applications. Chapter 7 presents a hybrid approach using knowledge distillation and layer dropping to further improve the performance-computation trade-off. Chapter 8 concludes the thesis with possible future research directions.

Chapter 2

Literature

Speech processing, being a crucial aspect of Natural Language Processing (NLP), has gained significant importance in recent years. Substantial amount of work has been reported in the direction of developing large and DNNs targeting different speech applications like spoken language understanding, sound classification, and specifically automatic speech recognition. These deep neural architectures provide state-of-the-art performance at the cost of high computational complexity, restricting their functionability on edge devices with limited and varying resource budgets. To enable the deployment of such large DNNs, compression needs to be performed to significantly decrease the computational complexity of DNNs.

Compression can be classified as:

1. **Static**: where a model follows the same computational graph, and fits inside a definite resource budget. If the available resources change, the compression needs to be done again. Methods like pruning, quantization, and knowledge distillation falls in this category, and are detailed in section 2.1.

2. **Dynamic**: where a model has the ability to adapt its computational graph with the available resource budget. There is no need to re-do the compression. Methods like collaborative intelligence, early exit, and layer dropping belongs to this category, and are explained in section 2.2.

2.1 Static Compression Methods

2.1.1 Pruning

Pruning technique refers to the removal / deletion of parameters of a neural network based on a criteria or some heuristic that is defined a priori. It can be categorized as: **(i) unstructured pruning:** where separate neurons / connections are removed, **(ii) structured pruning:** where complete block of the neurons / connections is removed. Figure 2.1 depicts different kinds of pruning technique that are applied as a post-training step to reduce the computational complexity of trained DNN with minimal performance degradation.

Optical Brain Damage (OBD) [72] was one of the earliest paper that performs parameters pruning depending on the value of diagonal second order derivative of network parameters. They expand the objective function using taylor series and analytically predict the effect of perturbing the network parameters, selecting those parameters to delete that cause minimal increase in the objective function. They also highlighted that the process needs to be iterative, i.e. pruning first followed by re-training and so on, to recover the loss in performance. [33] validated the effectiveness of OBD using a fully connected neural network, whereas works like Optimal Brain Surgeon (OBS) [45] improve the OBD method by invalidating the assumption

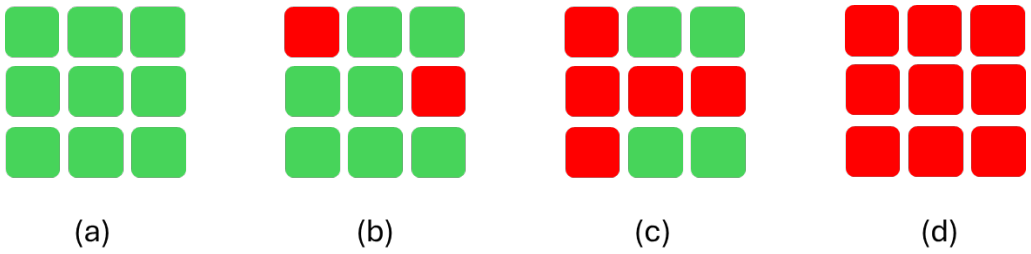

FIGURE 2.1: Illustration of different kinds of pruning. (a) depicts the layer without any pruning. (b) shows unstructured pruning where individual weights are zeroed out. (c) represents structured pruning where the complete row and column are zeroed out. (d) shows structured pruning as well where complete layer is skipped / zeroed. This is also applicable to convolutional maps and attention heads in transformers.

of diagonal Hessian matrix for each problem, and demonstrate that taking the non-diagonal Hessian matrix improves the overall performance of OBD method. Lately, [78] performed DNN pruning using OBD method which has the upper hand if retraining is not performed especially for high sparsity, and has comparable performance with retraining of DNN. Following that, [39] trained the network to identify the important connections which helps in removing the unimportant connections. This step is followed by retraining of the sparser network to compensate for the removed connections, and repeating the process.

Authors in [122] applied iterative unstructured pruning on gated recurrent neural network (GRNN) and long short term memory (LSTM) network using a predefined value that minimally affects the overall performance. The pruning step is followed by knowledge distillation where the model before pruning acts as a teacher network for the pruned and sparser model. Kim [66] proposed input pruning strategy which prunes the input tokens that have attention scores below a certain threshold. This results in decreased number of input tokens as the sequence passes through subsequent attention layers. [90] apply layer-wise pruning to reduce the model's size, and utilize knowledge distillation to compensate for the performance loss. Similarly, [68] used structured pruning, knowledge distillation and quantization techniques together to decrease the model's size and retain the performance. For ASR, [125] proposed Dynamic Sparsity Neural Networks (DSNN) which trains neural network of multiple sparsity inside one large neural network, and is capable of switching to any sparsity level at run-time. Similarly, [127] proposed a dynamic data pruning technique to achieve state-of-the-art performance with reduced dataset. Another study [92] applied three variations of structured pruning on pre-trained models to decrease the size of complete model using fixed and variable pruning factor.

2.1.2 Quantization

Quantization refers to the discrete number of small parts which is often assumed to be an integral multiple of a common quantity [36]. In terms of signal processing, it refers to the mapping of an analog signal into a set of discrete values for storage and bandwidth efficiency [29]. The difference in the mapping of actual analog value to the discrete value constitutes the quantization error, which decreases with the increase in number of discrete values (↑ discrete values, ↑ memory footprint,

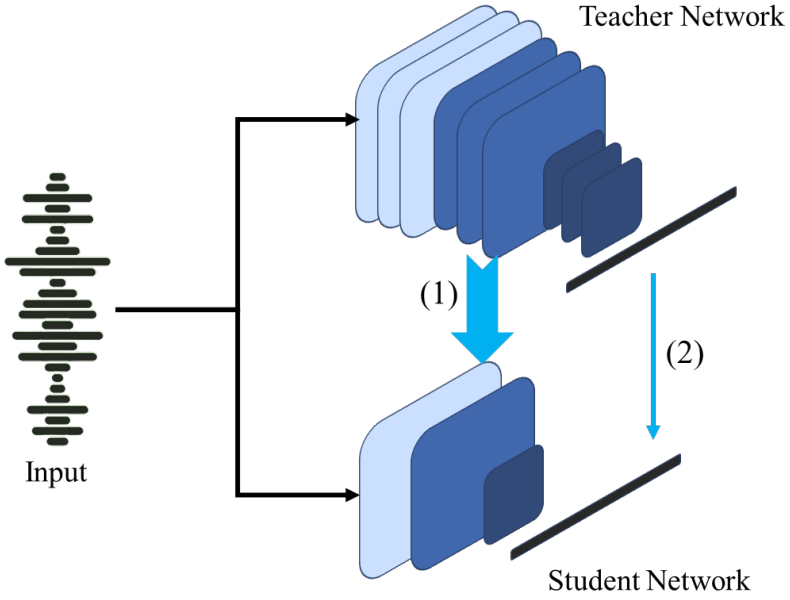

FIGURE 2.2: A typical teacher-student network for knowledge distillation. Light blue arrows show place of knowledge distillation. (1) refers to distilling knowledge from an intermediate location of teacher network to the student network. (2) refers to distilling knowledge from the final logits.

↓ quantization error). The number of available quantization levels determines the resolution of signal, i.e. 4 bit system can represent $2^4 = 16$ levels.

Current devices usually use 32-bit precision that represent each value in 2^{32} levels, which is extremely memory and computationally intensive. To tackle this, work has been done to reduce the precision of floating point representation and perform computation in reduced bits. [57] proposed a quantization scheme that enables inference using integer-only arithmetic for image related tasks. Works like [104, 19, 38, 98, 53, 22, 117, 21] proposed training neural networks using binary values (1 bit precision) or low-precision values. In speech processing, [1, 35, 79, 96, 82, 135] have worked on different quantization schemes, while mainly quantization is used in parallel with pruning [38, 68] and knowledge distillation [95, 96, 68, 135] to decrease the memory footprint. [131] proposed a mixed-precision quantization scheme for different parts of the network that achieved lossless quantization on Switchboard telephone speech [30] and AMI meeting transcription for speed-perturbation [10], i-Vector and learning hidden unit contribution (LHUC) based speaker adaptation with comparable performance with the state-of-the-art.

2.1.3 Knowledge Distillation

Knowledge Distillation (KD), originally proposed in [49], is one of the widely used compression methods where a large teacher model instills its rich representational ability into a tiny student model. KD enables the lightweight model to achieve comparable performance as the large model on a downstream task, which was previously achieved using ensemble of small models and is also cumbersome and memory intensive [7, 49].

Generally, the teacher is trained on substantial amount of data, and its knowledge is transferred to a tiny student model using a distillation loss like Kullback–Leibler

divergence [71], Jensen-shannon divergence [76], Wasserstein/Kantorovich distance [64, 110]. Existing works use cross-entropy loss [49, 25, 121, 89, 136] with temperature τ controlling the softness of distribution given by:

$$z = Softmax(e) = \frac{\exp(e_i/\tau)}{\sum_j \exp(e_j/\tau)} \tag{2.1}$$

where z are the class probabilities, e_i are class logits of i-th class, and τ is the temperature that is normally set to 1. However, [67] discuss that mean-squared error (MSE) loss can also be used for distilling information from a teacher model.

Besides loss function, some studies [60, 100] targeted compression of BERT architecture to accelerate inference time performance for speech downstream tasks. In addition to reduced latency, works like [106, 134, 133] investigated multiple-level knowledge distillation for end-to-end speech recognition models. Likewise, [27] performed distillation from an ensemble of CNN and LSTM based teacher networks, and elaborate the effectiveness of using multiple teacher models. In a similar fashion, [90, 140, 68] used knowledge distillation in conjunction with other compression methods, i.e. pruning, for different natural language processing tasks to recover the performance loss.

2.2 Dynamic Compression Methods

2.2.1 Offloading and Split-computing

Internet of Things (IoT) and edge-cloud continuum requires repeated execution of deep neural networks (DNNs) on limited resourced edge devices, heavily draining out their battery and limiting their life expectancy. To ease the computational load of edge devices, split-computing and offloading methods [118, 83] are used which transfer the demanding computational operation to the server over the volatile wireless channel. The server performs the computation and transmits back the output to edge device. The process is depicted in Figure 2.3.

Offloading raw-input do relieve the computational load of edge devices, however, it provides communication overhead as well as risks the data privacy [83, 5]. To this end, split-computing (also known as collaborative intelligence) is used that transmits intermediate representations of the network over the wireless channel, resolving the privacy concerns. Nonetheless, the intermediate representation can be influenced by erratic noise and interference due to volatile wireless channel, or can offer increased latency due to limited channel bandwidth. Moreover, these problems can aggravate if the edge device is in motion [118, 83]. Several works use collaborative intelligence [138, 59, 75, 105, 86, 54, 18, 5] to reduce the computational load of the edge device. The major portion of these studies propose different algorithms to find the best location in the network where the size of intermediate location is smaller than the raw-input size, ultimately leading to energy efficiency and negligible latency.

Some studies [2, 93, 9] reveal that the privacy concerns can also be handled in offloading and split-computing methods. [2, 93] utilize disentangled representation learning to achieve speech representations that filter out the unnecessary information from the audio sequence. However, the applicability of such methods is unfeasible on edge devices because of high computational resource requirement. [9] propose a method that partitions an audio into segments and strategically mask out

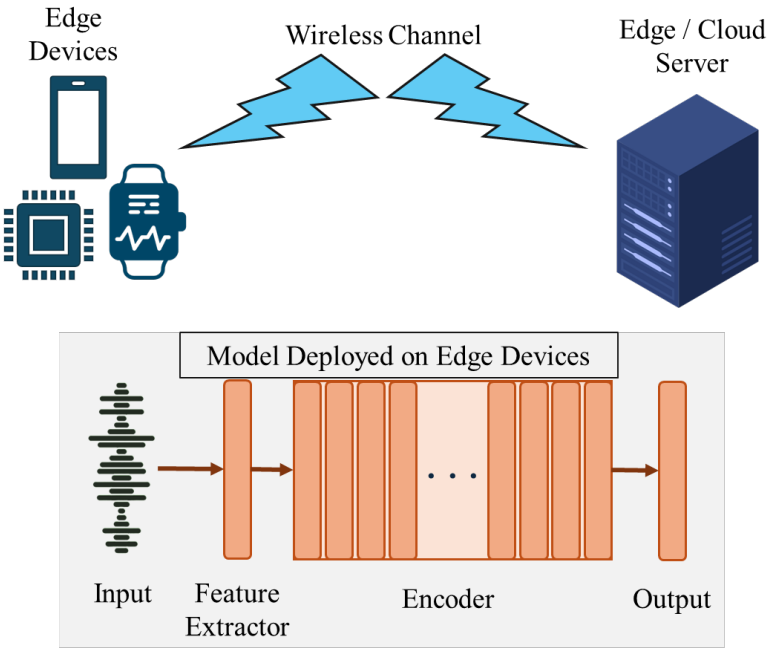

FIGURE 2.3: Offloading mechanism

majority of them, while hiding short-term details and capturing most of long-term dependencies.

2.2.2 Early Exit

The increased depth of the neural architectures yields state-of-the-art performance at the cost of high latency, high computational complexity, and enormous memory footprint. Early exit, first proposed in BranchyNet [108], augments auxiliary classifier branches to the network architecture allowing different samples to be processed till different network depths yielding output from different exit branches. This exiting through early branches reduces the runtime and provide energy efficiency. Branchynet was originally tested on well known LeNet [72], AlexNet [69], and ResNet [46] architectures, and is depicted in Figure 2.4.

With the increase in network's depth, the problem of vanishing gradient and overthinking arise that are addressed using early exit method [6]. Works like [65, 101] debate on the overthinking issue of large self-supervised models while proposing to mitigate it using early exits. Following that, [13, 74, 107] employed early exit with convolutional and transformer based networks for various speech related tasks, improving the network's latency.

In addition to reducing inference time, [101] debate that early exits alleviate the vanishing gradient and overfitting issues present in the static models by allowing them to exit through auxiliary branches. DeeBERT [129] and BERxiT [128] applied early exit to BERT architecture to reduce the inference time. [132] introduced an early exit based version of HuBERT, namely HuBERT-EE, to reduce latency and computational complexity using a confidence score for ASR. The confidence score is based on intermediate output's entropy, and the model terminates further computation if the entropy is less than a certain threshold. Likewise, [137] applied compared multiple compression methods including early exit, input downsampling etc. to compare

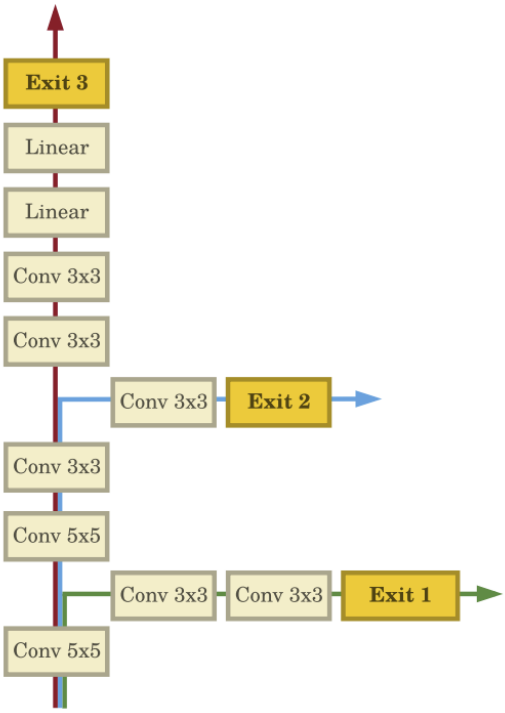

FIGURE 2.4: Early Exit method. Picture taken from original paper [108]. A simple BranchyNet with two branches added to the baseline (original) AlexNet. The first branch has two convolutional layers and the second branch has 1 convolutional layer. The "Exit" boxes denote the various exit points of BranchyNet. This figure shows the general structure of BranchyNet, where each branch consists of one or more layers followed by an exit point.

the effect on final performance for WavLM model. Similarly, [123] used early exit technique on a conformer based model to improve inference time latency.

2.2.3 Layer Drop

Layer drop ($\mathcal{LD}$), inspired from stochastic depth [52] in vision domain, has been widely used to transform static DNN into an adaptive (or a dynamic) DNN. $\mathcal{LD}$ is a kind of structured pruning that removes (or skips the execution of) layers from a DNN, hence, learning efficient sub-networks of different depths inside an over-parameterized DNN. It offers regularization effect during the training of DNN and improves the execution time by skipping execution of DNN layers [114, 52, 26]. $\mathcal{LD}$ is illustrated in Figure 2.5, and can be categorized as:

1. **Random Layer Dropping (RD)**: refers to the dropping of a structural unit / layer / module, or a part of it, depending on a random probability distribution [139, 26, 137].

2. **Input-driven dropping**: where the input of the network outlines the blueprint of dropped layers / modules [126, 112, 119, 15, 91, 28].

Recently, in vision domain, researchers have proposed different approaches to skip the processing of DNN layers using ResNet [46] architecture. BlockDrop [126] uses a reinforcement learning based approach to dynamically identify the layers of

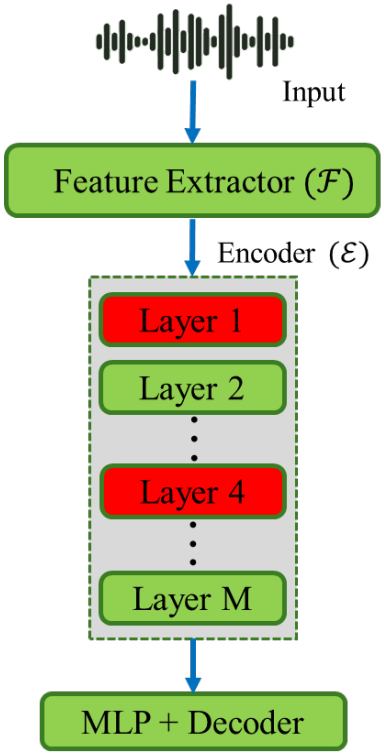

FIGURE 2.5: Layer Drop Method. Red color means that layers are dropped / skipped. Green color means layers are executed.

DNN to execute for current input image. SkipNet [119] adds a separate gating module to a group of layers, and used a hybrid of supervised and reinforcement learning to decide which layers to be executed. They also studied several variations of gating module, and their effect on final performance. [15] investigated input dependent dynamic filter selection strategy for CNNs to improve the model's generalizability. They used a separate Gater network that performs filter selection for the backbone CNN for each image. Similarly, [102] employed layer skipping for efficient action recognition by estimating the representations of individual frames, reducing the inference time computational cost.

In NLP domain, for transformer based pre-trained networks, [139] proposed a gating structure to skip the sub-layers (feed forward network or self-attention) of each transformer layer. This gating structure is governed with a progressive layer dropping strategy that gives a higher dropping probability to the earlier layers and lower dropping probability to the last layers of the network, and evaluated on BERT, RoBERTa, and XLNet architectures. Following that, [99] manually dropped different combinations of pretrained BERT architecture to evaluate its effect on downstream GLUE tasks [115]. For ASR, [137] used random layer dropping with a dropping probability of 0.5 to evaluate the wavlm [14] architecture, along with early exit and input downsampling method. [91] used an input-driven dropping strategy with transformer, however, they utilized gumbel-softmax trick to make the binary gates differentiable. [130] propose to learn the importance of each layer in the network to find the efficient sub-network in DNN. [28] apply $\mathcal{LD}$ on different structural units of a wav2vec 2.0 foundation model, whereas [24] applied $\mathcal{LD}$ on Llama models [109] with speculative decoding in a similar fashion as [139].

Chapter 3

Experimental Framework: Datasets, Neural Architectures, and Evaluation Metrics

---

This chapter provides a comprehensive description of the neural networks and architectures employed, the datasets used for training and inference, and the evaluation metrics used to evaluate the performance of each model.

3.1 Datasets

We evaluated the proposed methods for several audio and speech downstream applications, using the following datasets:

1. **LibriSpeech** [87] is the most common dataset for ASR downstream task. It consists of $\approx$ 1000 hours of read-aloud English audio-books where each audio file is sampled at 16KHz. The dataset is partitioned into training, validation and test splits of 960 hours, 20 hours, and 20 hours. The dataset is publicly available under “creative common attribution 4.0" license.

2. **TED-LIUM v3** [48] is another publicly available dataset for ASR downstream task, and is available under Attribution-NonCommercial-NoDerivatives 3.0 license. It comprises of 452 hours of Technology, Entertainment, and Design (TED) talks in english language where each audio file is sampled at 16KHz.

3. **Environmental Sound Classification (ESC-50)** [94] is a benchmark for sound classification, and consists 2000 labeled recordings of 50 dintinct sound events with 40 recordings per class. The sound events are loosely classified into “animal, human, natural, urban, and domestic sounds" categories.

4. **Fluent Speech Commands (FSC)** [81] is a dataset for spoken language understanding (SLU) task, and comprises of 248 utterances that map to 31 unique intents. The data was collected through crowd-sourcing, and required each one of 97 speakers to repeat the phrase twice. It is available under Public license only for academic purpose.

5. **Google Speech Commands (GSC)** [120] is a benchmark for keyword spotting task, and consists of spoken commands of $\approx$1 second duration. There are 35 speech commands spoken by 2618 speakers, and the available audio recordings are sampled at 16KHz. The dataset is available under “creative common attribution 4.0" license.

TABLE 3.1: Dataset Statistics. Total Duration in hours.

Dataset	Application	# of Utterances	Total duration	# of speakers
LibriSpeech	Automatic Speech Recognition	272,428	982.1	1,166
TED-LIUM v3	Automatic Speech Recognition	270,323	452	2,028
Fluent Speech Commands	Spoken Language Understanding	30,043	19.0	97
Google Speech Commands	Keyword Spotting	105,829	16.4	2,618
ESC-50	Sound Classification	2,000	2.78	-
IEMOCAP	Emotion Recognition	7,532	≈12	10

6. **IEMOCAP** [8] contains audiovisual data of 12 hours where actors performed scripted scenarios in dyadic sessions. In this thesis, only the audio modality is used to recognize the elicited emotion in the utterance. The dataset is available only for academic purpose.

The main characteristics of the datasets are listed in Table 3.1

3.2 Neural Architectures

This section describes the deep neural architectures that have been used in this thesis.

3.2.1 Conformer - Convolution-augmented Transformer for Speech Recognition

Conformer [37] is a well-known model that is mainly utilized for ASR. Prior work, like ContextNet [40], employed a mixture of CNN and RNN with squeeze-and-excitation networks [51] in each residual block to capture local features and long range context. However, this approach partially captures the global context because it applies global average pooling once over the whole input sequence, and requires more sophisticated elements to effectively capture long-range contexts. [124] proposed architecture with separate branches for attention mechanism and convolutional layers and concatenated the output of each branch, which leads to performance improvement in machine translation task.

The conformer architecture effectively captures the local fine-grained features and context-based global interactions by combining the strengths of CNNs and transformers. The conformer module consists of four sub-modules: first feed-forward

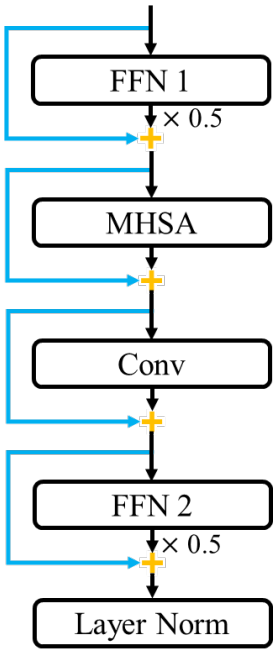

FIGURE 3.1: Architecture of a Conformer module.

layer, self-attention layer, convolutional layer, and second feed-forward layer, and a layer normalization at very end of each module. Inspired from [124, 80], the self-attention and convolutional layers are sandwiched between two feed-forward layers with an output weight of 0.5, as illustrated in Figure 3.1. Mathematically, the conformer can be represented as:

$$h_1^i = x^i + \frac{1}{2} f_{\text{FFN1}}^i(x^i) \tag{3.1}$$

$$h_2^i = h_1^i + f_{\text{SA}}^i(h_1^i) \tag{3.2}$$

$$h_3^i = h_2^i + f_{\text{Conv}}^i(h_2^i) \tag{3.3}$$

$$y^i = \mathcal{LN}^i\left(h_3^i + \frac{1}{2} f_{\text{FFN2}}^i(h_3^i)\right) \tag{3.4}$$

where x^i and y^i are the input and output of the conformer module, and h_1^i, h_2^i, and h_3^i are the intermediate representations. Each of these layers further contains many layers which are available in respective paper [37].

The conformer model is available in three versions of 10M, 30M, and 118M parameters depending on different number of attention heads, model's hidden dimension, and network depth. The original model also uses a single layer of LSTM as a decoder in each version along with a possibility of a 3-layered LSTM language model with a width of 4096.

Employed Conformer Architecture

In Chapters 4, 5, and 7, a modified in-house version of original conformer was used where:

1. The feature extractor was very shallow with two 1D convolution layers, without any activation and normalization layer in-between the convolution layers.

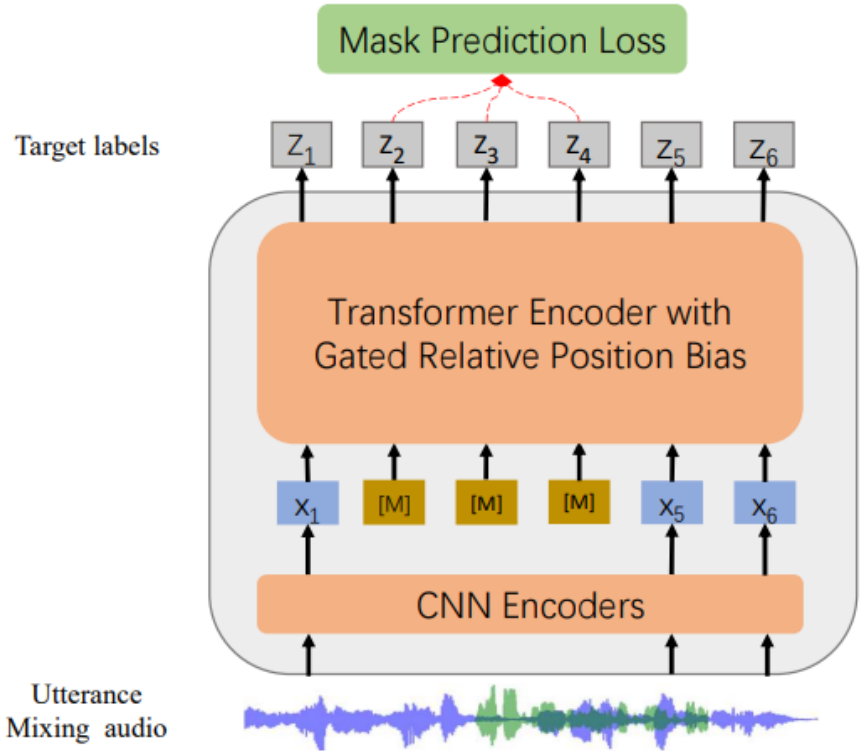

FIGURE 3.2: WavLM Architecture. Picture taken from original paper [14].

This limits the feature extractor to learn rich-semantic representation from the data, making the learning more challenging.

2. The LSTM layer at the top of conformer's encoder block was replaced with a simple linear layer of equivalent size. As a result, the long-term dependencies and the sequence context were not efficiently captured.

These modifications increased the complexity of downstream task along with challenging the development of dynamic compression framework.

3.2.2 WavLM

WavLM [14] is a speech foundation model (SFM) that is pre-trained in a self-supervised way to solve full-stack downstream speech tasks. WavLM learns universal speech representations from enormous 94k hours of unlabelled audio data which comprises of 60k hours of Libri-Light, 10k hours of GigaSpeech [12] and 24k hours of Voxpopuli [116] datasets. As a consequence, SFMs like WavLM, is easily employed for speech processing tasks even when the supervised data is scarce.

The pre-training step involves masked speech denoising and prediction framework where a few of the inputs are simulated noisy / overlapped speech with masks, and the target is to predict the pseudo-label of the original speech on the masked region. The training approach is similar to the one used in training HuBERT model [50], enabling the model to be used for full-stack speech processing tasks. In addition, WavLM uses gated relative position bias (grep) [16] instead of convolutional relative positional embedding which is used in wav2vec 2.0 and HuBERT, allowing the gates to adaptively condition the relative bias to the current speech content.

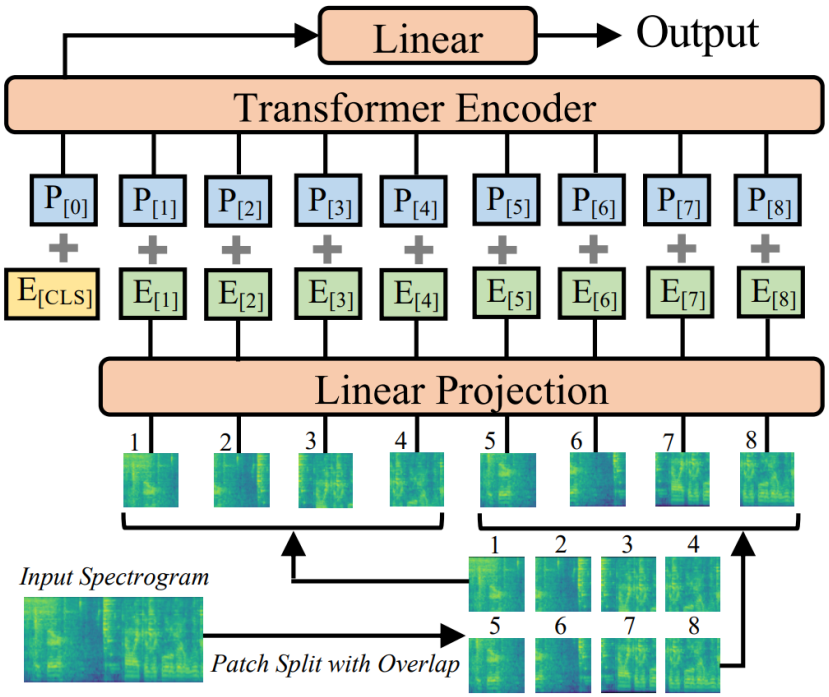

FIGURE 3.3: Audio Spectrogram Transformer Architecture. Picture taken from original paper [32].

The WavLM model consists of a convolutional front-end and a transformer based encoder backbone. The convolutional front-end contains 7 temporal convolution layers with 512 channels, strides of $\{5,2,2,2,2,2,2\}$, and kernel widths of $\{10,3,3,3,3,2,2\}$, where each convolution layer is followed by layer norm and GELU activations. The convolutional front-end's output $\mathbf{x}$ is masked, and fed to the transformer encoder which has a convolution-based relative positional embedding layer with a kernel size of 128 and 16 groups. In addition, gated relative position bias is added to improve the model's performance which is computed based on the offset between keys and queries of the attention mechanism.

For each utterance $\mathbf{u}$ and its simulated version $\hat{\mathbf{u}}$, psuedo-labels $\mathbf{z}$ are generated using last iteration network as in HuBERT using the latent representation or k-means clustering center on MFCC. This results in the hidden state $\mathbf{h}_t^L$ by feeding $\hat{\mathbf{u}}$ to the current network and optimizing the mask prediction loss function given below:

$$\mathcal{L} = \sum_{l \in K} \sum_{t \in M} \log p\left(z_t | \mathbf{h}_t^L\right) \tag{3.5}$$

3.2.3 Audio Spectrogram Transformer

Convolutional Neural Networks (CNNs) were the de facto choice for end-to-end classification networks because of their ability to learn rich semantic local features. Since the arrival of transformers [111] which capture long-range and contextual dependencies, a mixture of CNNs and transformers have been utilized for end-to-end audio classification tasks. Audio Spectrogram Transformer (AST) [32] was proposed to evaluate whether the CNN-transformer hybrid model is necessary or a purely attention based model is sufficient for good performance.

AST is a convolution-free attention-based model developed solely for audio classification tasks, that is applied directly on the mel-spectrograms and captures long-range contextual features. Figure 3.3 depicts the architecture of the AST model that converts the audio waveform of t seconds and computes 128-d log-mel spectrograms using a Hamming window of 25ms with a hop-length of 10ms. This transforms the input audio of duration t seconds into a log-mel spectrograms of $128 \times 100t$, that are used as an input to the AST model. The spectrogram is split into N 16×16 patches with an overlap of 6 across time and frequency dimensions, where $N = 12\,[(100t - 16)/10]$ is the number of patches and the effective input sequence length for the Transformer. Each patch is flattened and converted into a $1D$ patch embedding of size 768 and a $<$ CLS $>$ token is appended at the start of embedding. A learnable positional embedding of size 768 is added to the patch embedding to capture the spatial structure of the $2D$ spectrogram. The resulting embedding is forwarded to the transformer encoder, consisting of 12 layers, 12 attention heads and embedding dimension of 768. The output of $<$ CLS $>$ token represents the audio spectrogram representation, which is linearly projected followed by sigmoid activation providing the logits for classification.

The drawback of transformer only models, such as AST, is that they require substantial amount of data for training [23], which is hardly available for audio classification. To address this, AST utilizes cross-modal transfer learning from Vision Transformer (ViT) [23] which is pretrained on ImageNet dataset [20]. As spatial structure of spectrogram resembles with images, AST can use the expert knowledge of ViT that is trained on large-scale image dataset. This cross-modal transfer learning is accomplished with couple of modifications:

1. Input dimension mismatch, which is sorted by averaging the weights of each input channel of ViT.

2. Fixed input shape of ViT which is different from the variable size of audio waveform in AST. The transformers are capable of supporting variable length sequences and can be directly transferred from ViT to AST as well as the positional embedding of $<$ CLS $>$ token. However, the positional embedding of rest of the sequence needs to be carefully transferred using cut and bilinear interpolate method. This method ensures that the dimensions of positional embedding are aligned for cross-modal knowledge transfer.

3. Replacing the ViT's classification layer with a new classification layer

Table 3.2 summarizes the characteristics of conformer, wavlm and ASR architectures.

3.3 Metrics

The choice of evaluation metric is essential to correctly quantify the network's performance, and varies with respect to the downstream tasks. For Automatic Speech Recognition, word error rate (WER) or character error rate (CER) metrics are used which quantify the number of errors made in the predicted text against the available ground truth. For applications like sound classification and intent classification, metrics like accuracy or mean averaged precision are used which measure the correctness of model's predictions. In our published works, we have used **WER** and **accuracy** for different applications, and their details are given below:

TABLE 3.2: Comparing Conformer, WavLM and AST architectures with original configurations.

	Conformer	**WavLM**	**Audio Spectrogram Transformer**
Convolutional Frontend	✓ 2 layers	✓ 7 layers	×
Pre-training Method	Supervised Learning. Generally trained from scratch	Self-Supervised Learning. using a joint objective of masked speech denoising and psuedo-label prediction in masked region.	Supervised Transfer Learning. Initialized with pretrained ViT and finetuned for audio applications.
Parameters	Small - 10.3M Medium - 30.7M Large - 118.8M	Base+ - 94.7M Large - 316.6M	Base - 86M Large - 307M
Data Requirement for downstream tasks	High	Low to medium	Low to medium
Applications	Specifically Automatic Speech Recognition.	Full-stack speech tasks. Automatic Speech Recognition Speech Diarization Emotion Recognition	Audio Event Detection Audio Classification Audio/Music Recommendation systems.
Compression for edge devices	Low	High	High

1. **Word Error Rate (WER)**: compares the aligned predicted text from output of the model to the ground truth transcription. It estimates the quality of transcription by adding the number of substitutions, deletions and insertions in predicted text and dividing by the total number of words in reference transcription. Mathematically, it can be expressed as:

$$\text{WER} = \frac{S + D + I}{Total} \tag{3.6}$$

where S, D, I, and N represent substitutions, deletions, insertions, and total number of words in reference transcription.

It is a widely used metric for Automatic Speech Recognition. Generally, lower WER value is desired as it depicts that the model generates better transcription and made less number of errors.

2. **Accuracy**: is a measure of how correctly the model predicts the outcome. Mathematically, it can be expressed as:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.7}$$

where TP, TN, FP, and FN are true positives, true negatives, false positives, and false negatives respectively.

It is commonly used for audio classification tasks, like sound classification, intent classification, emotion recognition etc. Generally, higher accuracy is desired as it shows that the model can correctly predicts the outcome and is reliable.

Chapter 4

Improving Static and Lightweight ASR Models

This Chapter is extracted from our manuscript published at InterSpeech, 2025 [43]. It presents a semi-dynamic two-step framework addressing the limitation of compressing large models (using static compression methods) to fit inside a fixed computational budget with minimal performance degradation.

The success of AlexNet [69] architecture on image recognition problem, redirected the focus of machine learning community to create large and deep neural networks. From that moment, many deep architectures like VGGNet [103], GoogLeNet (Inception network) [105], ResNet [46], squeeze and excitation network (SENet) [51] etc. are trained that set the new SOTA performance on image recognition. This trend exploded with the arrival of transformers [111], extending the idea of developing deep architectures for all domains and downstream applications, including speech recognition featuring architectures like wav2vec 2.0 [4], and wavlm [14]. Such models have been pre-trained on thousands of hours of audio data, enabling them to perform effectively in many audio downstream applications. However, finetuning such large models is cumbersome, and requires substantial amount of energy and high-end computational equipment that is not accessible for everyone. In such scenario, lightweight models are preferred which display low computational cost and equivalent performance.

4.1 Objectives

Recent paradigms like internet of things (IoT), tiny machine learning (tiny ML) and green ML, targets developing lightweight and resource-friendly architectures that provide favorable performance in limited resource settings. These paradigms are in contrast with the trend of building large and deep architectures with billions of parameters that produce tons of carbon dioxide emission, consume staggering amount of energy and other resources. In addition, such models are practically unsuitable for deployment on end-user devices due to high resource requirement.

The downside of training smaller, lightweight and device-conditioned architectures aligning with the tiny and green ML paradigms, also requires substantial training time cost to converge with subpar performance. Therefore, it is desired to build smaller architectures capable of achieving favorable performance as well as requiring reduced computational and memory load.

In this regard, we propose an effective approach that exploits the knowledge of large reference model to obtain tiny models capable of performing favorably while

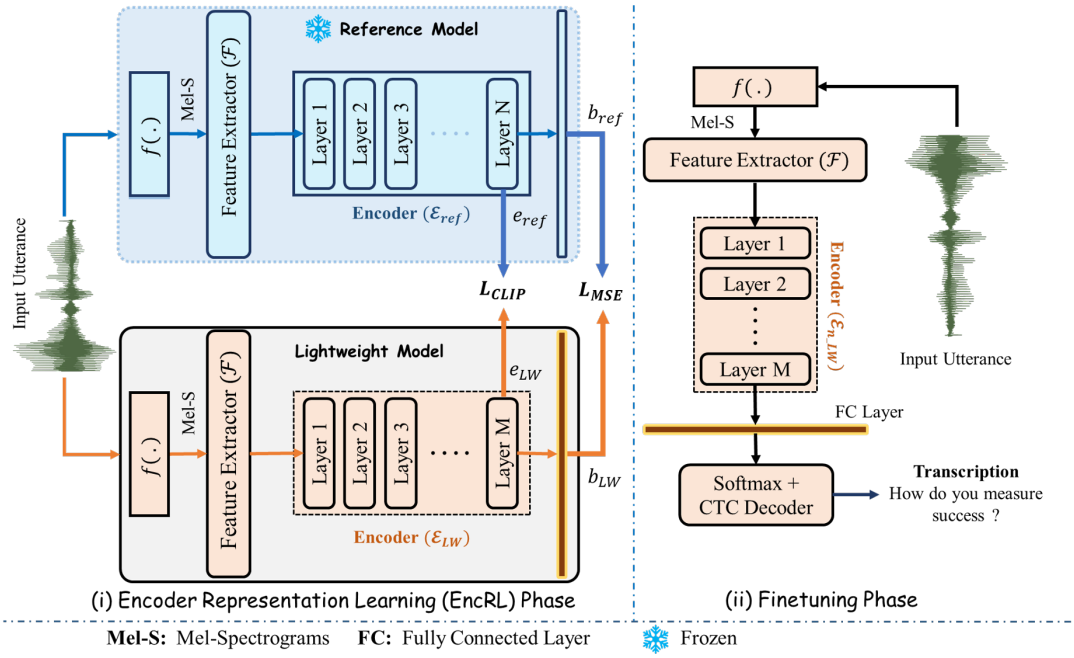

FIGURE 4.1: Here we illustrate the overall framework of the proposed method. (i) In the feature learning phase, the input utterance is fed to a large ASR model (reference model) to extract the features e_{ref} and b_{ref}. Afterwards, the same input is passed through the smaller model to obtain features e_{LW} and b_{LW}. To learn the knowledge of reference model, MSE loss is used between the outputs (b_{ref} and b_{LW}) of the classifiers of the reference and the small model. Symmetric cross-entropy loss is used on the features e^{N}_{ref} and e^{M}_{LW} to align the feature spaces. (ii) After transferring the knowledge to the encoder of light-weight model, CTC decoder is integrated and the model is finetuned to transcribe the input speech.

requiring low computational resources. To summarize, our contributions are as below.

- We propose a two-step based approach to train small ASR models that achieve promising results on ASR task.
- The introduced approach enables us to train several small and tiny models requiring less compute resources while providing promising performance compared to equivalent small models.
- We perform extensive experiments on two challenging ASR benchmarks demonstrating the effectiveness of proposed approach.

4.2 Methodology

In this work [43], we proposed a two-step feature learning based framework to achieve multiple small sized static models that require less training time as compared to the models trained from scratch, and provide significant performance improvements. The proposed method requires (i) *One time Encoder Representation Learning* (**EncRL**) to mimic the enriched representation of the large Reference model $\mathcal{M}_{ref}$, (ii) Finetuning for limited number of epochs.

Given a dataset $\mathcal{D} = \{(a_d^i, t_d^i)\}_{i=1}^K$ where a_d^i and t_d^i are the corresponding audio-transcription pair of the i^{th} instance with K total instances, we train a reference model $\mathcal{M}_{ref}$ consisting of N encoder blocks for Z number of epochs. This trained model provides the *baseline results* as well as serves as the reference model for representation learning phase.

4.2.1 Encoder's Representation Learning (EncRL)

In this phase, our objective is to enable the light-weight model $\mathcal{M}_{LW}$ to learn the information from the deep reference model $\mathcal{M}_{ref}$ that $\mathcal{M}_{ref}$ has learned by training on a large amount of data for a greater number of epochs Z. To this end, reference model $\mathcal{M}_{ref}$ and light-weight model $\mathcal{M}_{LW}$ are created having N and M number of layers, respectively, where $M \leq \frac{N}{2}$. Subsequently, we take the output features e_{LW} and e_{ref} of the last layer of the corresponding encoders and utilize the symmetric cross-entropy $\mathcal{L}_{CLIP}$ loss to align the feature representations as shown in Figure 4.1. The $\mathcal{L}_{CLIP}$ loss minimizes the distance between the feature representations (e_{LW} and e_{ref}) of the same samples while maximizing for the different samples. Similarly, to enforce the light-weight model $\mathcal{M}_{LW}$ to produce the similar output as the reference model $\mathcal{M}_{ref}$ generates, we utilize mean squared error ($\mathcal{L}_{MSE}$) loss on the feature embeddings b_{LW} and b_{ref} of the two models. This loss combination will encourage the light-weight model $\mathcal{M}_{LW}$ to learn rich semantic information from the model $\mathcal{M}_{ref}$. The total loss is given as:

$$\mathcal{L}_{EncRL} = \mathcal{L}_{CLIP} + \mathcal{L}_{MSE} \tag{4.1}$$

$$\mathcal{L}_{CLIP} = \max_{i \neq j}\Big(\min_{i=j}\Big(\sum_{i,j}^{B} e_{ref}^i \odot e_{LW}^j\Big)\Big) \tag{4.2}$$

$$\mathcal{L}_{MSE} = \frac{1}{B}\sum_{i}^{B}(b_{ref}^i - b_{LW}^i)^2 \tag{4.3}$$

where $\odot$ represents element-wise multiplication.

During this phase, the reference model $\mathcal{M}_{ref}$ is kept frozen and the transcription decoder is not utilized. We perform one-time feature representation learning of the light-weight model $\mathcal{M}_{LW}$ by training for $\frac{2\times Z}{3}$ epochs, enabling it to learn the enriched representation of $\mathcal{M}_{ref}$. Afterwards, we finetune the model $\mathcal{M}_{LW}$ for a few epochs as explained in the next section.

4.2.2 Finetuning phase

After the feature representation learning phase, we integrate the Connectionist Temporal Classification (CTC) decoder in the light-weight model $\mathcal{M}_{LW}$. Then, we initialize the model $\mathcal{M}_{LW}$ with the weights of the representation learning phase. We feed the input audio utterances to the model $\mathcal{M}_{LW}$ and generate the output transcriptions. We finetune the model for $Z/3$ number of epochs. During finetuning, we utilize the connectionist temporal classification loss to measure the performance of the model. Similar to the model $\mathcal{M}_{LW}$, we further finetune the tiny models $\mathcal{M}_{n\_LW}$ having number of layers less than M. Specifically, we finetune the model $\mathcal{M}_{n\_LW}$ having encoder layers n where $n \in [2, 4, ..., M]$. This enables the smaller models to perform favorably without requiring training for a large number of epochs.

For instance, given a mini-batch B in the dataset $\mathcal{D}$, the model processes the audio inputs a^i producing the corresponding embedding b^i, followed by a *Softmax* layer

that converts the raw logits into the respective token probabilities p^i as illustrated in Figure 4.1. The tokens are forwarded to a built-in CTC decoder transforming the incoming data into the final transcription t^i_{pred}. Finally, the model's transcription is optimized by quantification of CTC loss by comparing against the ground-truth transcription.

$$\begin{aligned} t^i_{pred} &= CTC\_Decoder\ \left(Softmax(b^i)\right) \\ \mathcal{L}_{ctc} &= f_{CTC}(t^i - t^i_{pred}) \end{aligned} \tag{4.4}$$

4.3 Implementation Details

The training is performed using NVIDIA $A40$ GPU with a batch size of 64 using AdamW optimizer with $L2$ normalization of $1e^{-6}$ and 10000 warmup iterations followed by exponentially decreasing the learning rate till the end of training. For each audio input, mel-spectrograms (using 80 mel-filterbanks) are computed for $20ms$ window length with a hop-length of $10ms$ along with SentencePiece byte-pair-encoding (BPE-256) tokenizer [70]. To suppress the over-fitting issue, we applied *SpecAug* augmentation [88] with frequency and time masking of 27 and 80 samples. **Architecture:** We adapted open-sourced Conformer (small) architecture with feed-forward dimension of 256 and 12 encoder layers for the reference model. Consequently, the number of encoder layers M for light-weight model are 6. Furthermore, we replaced the 3-layered LSTM based decoder with a linear projection layer, whose output is forwarded to the CTC decoder for transcription.

4.4 Results

4.4.1 Comparison with small models:

We compare the proposed framework with existing light-weight models, as well as smaller models that are trained for Z number of epochs. Table 4.1 reveals that our approach achieves superior performance as compared to the other smaller models for various model sizes (different number of encoder layers). For instance, our method obtains WER of 6.27% and 14.68% for the model having 6 and 2 encoder layers on LibriSpeech test-clean split, whereas the smaller models that are trained for greater epochs achieve WER of 8.61% and 15.67% respectively. Similarly, our approach achieves reasonable improvement in WER score of 3.5% and 0.36% for model having 6 and 4 encoder layers on LibriSpeech test-other split. Likewise, on Tedlium (v3) test split, our approach obtains absolute improvement of 2.1% and 26.44% in WER metric as compared to model trained from scratch.

In addition, to train W models of various sizes, our method requires $(W \times Z)/3$ training epochs compared to the other smaller models that require $W \times Z$ epochs, achieving $3\times$ faster training and better performance. In short, our approach offers flexibility to create several light-weight and small-sized models requiring considerably less training time and improved performance.

4.4.2 Why EncRL is required?

We performed the finetuning of smaller model having 6 encoder layers while initializing the weights from the final states of $\mathcal{M}_{ref}$. We observe that despite of intializing the weights from $\mathcal{M}_{ref}$ model, the light-weight model struggles to provide better

TABLE 4.1: WER evaluated on test-clean and test-other splits of LibriSpeech dataset. First row indicates the large ASR model trained for n number of epochs. * represents that the smaller model is trained for n number of epochs similar to large model. † represents the results of a Conformer model from [123] with twice number of attention heads and doubled feed-forward dimension in conformer submodules. × represents not reported.

Model	Epochs	# of Encoder Layers	Params (M)	WER (%)	
				clean	other
Conformer (ref)	150	12	18.60	5.60	14.38
Conformer†	×		31.601	6.5	17.7
Conformer*	150	6	9.462	8.61	20.97
Conformer†	×		15.963	7.6	20.0
Conformer (ours)	50		9.462	**6.27**	**17.47**
Conformer*	150	4	6.416	8.91	22.18
Conformer†	×		10.750	9.8	24.3
Conformer (ours)	50		6.416	**7.95**	**21.82**
Conformer*	150	2	3.370	15.57	**32.57**
Conformer†	×		5.537	17.6	36.1
Conformer (ours)	50		3.370	**14.68**	33.70

performance and achieves a WER of 7.41%. Whereas our approach provides better performance by achieving WER of 6.27%.

4.4.3 Vitality of Finetuning:

To investigate the requirement of finetuning, we integrated the CTC decoder to the light-weight model obtained after representation learning. We then evaluated the model on LibriSpeech test-clean split. It was observed that the model failed to recognize the audio signal resulting in extremely high WER of ($\approx$ 95%) highlighting the need of finetuning phase.

4.4.4 Quantifying the efficiency of Loss functions

We employed various loss functions during the representation learning phase including CLIP, MAE, MSE, and their combinations. Table 4.3 enlists the obtained WER for each loss function, highlighting that combination of CLIP and MSE loss provide the rich feature representations which enable the model to obtain best performance after finetuning phase achieving WER score of 6.27%.

4.4.5 Do we need separate EncRL for various sizes of model?

To answer this question, we performed *EncRL* for a model having 4 encoder layers followed by finetuning for $Z/3$ epochs, and compared it with the finetuned model that is initialized with the last 4 layers of encoder's state dictionary of *EncRL* with 6 layers. We observed slight improvement of 0.53% in WER when we separately

TABLE 4.2: WER evaluated on test split of TEDLIUM-v3 dataset. First row indicates the large ASR model trained for n number of epochs. * represents that the smaller model is trained for n number of epochs similar to large model. † represents the results of a Conformer model from [123] with twice number of attention heads and doubled feed-forward dimension in conformer submodules.

Model	# of Encoder Layers	Params (M)	WER (%)
Conformer (ref)	12	18.60	15.28
Conformer†		31.601	16.4
Conformer*	6	9.462	21.82
Conformer†		15.963	25.5
Conformer (ours)		9.462	**19.72**
Conformer*	4	6.416	49.30
Conformer†		10.750	35.4
Conformer (ours)		6.416	**22.86**

TABLE 4.3: Evaluating the effect of different Loss functions used during Encoder's Representation Learning phase. # of encoder layers used in these experiments is 6.

Loss Function	WER (%)
CLIP only	9.06
MAE only	6.40
MSE only	6.36
CLIP + MAE	6.42
CLIP + MSE	**6.27**

perform encoder representation learning for 4 encoder layers demonstrating that the repeating encoder representation learning step does not provide significant improvement in performance but consumes twice as much computational cost.

4.4.6 Pruning of reference model

We applied pruning technique on the reference model to compare its performance against the proposed framework on LibriSpeech test-clean split. To prune the model, we employed two different schemes: (i) pruning conformer module completely by a given amount, (ii) pruning other conformer sub-modules except the convolutional sub-module, and the results are given in Table 4.4. For the *former* case, we observed that pruning only 5% of weights significantly degrades the performance, scoring 13.64% WER. The performance reduces further if we increase the pruning amount by 2.5%, and totally collapses at pruning amount of 10% giving a WER of 49.44%. For the *latter* case, the model shows WER of 10.51% for 7.5% pruned weights which is 13.16% less than we achieved in the former pruning method. Similarly, the WER substantially increases by 11.69% and 23% for pruning amount of 15% and 22.5% highlighting the limitation of pruning to maintain good performance. However,

TABLE 4.4: Comparing WER of small models generated using our framework against Pruning a large model in two different ways.

Pruning Conformer module including convolutional sub-module		**Pruning Conformer module without convolutional sub-module**		**Our Approach**	
Pruning Amount (in %)	**WER (in %)**	**Pruning Amount (in %)**	**WER (in %)**	**Equivalent Pruning Amount (in %)**	**WER (in %)**
5	13.64	7.5	10.51	**50**	**6.27**
7.5	23.67	15	22.20	**66.7**	**7.95**
10	49.44	22.5	33.51	**83.33**	**14.68**

our framework significantly outperforms the pruning schemes, providing a mere WER of 6.27%, 7.95% and 14.68% for equivalent pruning amount of 50%, 66.7%, and 83.33%, highlighting the efficacy of the framework.

Table 4.4 also signifies that convolutional sub-module of the conformer is essential for sustainable overall performance, and any efforts to prune it would significantly affect the intermediate representations, hence, producing poor performance. On the other hand, pruning feed-forward and self-attention sub-modules is more effective and have reduced impact on the final performance.

Chapter 5

Experimental Analysis on Random Layer Dropping

This Chapter is derived from our manuscript published at EUSIPCO, 2024 [41]. It performs an exhaustive experimental analysis using different dropping probability values for $\mathcal{LD}$ during training, and quantize the model's performance for different levels of depth.

To deploy large models on devices with limited and varying resources, we need to instill structural adaptability in the model that will ensure the operation would be performed inside the available resource budget. This structural adaptability is introduced using Layer Dropping method that is described in section 2.2.3 and further in this chapter in section 5.2.2.

5.1 Objectives

This work targets an exhaustive experimental analysis of layer dropping $\mathcal{LD}$ method technique training and inference phases. In addition, several inference time strategies are investigated while quantifying their respective performance-computation trade-off, as well as comparing $\mathcal{LD}$ with early exits and small conformer architectures trained from scratch. Based on the performance-computation trade-off, the ideal dropping rate is presented [41].

5.2 Methodology

5.2.1 Overall Architecture

Figure 5.1 illustrates the overall architecture of the *conformer* model with modified gating mechanism. The model takes Mel Frequency Cepstral Coefficients (MFCCs) as an input which are extracted from the audio waveform by utilizing 512-points short time Fourier transform (STFT) on 320 samples (20 ms) chunk with a hop-length of 160 samples (10 ms). The model comprises of two 1d convolutional layers followed by 12 conformer encoder blocks and a linear layer serving as a decoder. Each conformer module has a gate g whose binary value follows the output of Bernoulli distribution (BD) for a specified dropping probability (p_{in}), governing the input flow through the network. The probability mass function of BD is given by:

$$f(k, p_{in}) = p_{in}^{k}(1 - p_{in})^{1-k} \; for \; k \in \{0, 1\} \tag{5.1}$$

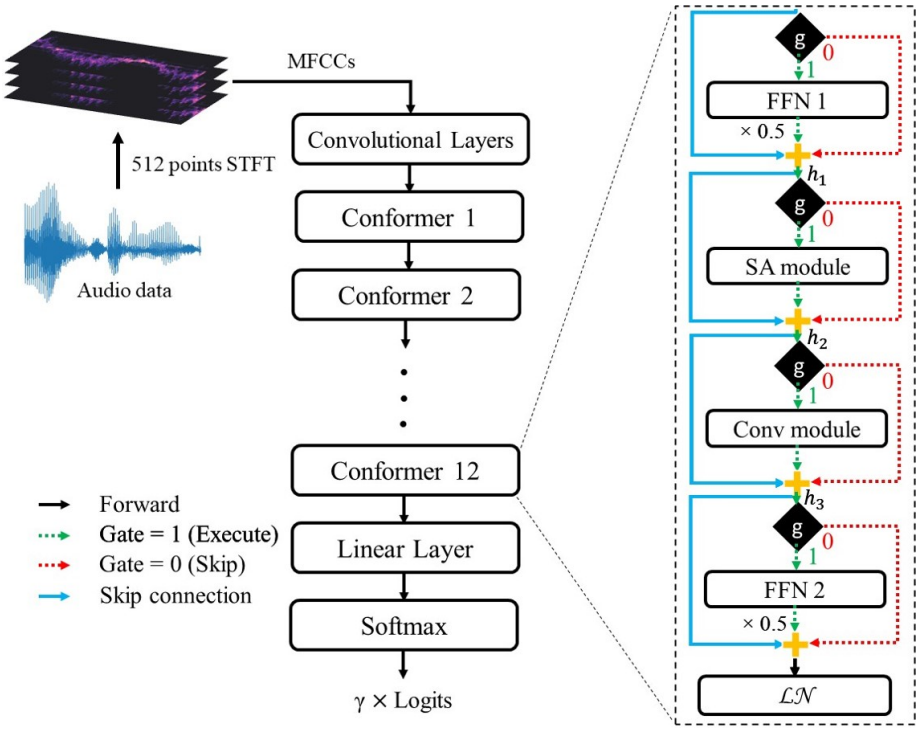

FIGURE 5.1: Left: overall architecture of the conformer. Right: conformer's structure with modified Gating mechanism. FFN, SA, Conv, and γ refers to Feed-forward network, Self-Attention, Convolutional, and scaling-factor. Red arrows show the path when a block is dropped ($g = 0$) and blue arrows refer to the original conformer's skip connection.

where p_{in} is the probability for the output to be 1 and k are the possible outcomes. The last conformer block's output is forwarded to the linear layer followed by a *log-softmax* operation predicting probability of the tokens. These token probabilities are provided to a connectionist temporal classification (CTC) decoder to produce transcript for the respective audio.

5.2.2 Layer Drop Strategy

Layer Drop is a technique that provides regularization effect during training process [26] by bypassing specified conformer blocks to prevent over-fitting (Figure 5.1). It is equivalent to the product of input with binary gate value g.

$$y = x \times g \tag{5.2}$$

where x and y being the current input and the respective output of a conformer module.

The static dropping probability $p_d = 1 - p_{in}$ is set before training defining the approximate percentage of model used on average during an epoch. In addition, each gate's value is specified by *BRV's* output giving a possible outcome of either 0 with probability $1 - p_d$ or 1 with a probability of p_{in}. For instance, for a single conformer module with $g = 1$ as shown in Figure 5.1, the input goes through each sub-module as:

$$h_1^i = x^i + g^i \times f_{\text{FFN1}}^i(x^i) \times 0.5 \tag{5.3}$$

$$h_2^i = h_1^i + g^i \times f_{\text{SA}}^i(h_1^i) \tag{5.4}$$

$$h_3^i = h_2^i + g^i \times f_{\text{Conv}}^i(h_2^i) \tag{5.5}$$

$$y^i = \mathcal{LN}^i(h_3^i + g^i \times f_{\text{FFN2}}^i(h_3^i) \times 0.5) \tag{5.6}$$

where i is the index of conformer module ($i \in 1, 2, 3, ..., 12$), x^i, y^i and g^i are input, output and gate of the i^{th} conformer module, while h_1^i, h_2^i, and h_3^i are the outputs of feed-forward network 1, self-attention, and convolutional sub-module. h_{i3} flows through feed-forward network 2 referred as f_{FFN2}, before processed by final *Layer Normalization* ($\mathcal{LN}$). The gate g^i is shared among the submodules of one conformer module and controls whether all other conformer sub-modules except $\mathcal{LN}$ will be skipped or not. If $g^i = 0$, then the output of conformer module is represented as:

$$y^i = \mathcal{LN}(x^i) \tag{5.7}$$

The reason for retaining $\mathcal{LN}$ is that it helps the model to converge quickly as compared to when it is dropped. More discussion about the effect of layer normalization is discussed in section 5.4.4.

5.3 Implementation Details

The model is trained on NVIDIA A40 GPU with a batch size of 48, and AdamW optimizer with beta values of (0.9, 0.98) and weight decay of $5e^{-4}$. In our model. the embedding and feed-forward dimensions were set to 256 and 2048 whereas the vocabulary sizes for encoder and decoder were set as per the sentencepiece-bpe-256 [70] sizes, constituting 31.6M learnable parameters. The learning rate was increased linearly for initial mini-batches and decreased exponentially afterwards using a scheduler.

5.4 Results

5.4.1 Analysis using Probabilistic dropping and SOTA Comparison

Figure 5.2 illustrates the WER obtained on *LibriSpeech test-clean* set for models obtained with different training probabilities and applying *probabilistic dropping* at inference for $p_{d-inf} = 0.0$ to 0.8. For $p_{d-inf} = 0.0$ and 0.1, the model trained with $p_{d-tr} = 0.2$ achieves the best WER of 6.03% and 6.33% for high resource-settings as compared to model trained with other dropping probabilities. In contrast, the model trained with $p_{d-tr} = 0.8$ outperforms others in low-resource settings, providing a WER of 29.79% for $p_{d-inf} = 0.8$ whereas for $0.3 \geq p_{d-inf} \geq 0.6$, the model trained with $p_{d-tr} = 0.5$ has the edge over other models. The $p_{d-tr} = 0.5$, which is extensively used in literature for fine-tuning large models for various applications, prompts as the best performance-computational reduction trade-off that provides comparable performance to other trained models in both high and low-resource scenarios. In addition, the reported results in [137] shows an increase of 7.19%, 4.23%, and 2.47% in WER for p_{d-inf} of $0.5, 0.4$, and 0.3 after fine-tuning the pre-trained WavLM large model with layer dropping probability of 0.5. Our conformer model that is trained from scratch with $p_{d-tr} = 0.5$, exhibits 2.26% and 0.31% increase in WER for p_{d-inf} of 0.5 and 0.4 as compared to the WER obtained with $p_{d-inf} = 0.0$, and even improves the WER by 0.46% WER for $p_{d-inf} = 0.3$ on LibriSpeech test-clean set. The improvement in terms of WER underlines that training from scratch

TABLE 5.1: Comparative results (in WER %) on LibriSpeech test-clean for different dropping strategies, early exits and small sized conformer models. p_{d-tr} indicates the dropping probability used during training. n refers to the number of blocks removed during inference out of 12 conformer blocks. Model's WER for $p_{d-tr} = 0$ is 6.56%. Small Conformers are models trained with less (12 - n) number of conformer modules. For Random and Greedy dropping, bold values show best performing models for different n. Values with * are taken from [123].

	Random dropping			**Greedy dropping**			**Early Exit**	**Small Conformers**
	p_{d-tr} **= 0.8**	p_{d-tr} **= 0.5**	p_{d-tr} **= 0.2**	p_{d-tr} **= 0.8**	p_{d-tr} **= 0.5**	p_{d-tr} **= 0.2**		
0	21.34	7.69	6.03	-	-	-	5.17*	6.56
2	15.77	6.95	**6.47**	15.53	6.65	**5.82**	5.27*	6.89
4	13.45	**6.92**	7.57	12.77	**6.74**	8.89	5.93*	7.10
6	12.74	**7.98**	13.36	11.83	**6.98**	10.18	6.82*	7.63
8	13.72	**12.54**	38.77	12.26	**10.41**	53.36	11.60*	9.86
10	**23.88**	43.62	87.37	**24.18**	38.90	81.61	23.98*	17.64

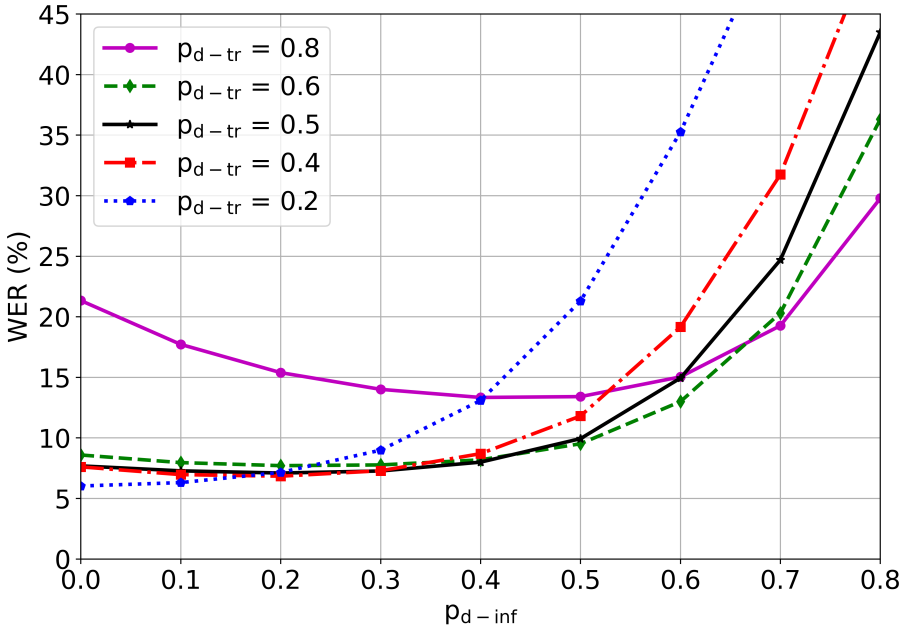

FIGURE 5.2: Inference time WER variation for **probabilistic dropping** strategy on model trained with different static dropping probability.

is more beneficial than fine-tuning of large models for ASR applications, hence, enabling our model to effectively fit with-in available resources.

TABLE 5.2: WER on TED-LIUM's test split for different dropping strategies, early exits and small conformers. Model's WER for $n = 0$ is 13.80%. Values with * are taken from [123]

n	Random dropping p_{d-tr} = 0.5	Greedy dropping p_{d-tr} = 0.5	Early Exit	Small Models
0	14.03	-	14.63*	13.80
2	13.94	13.69	14.95*	15.00
4	14.25	14.02	16.13*	16.83
6	15.78	14.71	18.06*	18.15
8	22.43	19.49	23.41*	21.61
10	55.30	48.51	43.87*	32.98

TABLE 5.3: Comparison of WER for the pre-defined approaches in [99] versus proposed approaches when tested on the model trained with $p_{d-tr} = 0.5$ using LibriSpeech. Here, value of n is set to 6.

	Pre-defined dropping techniques				Proposed Approaches	
	top	central	bottom	even alternate	Random dropping	Greedy dropping
WER	24.08	8.53	11.72	7.34	7.98	6.98

5.4.2 Analysis using Random and Greedy dropping

Table 5.1 enlists our model's performance using Random and Greedy dropping, and compares them to the early-exits and smaller conformers on LibriSpeech corpus. The $p_{d-tr} = 0.2$ model performs better when the number of dropped conformer blocks (n) is 2 while $p_{d-tr} = 0.8$ is effective for $n = 10$. The $p_{d-tr} = 0.5$ provides good transcription for a wider range of dropped conformer blocks ($n = 4, 6, 8$) and also provides approximated performance to the early-exits model without a separately trained decoder for each exit location, which itself is delicately chosen in EE models. Our model even surpasses the early-exit's performance for $n = 8$ but fails to keep up the performance with early-exits strategy for $n = 10$ due to the lack of specialized decoder. The results for models $p_{d-tr} = [0.4, 0.6]$ exhibits similar trend with slightly poor performance than $p_{d-tr} = 0.5$, that are listed in Table 5.1 for both random and greedy dropping. Similarly, the small conformers performs slightly better as compared to $\mathcal{LD}$ due to the optimized decoder and other model's hyper-parameters. Table 5.2 enlists our model's performance on TED-LIUM dataset trained with $p_{d-tr} = 0.5$. It can be seen that, differently from what reported in Table 5.1, the model with $\mathcal{LD}$ has better performance as compared to early-exits strategy and small conformers for most of the number of dropped conformer modules, showing the efficacy of $\mathcal{LD}$ strategy. Only for $n = 10$, early-exits and small conformers has better performance than the strategies used in this paper.

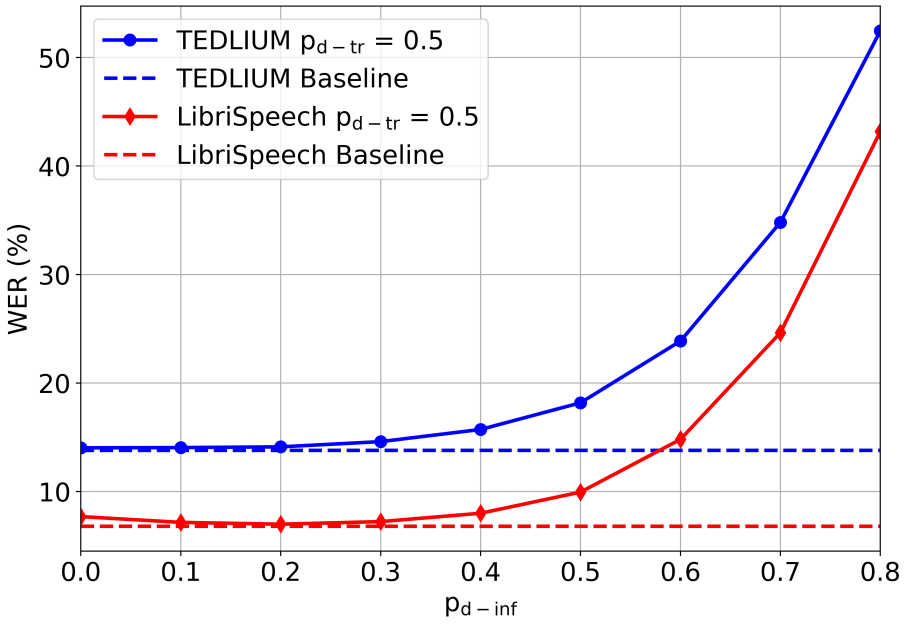

FIGURE 5.3: Architectural adaptability illustration for model trained with $p_{d-tr} = 0.5$ on LibriSpeech and TED-LIUM corpora along with comparison with Baseline model without $\mathcal{LD}$.

5.4.3 Comparison with pre-defined dropping techniques

Table 5.3 lists WER using the proposed strategies in [99] to drop multiple pre-defined combinations of layers. For model trained with $p_{d-tr} = 0.5$ on LibriSpeech-1000, dropping *top-6* conformer blocks (conformer # 7 - 12) performs worst in our case with a WER of 24.08% than dropping *bottom-6* conformer blocks (conformer # 1 - 6), which gives a WER of 11.72%. Moreover, *symmetric* dropping of conformer blocks (central-6) displays slightly better performance than dropping top-6 and bottom-6 with a WER of 8.53% while dropping *even/odd-numbered* conformer blocks achieves comparable performance to ours. Overall, irrespective of the difference in model sizes, the dropping methods used in this work outperforms the results in [99], displaying a WER of 6.98%.

In general, using $p_{d-tr} = 0.5$ as in [137], turns out to be the best performance-computation trade-off on LibriSpeech and TED-LIUM datasets (see Table 5.1 and 5.2), providing the optimum architectural adaptability for a range of p_{d-inf}'s as depicted in figure 5.3. Nevertheless, other p_{d-tr} values may be preferable if the model is expected to work in low or high-resource conditions. Furthermore, there are only marginal differences between random dropping and the pre-defined dropping strategies in [99] that makes $\mathcal{LD}$ to be an optimal choice for unknown operating conditions as pre-defined approaches do not provide structural adaptability.

5.4.4 Training with Layer Normalization

We experimented with dropping complete conformer blocks as well as retaining the last *Layer Normalization*. Figure 5.4 shows how retaining the $\mathcal{LN}$ during training helps the model converge better as compared to dropping the complete conformer block. The work [26] applied $\mathcal{LD}$ on transformer-based models concluding that partial dropping in their architecture doesn't resulted in performance improvement, whereas, the training stability is improved in our case by retaining the last $\mathcal{LN}$.

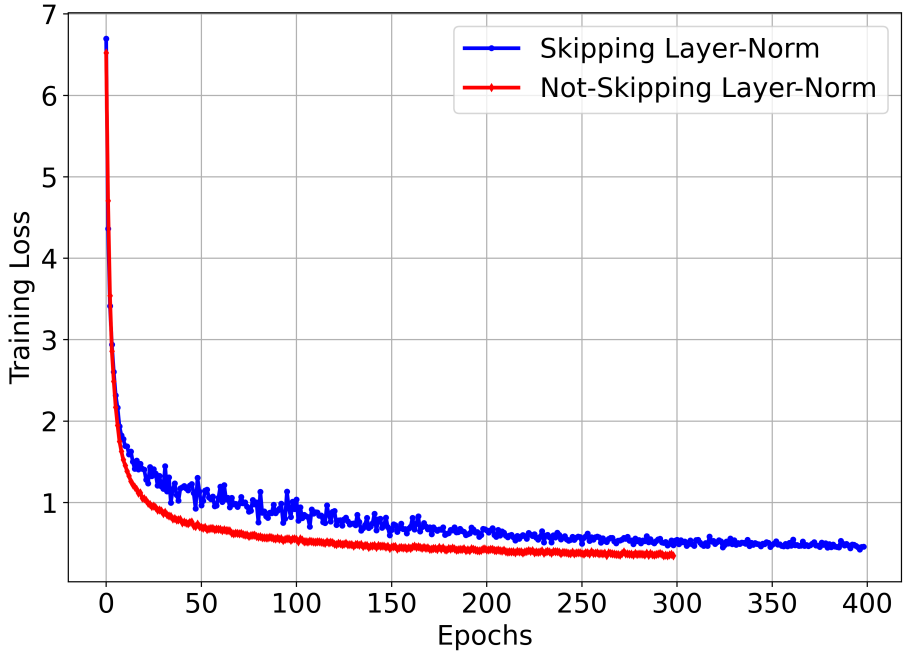

FIGURE 5.4: Convergence variation with dropping complete conformer block including $\mathcal{LN}$ (in blue) vs retaining the $\mathcal{LN}$ (in red). Utilizing the last $\mathcal{LN}$ (with gate = 0) provides smoother curve and better convergence.

Chapter 6

Input Driven Layer Dropping

This Chapter is taken from our manuscript published at IEEE Workshop on Machine Learning and Signal Processing, 2025 [42]. It focuses on improving the performance-computation trade-off of random layer dropping by adding input-conditioning for layer dropping.

6.1 Objectives

Random Layer Dropping introduces the architectural adaptability in the network, featuring improved functioning in variable resource settings. However, the performance-computation trade-off is sub-optimal and should be improved for better efficiency. In addition, **RD** neglects the input information to select the encoder sub-modules which limits the overall efficiency. To address this random selection of encoder sub-modules, we proposed an input conditioned dropping method [42] that incorporates the information of each input to decide which encoder sub-modules are more important for the current input. Such input conditioning results in more suitable selection of encoder sub-modules for each audio, giving improved performance. The input conditioning is performed using a lightweight *layer selecting* block that takes the output of feature extractor and supervises the encoder in the selection of encoder sub-modules. Moreover, we also evaluated our method in the absence of feature extractor (see section 6.4.2) by giving the input directly to the layer selecting block.

6.2 Methodology

To mitigate the performance limitations without significantly altering the architecture of foundation model, we propose a lightweight, plug-and-play mechanism called as **input-driven layer dropping (IDLD)** using a **Layer Selecting** block as illustrated in Figure 6.1. For each input sample, the LS block selects the finest combination of encoder layers achieving optimal performance for various resource settings.

Given a dataset consisting of audio-label pairs $\{(a^i, t^i_{ref})\}_{i=1}^{D}$ with D instances and a foundation model $\mathcal{M}$ consisting of a feature extractor $\mathcal{F}$, encoder $\mathcal{E}$ with N layers, and a linear layer as a decoder, the objective is to transform the static model $\mathcal{M}$ in to a dynamic one, so that it can produce the best possible output irrespective of the number of encoder layers executed for the current sample.

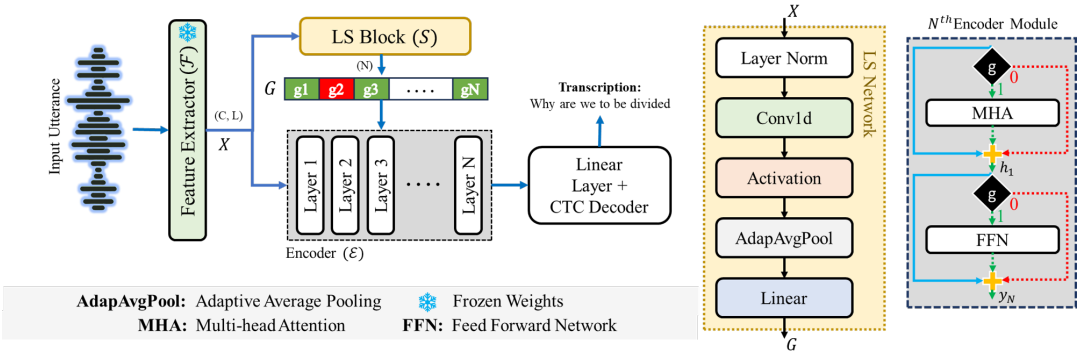

FIGURE 6.1: Illustration of the proposed input-driven layer skipping approach using a pretrained encoder. The output of the feature extractor (X) is forwarded to both Encoder $\mathcal{E}$ and Layer Selector network $\mathcal{S}$. The latter provides layer scores $G \in \mathbb{R}^N$, for each layer of the encoder. Top-k values from $G = \{g^1, g^2, .., g^N\}$ are selected to enable corresponding encoder modules while skipping the remaining ones.

6.2.1 Data Flow

Figure 6.1 illustrates the overall framework of the proposed data-driven approach. The feature extractor $\mathcal{F}$ takes the i^{th} audio sample a^i to produce a feature representation $X^i = \mathcal{F}(a^i)$ that is utilized by encoder $\mathcal{E}$ and Layer Selecting block $\mathcal{S}$. The final logits p^i are obtained by linearly projecting the output of N^{th} encoder layer, and depending on the downstream application, either forwarded to a CTC decoder to get the transcription or is used to infer the target class.

$$p^i = \texttt{Softmax}\big(f_{\texttt{linear}}(\mathcal{E}(X^i))\big) \tag{6.1}$$

6.2.2 Static vs Dynamic Encoder

For static models, each encoder layer consists of a multi-head attention (MHA) and feed-forward network (FFN) and can be formulated as:

$$\begin{aligned} \hat{y}^j &= y^{j-1} + f^j_{\texttt{MHA}}(y^{j-1}) \\ y^j &= \hat{y}^j + f^j_{\texttt{FFN}}(\hat{y}^j) \end{aligned} \tag{6.2}$$

where y^{j-1} and y^j are the input and output of encoder layer $\mathcal{E}^j$ $(j = \{1 \dots N\})$. To allow skipping encoder layers, we add a gating mechanism (right most part of Figure 6.1) that decides whether to skip or execute the complete encoder layer $\mathcal{E}^j$, instead of dropping separate sub-modules (MHA and FFN) as in [91]. So, equation 6.2 can be written as:

$$\begin{aligned} \hat{y}^j &= y^{j-1} + g^j \times f^j_{\texttt{MHA}}(y^{j-1}) \\ y^j &= \hat{y}^j + g^j \times f^j_{\texttt{FFN}}(\hat{y}^j) \end{aligned} \tag{6.3}$$

where the value of g^j is given by Layer Selecting block.

The **Layer Selecting block** shown in Figure 6.1, normalizes the feature representation X^i of the sample a^i and applies 1-D convolution with C output channels followed by GELU activation. Afterwards, adaptive average pooling is applied to overcome input's dimensional variability (due to variable utterance length) followed

by linear projection to get soft gate descriptors $\bar{g}^i \in \mathbb{R}^N$.

$$G = [\bar{g}^1, \ldots, \bar{g}^N] = S(X) \tag{6.4}$$

During **training**, the number of encoder layers per sample are selected using the top-k values of G where k is sampled from uniform distribution $U(1, N)$. In this way, using top-k method, the soft-gate values are transformed to binary gate descriptors g^j of corresponding encoder layer $\mathcal{E}^j$.

$$[g^1, \ldots, g^N] = \text{top-}k\big(G, k\big) \text{ where } k \in U(1, N) \tag{6.5}$$

For ASR, the logits p^i of Eq. 6.1 are forwarded to a CTC decoder [34] to generate the transcription using k encoder layers $t^i_{pred}|\mathcal{E}_{top-k}$, which is employed to estimate the CTC loss $\mathcal{L}_{CTC}$ to be minimized:

$$\min \mathcal{L}_{CTC}\left(t^i_{pred}|\mathcal{E}_{top-k},\ t^i_{ref}\right) \tag{6.6}$$

For other downstream tasks, the logits computed with k encoder layers $p^i|\mathcal{E}_{top-k}$ are utilized directly to estimate the cross-entropy loss $\mathcal{L}_{CE}$:

$$\min \mathcal{L}_{CE}\left(p^i|\mathcal{E}_{top-k},\ t^i_{ref}\right) \tag{6.7}$$

At **inference**, we either set k in top-k to select a specified number of encoder layers or we define a threshold on the gate scores to select the relevant layers for a particular input (details in Sec 6.4.4).

6.3 Implementation Details

6.3.1 WavLM

For **ASR**, we freeze the feature extractor of pretrained WavLM-base model and finetune the model for 50 epochs. The training is performed on a NVIDIA $A40$ GPU with a batch size of 8 using Adam optimizer along with SpecAug [88] with the default settings. We utilized the Connectionist Temporal Classification (CTC) loss for optimization while linearly increasing the learning rate followed by exponential decrease [111].

For **IC** task, we employed AdamW optimizer with high L2 regularization of $1e^{-2}$ to prevent overfitting on FSC dataset. We trained the model on $A40$ GPU for 20 epochs optimizing the cross entropy loss with a batch size of 32 and initial learning rate of $1e^{-4}$ (decreased based on validation loss). The experiments are repeated 3 times and average values are reported.

6.3.2 Audio Spectrogram Transformer

For remaining tasks, we finetune the AST model pretrained on AudioSet using AdamW optimizer while reporting averaged accuracy of 3 runs. The training is performed on $A40$ GPU with $L2$ regularization of $1e^{-2}$ and initial learning rate of $1e^{-4}$ (decreased based on validation loss). As recommended in the respective articles, we applied 5-fold cross validation on ESC-50 dataset (**SC** task) and 10-fold cross validation on IEMOCAP dataset (**ER** task) for 50 and 45 epochs, while finetuning the model for 75 epochs on FSC dataset (**IC** task). The training period (different number

TABLE 6.1: Dynamic behaviour depiction using WavLM model for (i) Automatic Speech Recognition, (ii) Intent Classification. (**n** - number of dropped layers, **RD** - Random Dropping, **IDLD** - Input-Driven Layer Dropping, **EE** - Early Exit)

	LibriSpeech (WER (%))			**TEDLIUM-v3** (WER (%))			**FSC** (Accuracy (%))		
n	**RD**	**IDLD**	**EE**	**RD**	**IDLD**	**EE**	**RD**	**IDLD**	**EE**
0	5.47	4.17	**3.87**	10.32	9.91	**9.11**	99.14 ±0.12	**99.27 ±0.10**	98.75 ±0.74
2	5.78	**4.22**	4.63	10.75	**9.96**	10.66	99.19 ±0.13	**99.57 ±0.07**	98.69 ±0.73
4	6.52	**4.59**	6.55	12.53	**11.31**	13.92	98.84 ±0.37	**99.48 ±0.09**	99.34 ±0.13
6	9.01	**6.08**	9.09	17.74	**15.09**	18.20	98.63 ±0.26	**99.24 ±0.08**	99.33 ±0.20
8	17.59	**11.24**	14.84	31.90	**26.33**	27.06	94.76 ±0.37	95.23 ±0.07	**98.98 ±0.23**
10	60.10	**38.75**	38.85	76.76	63.52	**55.91**	60.59 ±0.88	74.61 ±0.44	**97.64 ±0.16**

of epochs) lasted for different duration depending on different sizes of the datasets, and to allow the AST model to converge. Furthermore, we empirically observed that the absence of a feature extractor is AST affects the extent of training required for model's convergence. Finally, to increase the available data for finetuning, we utilized SpecAug with frequency and time masking of 27 and 80 samples respectively.

6.4 Results

6.4.1 WavLM

Table 6.1 enlists the obtained results in terms of: (i) WER for ASR on LibriSpeech test-clean and TEDLIUM-v3 corpus, and (ii) Accuracy for IC using FSC dataset. In case of **ASR** (left half of Table 6.1), IDLD outperforms RD for all values of dropped encoder layers n (note that $n = N - k$), especially for n $= 10$ and 8 with a staggering difference of 27.66% and 10.07% on LibriSpeech. Similarly, IDLD outmatches EE for n ≥ 2 showing the overall efficacy of input-conditioned layer dropping. In case of TEDLIUM-v3, we observe a similar trend as IDLD comfortably surpasses RD for all values of dropped encoder layers, with the difference becoming more evident for dropping n ≥ 6 encoder layers.

For **IC** task, IDLD again surpasses RD in terms of performance for all values of n. The difference of 14.02% in accuracy for n $= 2$ using RD endorses our claim that input-conditioned dropping of layers yields significant improvement as compared to RD. However, when evaluated against EE it shows comparable (or better) performance for n ≤ 6 while lagging behind for n $= 8$ and 10.

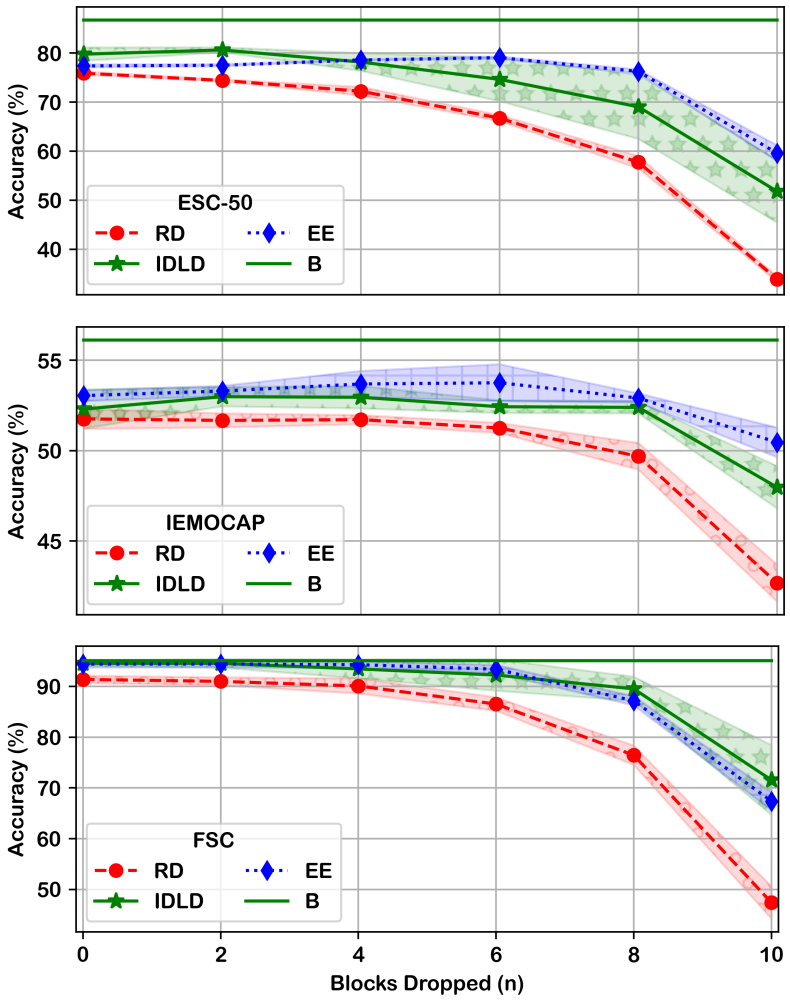

FIGURE 6.2: Performance evaluation against the model's varying size using AST for (i) Sound Classification, (ii) Emotion Recognition, and (iii) Intent Classification. Light shaded color represents standard deviation of averaged runs.

6.4.2 Audio Spectrogram Transformer

It is worth mentioning that AST being a purely attention-based classification model, doesn't contains a convolutional feature extractor which limits the learning of rich semantic representation required for LS block. Nevertheless, our method properly works with plain mel-spectrograms, and easily outperforms the random dropping as well as produce comparable results (even surpass in some cases) to early exit. Figure 6.2 illustrates the observed variation in averaged accuracy using RD, IDLD, and EE methods. It is evident from Figure 6.2 that input-driven layer dropping (solid green line) is far more superior than RD for all model sizes in each application, and the performance gap becomes more significant as n decreases ($\downarrow$ model size / $\uparrow$ n). For instance, RD tries to keep up with IDLD for $n \leq 6$ but collapses after $n > 6$ due to the randomly selecting encoder modules. Quantitatively, RD exhibits 17.08%, 24.08%, and 5.31% less accuracy than IDLD for $n = 10$ on ESC-50, FSC, and IEMOCAP datasets. Conversely, when compared against EE, IDLD performs better for some values of n and lags behind for others on random across different applications. For example, for ESC-50 dataset, IDLD demonstrates a $5-$fold averaged accuracy of 80.61% and 78.15% as compared to 77.50% and 78.57% for EE, when dropping 2 and

TABLE 6.2: Performance evaluation against the model's varying size using Audio Spectrogram Transformer (AST) model for (i) Sound Classification (ii) Intent Classification (iii) Keyword Spotting (iv) Emotion Recognition. (**n** - number of dropped blocks, **RD** - Random Dropping, **IDLD** - Input-Driven Layer Dropping, **EE** - Early Exit, **B** - Baseline)

	Environmental Sound Classification (ESC-50)			Fluent Speech Commands (FSC)			Google Speech Commands (GSC)			IEMOCAP		
n	RD	IDLD	EE	RD	IDLD	EE	RD	IDLD	EE	RD	IDLD	EE
0	75.88 ±0.31	**79.76 ±1.36**	77.37 ±0.31	91.39 ±0.75	**94.56 ±0.53**	93.86 ±0.33	96.84 ±0.06	96.87 ±0.05	**96.98 ±0.12**	51.76 ±0.56	52.30 ±1.08	**53.04 ±0.30**
2	74.40 ±0.23	**80.61 ±0.51**	77.50 ±0.21	91.01 ±0.78	**94.56 ±0.85**	93.81 ±0.31	96.67 ±0.15	96.86 ±0.05	**96.94 ±0.10**	51.67 ±0.39	52.99 ±0.54	**53.30 ±0.29**
4	72.20 ±0.88	78.15 ±1.81	**78.57 ±0.09**	90.10 ±1.49	93.46 ±1.87	**93.72 ±0.34**	96.44 ±0.21	**96.91 ±0.07**	96.90 ±0.19	51.71 ±0.25	52.95 ±0.62	**53.69 ±0.72**
6	66.72 ±0.62	74.62 ±4.45	**79.05 ±0.25**	86.54 ±1.34	92.26 ±3.10	**92.90 ±0.42**	96.04 ±0.23	**96.86 ±0.06**	96.85 ±0.22	51.25 ±0.28	52.43 ±0.36	**53.76 ±1.02**
8	57.75 ±1.27	69.03 ±6.56	**76.23 ±0.39**	76.45 ±1.90	**89.53 ±2.39**	86.54 ±1.12	94.33 ±0.22	95.96 ±0.07	**96.01 ±0.18**	49.68 ±0.74	52.39 ±0.33	**52.91 ±0.26**
10	33.92 ±0.52	51.80 ±6.37	**59.58 ±1.59**	47.39 ±3.18	**71.56 ±6.88**	66.76 ±2.75	82.54 ±1.48	91.54 ±0.61	**92.53 ±0.30**	42.66 ±1.03	47.97 ±1.17	**50.46 ±0.82**
B	**86.70**			**95.09**			**97.50**			**56.12**		

4 encoder layers. The averaged accuracy for IDLD and EE closely follows each other with both surpassing one another in difference resource settings for ESC-50 and FSC datasets. Moreover, the minute difference in final accuracy for different number of dropped layers among IDLD and EE in different applications, highlights the technique's robustness with single classifier coupled with input-conditioning against EE with auxiliary classifiers. In addition to Figure 6.2, the results for all downstream applications are listed in Table 6.2, with additional results on Google Speech Commands dataset.

6.4.3 Further Analysis of EE vs IDLD

In Table 6.1 and Figure 6.2, EE demonstrates good performance in all applications. It should be noted that EE trains several specialized auxiliary classifiers and optimizes the overall joint loss function. Conversely, IDLD uses a solitary classifier with a simple loss function, working in conjunction with the LS block, to generate optimal output for all selected sub-networks. This fact favors EE when dealing with harder tasks or heavy mismatch between the pretrained model and the downstream task. Finally, note that due to the uniform sampling of k during training, the IDLD model observes more often the lowest exits than the highest ones, being better optimized for them.

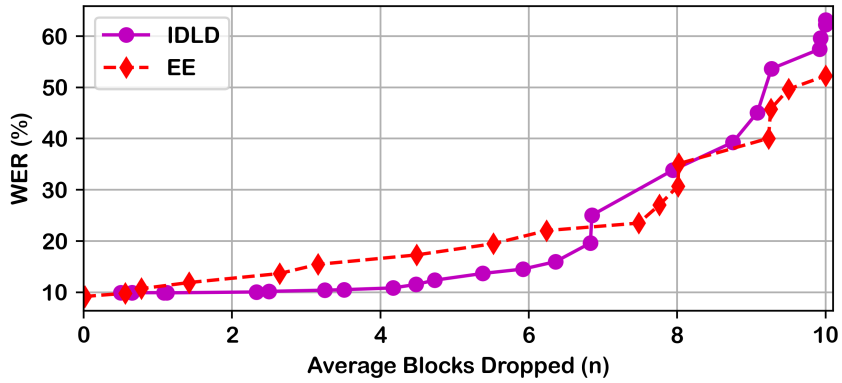

FIGURE 6.3: Threshold based WavLM's encoder block selection on Tedlium test split.

TABLE 6.3: WER (in %) for IDLD and Random Dropping on LibriSpeech test-clean split using WavLM model

n	IDLD	Th-IDLD	RD with p_{d-tr} =			
			0.2	0.5	0.8	0.2 - 0.9
0	**4.17**	6.10	4.36	5.47	14.38	4.76
2	**4.22**	5.17	5.12	5.78	13.78	5.13
4	**4.59**	4.94	7.48	6.52	13.58	6.09
6	6.08	**5.83**	14.79	9.01	14.62	8.36
8	**11.24**	11.90	41.46	17.59	18.70	15.47
10	**38.75**	62.82	94.05	60.10	39.45	45.91

6.4.4 Threshold based selection

At inference, besides using top-k approach, the encoder layers required for each sample can be automatically inferred using a threshold-based mechanism. To accomplish this, the soft gate values $\bar{g}^i \in \mathbb{R}$ are compared against a threshold (Γ) to deduce the best suited encoder layers for every sample, and binarizing $\bar{g}^i$ into $g^i \in \{0,1\}$. Likewise, for EE model, the average entropy of the token's distributions is computed from each auxiliary exit, and the final output is taken from the lowest exit with entropy below the threshold value [123]. Figure 6.3 illustrates the averaged encoder modules selected along with corresponding WERs when varying the gate-threshold value from -0.6 to 0.4 for IDLD, and the entropy-threshold from 0.025 to 0.7 for EE. Note that similarly to the results in Table 6.1 and Figure 6.2, IDLD is superior than EE in the middle and final exits while follows EE in the lowest exits.

6.4.5 IDLD vs RD: Does LS Block really works ?

To establish the superiority of IDLD over RD, we separately trained models with distinct dropping probabilities p_{d-tr} that are suitable for different resources as well as variably selecting the dropping probability in the similar range as IDLD. Table 6.3 validates our claims that IDLD always perform better than random dropping, yielding better performance-computation trade-off for all model's sizes. In addition,

a single model trained with our approach suffices individual models trained with random dropping for various resource settings, providing memory and performance efficiency.

6.4.6 Limitations

Table 6.1 and Figure 6.2 highlight the superiority of IDLD over RD in different audio applications. However, there is still room to further enhance the overall performance, primarily by tweaking the LS block's architecture. Secondly, the output's standard deviation with IDLD method is greater than RD in AST model, hinting the reliance on deep feature extractor that is absent in AST. Shortly, we believe that optimizing the LS block and minimizing the output's standard deviation will further boost our method's efficiency.

Chapter 7

Knowledge Distillation meets Layer Dropping

This Chapter is extracted from our manuscript under review at ICASSP, 2026. It pushes to further improve the performance-computation trade-off using knowledge distillation in parallel with $\mathcal{LD}$.

7.1 Objectives

Incorporating input conditioning to the layer dropping technique resolves the performance limitations caused by random selection, however, the performance-computation trade-off can be further improved. This chapter targets to improve the performance-computation trade-off by distilling knowledge from a teacher model which supervises the final output of dynamic student model.

7.2 Experimental Setup

7.2.1 Architectures

The proposed DLD framework is evaluated on two architectures: (i) **Conformer:** a modified version with light feature extractor along with linear projector instead of LSTM-based projector (similar to [41, 42]), (ii) **WavLM-base** model [14].

In case of conformer model, mel-spectrograms are extracted for each 320 samples (20 ms) of input sequence with a hop-length of 160 samples (10 ms), whereas the input sequence is fed directly to the WavLM model which contains a 7-layered feature extractor module.

7.2.2 Implementation Details

For both architectures, we kept the reference model frozen and finetuned the dynamic student model using CTC loss on a NVIDIA $A40$ GPU. In case of Conformer, the weights of $\mathcal{M}_{\text{DS}}$ are initialized with pre-trained weights of static model and finetuned for 100 epochs using batch size of 64 and L2 regularization of $5e^{-4}$. The learning rate is increased for 10k warm-up steps and exponentially decreased till the end of training. In case of WavLM-base, we initialized $\mathcal{M}_{\text{DS}}$ with pre-trained weights and finetuned using Adam optimizer with a batch size of 8 and default configuration settings. We measured the architecture's performance in terms of Word Error Rate (WER).

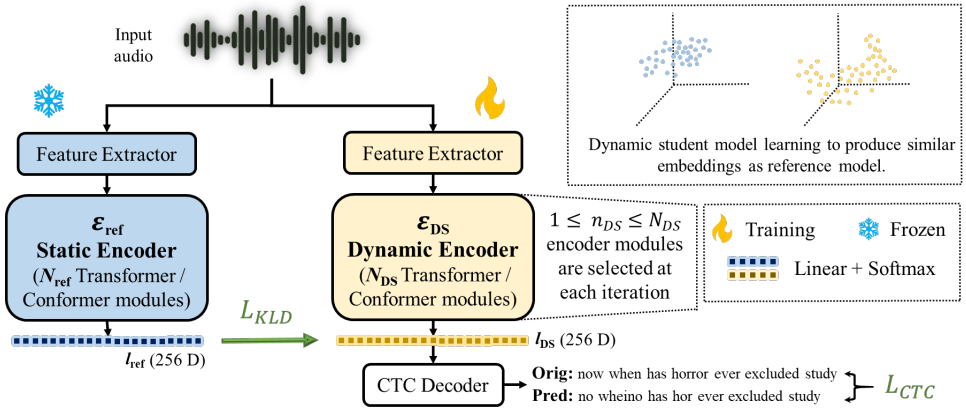

FIGURE 7.1: Illustration of proposed DLD framework. $\mathcal{M}_{\text{ref}}$ uses all encoder layers to supervise the embeddings of $\mathcal{M}_{\text{DS}}$.

7.2.3 Datasets

For ASR downstream task, we employed two publicly available corporas: LibriSpeech-1000 [87] and TED-LIUM v3 [48].

7.2.4 Baseline

For conformer architecture, we compare against Random Dropping (RD) method with dropping probability $p_d = 0.5$ [41] (referred as "RD_{LD}" in this paper) that is trained for 300+ epochs and an input-driven dropping strategy [91] (referred as "I3D") trained on 100 hours of librispeech corpora. Additionally, we compare against a conformer trained from scratch "RD_{sc}" with $p_d = 0.5$ for 150 epochs. For WavLM, we compared against RD with $p_d = 0.5$ [42] (referred as "RD_{w-sc}" in this paper). Unless stated, all training procedures are performed using complete datasets (librispeech (1000hrs) and tedlium-v3 (452hrs)).

7.3 Methodology

7.3.1 Architecture and Data Flow

$\mathcal{M}_{\text{ref}}$ consists of a feature extractor $\mathcal{F}_{\text{ref}}$, encoder $\mathcal{E}$ with N_{ref} transformer/conformer modules, and a linear projector whose output undergoes a softmax operation to get latent embedding l_{ref}. Similarly, $\mathcal{M}_{\text{DS}}$ comprises of a feature extractor $\mathcal{F}_{\text{DS}}$, encoder $\mathcal{E}_{\text{DS}}$ with a total of N_{DS} transformer/conformer modules from which n_{DS} are used per iteration, and a linear projector followed by a softmax, providing latent embedding l_{DS}. The $\mathcal{E}_{\text{DS}}$ is equipped with a gated mechanism that allows to process or skip the i-th encoder module using a gate g^i. The gate $g^i \in \{0, 1\}$ follows the bernoulli distribution (BD) with a probability of 0.5.

$$y^{i+1} = g^i \times y^i \tag{7.1}$$

where y^i and y^{i+1} are the input and output of the i-th encoder module. As g^i follows the bernoulli distribution, n_{DS} also becomes a bernoulli random variable whose value can be determined as $n_{\text{DS}} = \sum_{i=1}^{N_{\text{DS}}} g^i$ for each iteration during training period. At inference, the value of n_{DS} is defined enabling us to evaluate the model's performance to different encoder depths, i.e. $n_{\text{DS}} = 2, 4, \ldots, N_{\text{DS}}$.

For a dataset $\mathcal{D} \in \{(a^j, t^j)\}_{j=1}^{J}$ comprising of J audio-transcription pairs, $\mathcal{M}_{\text{ref}}$ produces a latent space representation $l_{ref}^{j}|\mathcal{E}_{N_{\text{ref}}}$ for the j-th input using N_{ref} encoder modules. In contrast, $\mathcal{M}_{\text{DS}}$ utilizes n_{DS} number of encoder modules for each iteration to produce the latent space representation $l_{\text{DS}}^{j}|\mathcal{E}_{n_{\text{DS}}}$ where $1 \leq n_{\text{DS}} \leq N_{\text{DS}}$.

7.3.2 Objective Function

To align the distributions of reference model's embedding $l_{\text{ref}}^{j}|\mathcal{E}_{N_{\text{ref}}}$ and dynamic student model's embedding $l_{\text{DS}}^{j}|\mathcal{E}_{n_{\text{DS}}}$, we employ Kullback–Leibler divergence ($\mathcal{L}_{\text{KLD}}$) loss which minimizes the statistical distance between both embeddings.

$$\mathcal{L}_{\text{KLD}} = \min \sum_{j=i}^{J} \mathrm{D}_{\text{KL}}\left(l_{\text{ref}}^{j} \,||\, l_{\text{DS}}^{j}\right) \tag{7.2}$$

In addition, the token probabilities $l_{\text{DS}}^{j}|\mathcal{E}_{n_{\text{DS}}}$ produced by $\mathcal{M}_{\text{DS}}$ for input a^j, are forwarded to a Connectionist Temporal Classification (CTC) decoder [34] to generate predicted text, which is employed to estimate CTC loss ($\mathcal{L}_{\text{CTC}}$) by comparing with ground-truth transcription t^j.

$$\mathcal{L}_{\text{CTC}} = \min \sum_{j=1}^{J} \mathbf{F}_{\text{CTC}}\left(l_{\text{DS}}^{j}|\mathcal{E}_{n_{\text{DS}}}, t^{j}\right) \tag{7.3}$$

The overall loss function to be minimized is given as:

$$\mathcal{L} = \mathcal{L}_{\text{KLD}} + \mathcal{L}_{\text{CTC}} \tag{7.4}$$

7.4 Results

7.4.1 Conformer

Table 7.1 and 7.2 enlists the measured WER using aforementioned architectures on LibriSpeech test-clean and TED-LIUM v3 test sets. For conformer architecture, it is evident from Table 7.1 that our framework, leveraging from the knowledge distillation to train the dynamic student model, outperforms models trained without such knowledge transfer. We notice that RD_{sc} and RD_{LD} models are capable of adapting to varying resource settings, however, their performance degrades when evaluated with less or no dropping ($n_{\text{DS}} \geq 8$), which is superbly resolved by proposed DLD. We observe a trend of improved performance-computation trade-off on test splits of both datasets, in addition to retaining performance with low or no inference time dropping (2.25% and 1.67% less WER for $n_{\text{DS}} = 12$ without dropping any encoder module from respective baseline models). Moreover, for $n_{\text{DS}} = 6$, our framework achieves 2.35% and 0.89% improved WER with 2x computational speed-up on TED-LIUM v3 test split. Similarly, when trained on librispeech-100 split, our method outclass I3D (input-driven strategy) [91] that contains transformer-based encoder structure with deep feature extractor and triple number of encoder layers. They report a WER of ≈13.5% and ≈12.2% for dropping 50% and 25% of encoder modules for librispeech test-clean split, which is ≈ 3.56% and ≈ 4.06% higher than the proposed framework.

TABLE 7.1: Comparing WER (in %) of proposed framework on conformer architecture against RD based baseline methods when trained on LibriSpeech 1000. RD - random dropping, Params column depicts number of executed parameters.

n_{DS}	LibriSpeech			TEDLIUM v3			Params
	RD_{sc}	RD_{LD}	Ours	RD_{sc}	RD_{LD}	Ours	M
12	8.07	7.69	5.82	13.72	14.03	12.36	31.2
10	7.28	6.95	5.90	14.01	13.94	12.69	26.06
8	7.28	6.92	6.32	15.12	14.25	13.05	20.91
6	8.39	7.98	7.32	17.24	15.78	14.89	15.76
4	12.81	12.54	11.81	24.46	22.43	20.88	10.61
2	39.87	43.62	38.30	54.20	55.30	51.40	5.47
$\mathcal{M}_{ref}$		5.29			11.83		31.2

TABLE 7.2: Comparing measured WER (in %) for finetuning WavLM with RD with and without DLD framework.

n_{DS}	LibriSpeech		TEDLIUM v3		Params	Speed-up
	RD_{w-sc}	Ours	RD_{w-sc}	Ours	M	
12	5.47	4.57	10.32	9.19	94.40	1x
10	5.78	4.66	10.75	9.24	80.22	1.17x
8	6.52	5.20	12.53	10.55	66.04	1.43x
6	9.01	6.73	17.74	13.89	51.87	1.82x
4	17.59	12.76	31.90	24.61	37.69	2.50x
2	60.10	50.78	76.76	68.79	23.51	4.01x
$\mathcal{M}_{ref}$	3.5		10.12		94.40	1x

7.4.2 WavLM

To transform a static WavLM foundation model into a dynamic one, our framework significantly improves the performance-computation trade-off as illustrated in Table 7.2. For instance, we achieve 4.83% and 7.29% better WER on LibriSpeech and TEDLIUM datasets than the baseline with random dropping. Furthermore, our framework surpasses learnable masking method proposed in [28] by 0.37%, 4.57%, and 10.65% for dropping of 25%, 50%, and 75% encoder modules.

We observe from Figure 7.2 that [28] also provides reduction in model size, however, they employed Wave2Vec2 foundation model containing $317M$ parameters (3.37x more than ours) along with a bi-LSTM layer (linear projection in our case). The number of utilized parameters of their 75% trimmed model is similar to our full-sized model with substantially low performance (33.9% vs 4.57%). Figure 7.2 illustrates the variation in computational load and performance-computation trade-off between WavLM (ours) and wav2vec2 [28].

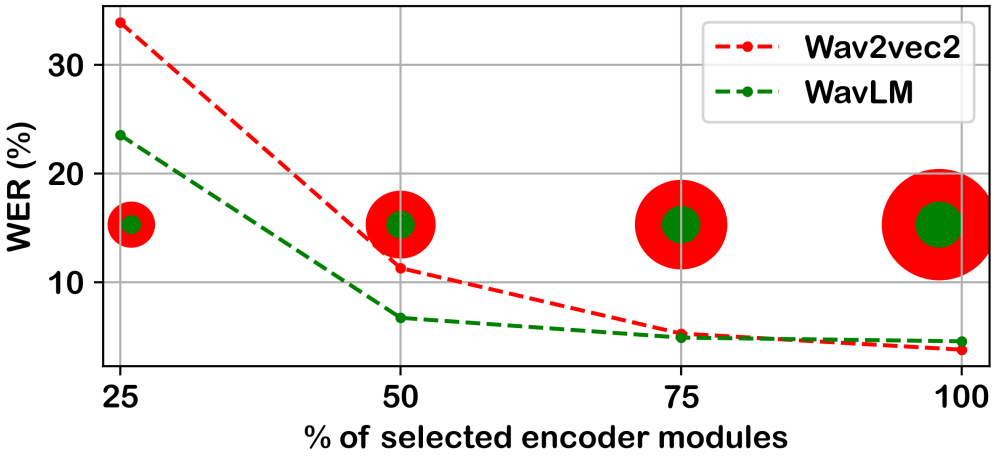

FIGURE 7.2: Comparing dynamic versions WavLM (random dropping based) vs Wav2Vec2 (learnable masking based). Circle depicts the utilized parameters for different encoder sizes.

TABLE 7.3: Evaluation of proposed framework on LibriSpeech test-clean split using Conformer model as training progresses.

n_{DS}	RD_{sc}	RD_{LD}	WER at Epoch			
			25	**50**	**75**	**100**
12	8.07	7.69	6.39	6.09	5.89	5.82
10	7.28	6.95	6.47	6.19	6.04	5.90
8	7.28	6.92	6.93	6.64	6.40	6.32
6	8.39	7.98	8.16	7.74	7.49	7.32
4	12.81	12.54	13.26	12.58	12.09	11.81
2	39.87	43.62	41.26	39.77	38.82	38.30

7.4.3 On performance degradation problem

Table 7.1 highlights that DLD framework resolves the inference time degradation problem present in the conformer baseline methods (RD_{sc} and RD_{LD} [41]). For $n_{DS} = \{12, 10, 8\}$, our framework scores WER of $\{5.82, 5.90, 6.32\}$ with minimal performance drop for no and low dropping values, whereas RD_{sc} and RD_{LD} gives WER of $\{8.07, 7.28, 7.28\}$ and $\{7.69, 6.95, 6.92\}$ respectively, depicting substantial performance degradation. We attribute this performance limitation of baseline methods to the shallow feature extractor which fails to provide rich-semantic representation for the input audio. Yet, our framework produces efficient results with two-third number of epochs of what are required to train the model without our framework from scratch. We also conjecture that increasing the depth of feature extractor can mitigate this issue as shown in the case of WavLM model (see Table 7.2) where our framework maintains its superiority over the baseline $RD_{w\text{-}sc}$ [42].

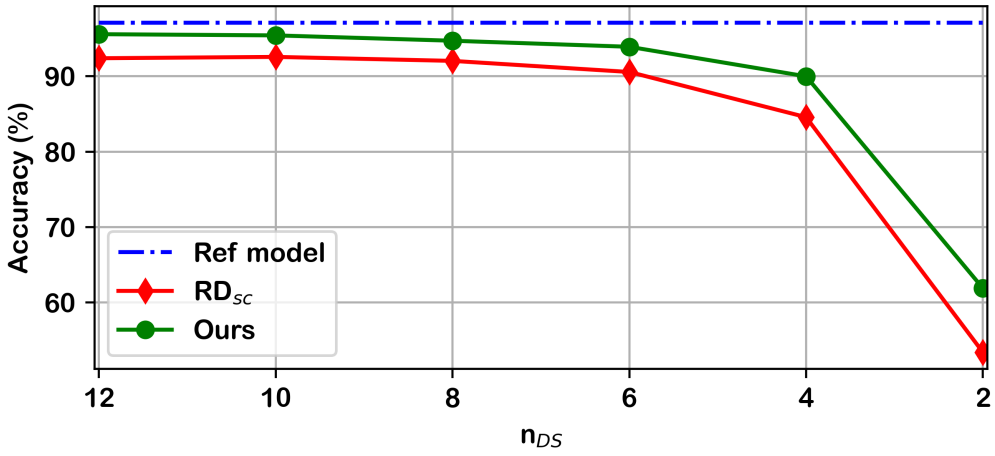

FIGURE 7.3: Evaluating generalization capability of our framework on spoken language understanding task using FSC dataset. Baseline Accuracy (with full model $\mathcal{M}_{\text{ref}} = 97.1\%$)

7.4.4 Framework performance evaluation as a function of training time

To establish the efficiency of proposed framework, we compute WER for different encoder sized models over the course of training, i.e. at epoch 25, 50, 75 and 100. Table 7.3 demonstrates that our framework surpasses the baseline models (referred as RD_{sc} and RD_{LD} in Table 7.1) at epoch 50 that is $\leq \frac{1}{3}$ of the training time required for the baseline methods. This also illustrates that embedding's alignment during training yields fast convergence and improves overall performance.

7.4.5 Generalization of proposed framework on other downstream tasks

To signify the generalizability of our approach, we adapted the conformer architecture for spoken language understanding task, and evaluated it on fluent speech commands (FSC) [81] dataset. For the reference model, we trained the conformer architecture from scratch achieving a baseline accuracy of 97.1% which is in-line with original paper. Similarly, we trained the conformer using RD with $p_d = 0.5$ and our KD-based framework. Figure 7.3 presents the accuracy for different value of n_{DS}, signifying that effectiveness of DLD is not limited to ASR downstream task only, rather it can be applied to different applications yielding improved performance.

Chapter 8

Concluding Remarks

8.1 Takeaways

This thesis investigates and advances the applicability of layer dropping method for different speech and audio tasks, especially automatic speech recognition. The contributions are made in two areas: (i) developing a semi-dynamic approach to achieve efficient lightweight models with reduced training time, (ii) transforming the static computational graph of a neural network to an adaptable and dynamic computational graph, enabling the neural network to work in a range of computational resources without re-training or re-compression.

The static compression techniques suffer to provide dynamic models that can operate in varying computational budget. In contrast, dynamic compression techniques result in models that can adapt to different computational budgets, however, such techniques offer different issues. Chapter 2 provides a review of the compression techniques along with their limitations.

In Chapter 4, we presented a semi-dynamic two-step framework addressing the limitation of compressing large models (using static compression methods) to fit inside a fixed computational budget with minimal performance degradation. As a first step, we train an encoder-only student model that mimics the representations of a teacher model using knowledge distillation. Once completed, a decoder is appended on the top of lightweight student model, and finetuning is performed to get the final student model. This training recipe enables faster training of lightweight models with minimal performance drop, and can be readjusted to decreased computational budget at a faster rate than previous state-of-the-art approaches. Moreover, the approach can be conditioned for other audio and speech applications with minimal efforts, enhancing the generalizability of the proposed framework.

In Chapter 5, we analyzed the dynamic compression of conformer architecture using random layer dropping ($\mathcal{LD}$). We worked to answer the following question: what should be the optimal value of dropping probability to use for $\mathcal{LD}$ during training to achieve best possible performance-computation trade-off ? To answer this question, we performed an exhaustive experimental analysis using different dropping probability values for $\mathcal{LD}$ during training, and recorded the model's performance for different levels of depth. We deduced that the optimal dropping probability during training is 0.5, and provides best performance at various model depths.

In Chapter 6, we focused on improving the performance-computation trade-off of random layer dropping by adding input-conditioning for layer dropping. We proposed a lightweight **layer selecting block** that takes the feature extractor's output and selects the most relevant encoder layers for that input. The main advantage

of our approach is that it can be used in plug-and-play fashion with different foundation models, without the need of layer-wise classifiers or computational graph modification.

In Chapter 7, we further push to improve the performance-computation trade-off using knowledge distillation in parallel with $\mathcal{LD}$. The proposed framework drives the dynamic model to mimic the teacher model's embeddings for all computational budgets, resulting in improved performance-computation trade-off. In addition, it solves the performance degradation issue where the dynamic model is conditioned to operate in varying resource scenario, struggles to maintain its performance even when it uses all the model layers. Typically, this issue is observed in models featuring shallow feature extractor failing to effectively capture the rich semantic features, whereas the issue is not present in models with deep feature extractor.

To generalize, this thesis addresses the dynamic computational compression of neural networks and conditions them to perform in varying computational budget. However, compression also involves reducing memory footprint for efficient storage which is not studied, and lays the future direction of this thesis. The next section explains the future works that will stem from this thesis.

8.2 Future directions

In this section, we propose future directions resulting in additional compression in terms of storage as well as further enhance the overall performance.

1. **Incorporating static compression:** As discussed in Chapter 2.1.2, quantization technique involves performing tensor operations with reduced precision leading to faster inference and reduced memory footprint. Additionally, quantization also results in performance deterioration requiring further finetuning of the architectures. Further experimentation will be conducted to incorporate the quantization in mechanisms discussed in Chapter 6 and 7, to speed-up the inference and reduced memory footprint. The resulting network will also become energy efficient, hence, becoming more suitable for edge devices.

2. **Towards non-euclidean learning:** Hyperbolic geometry has proven to better capture the hierarchical structure, better preserve the distances among embeddings, and improve representational capacity [85, 11, 47]. Recent works [47, 44] show an increase in using a hybrid of euclidean and hyperbolic space to train neural architectures, resulting in improved performance. We will extend the experimentation by incorporating hyperbolic geometry to distill information to the dynamic and adaptable network.

3. **Speaker Bias:** The dynamic architecture offers the structural flexibility, however, studies need to be carried out to investigate the number of layers selected for threshold-based automatic layer dropping scenario. It will provide insights into whether the model consistently selects less number of layers or it has traces of speaker, accent and gender related bias. This analysis will help to further finetune the training recipes, and boost overall performance.

4. **Towards mixed-precision layer dropping for multi-modality data:** This would be the ultimate goal of the work to develop a dynamic framework that will apply quantization-aware training in conjunction with layer dropping for speech recognition. In addition, we will incorporate parameter-efficient finetuning

(PEFT) methods to initially extend the model for different audio tasks, followed by adding other modalities for multi-modal downstream applications.

Bibliography

[1] Huseyin Abut, R Gray, and Guillermo Rebolledo. "Vector quantization of speech and speech-like waveforms". In: *IEEE transactions on acoustics, speech, and signal processing* 30.3 (2003), pp. 423–435.

[2] Ranya Aloufi, Hamed Haddadi, and David Boyle. "Privacy-preserving voice analysis via disentangled representations". In: *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*. 2020, pp. 1–14.

[3] Bishnu S Atal and Suzanne L Hanauer. "Speech analysis and synthesis by linear prediction of the speech wave". In: *The journal of the acoustical society of America* 50.2B (1971), pp. 637–655.

[4] Alexei Baevski et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations". In: *Advances in neural information processing systems* 33 (2020), pp. 12449–12460.

[5] Arian Bakhtiarnia et al. "Dynamic split computing for efficient deep edge intelligence". In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[6] Dan Berrebbi, Brian Yan, and Shinji Watanabe. "Avoid overthinking in self-supervised models for speech recognition". In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[7] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 535–541.

[8] Carlos Busso et al. "IEMOCAP: Interactive emotional dyadic motion capture database". In: *Language resources and evaluation* 42 (2008), pp. 335–359.

[9] Dongqi Cai et al. "SILENCE: Protecting privacy in offloaded speech understanding on resource-constrained devices". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 105928–105948.

[10] Jean Carletta et al. "The AMI meeting corpus: A pre-announcement". In: *International workshop on machine learning for multimodal interaction*. Springer. 2005, pp. 28–39.

[11] Ines Chami et al. "Hyperbolic graph convolutional neural networks". In: *Advances in neural information processing systems* 32 (2019).

[12] Guoguo Chen et al. "GigaSpeech: An Evolving, Multi-Domain ASR Corpus with 10,000 Hours of Transcribed Audio". In: *Interspeech 2021*. 2021, pp. 3670–3674. DOI: 10.21437/Interspeech.2021-1965.

[13] Sanyuan Chen et al. "Don't shoot butterfly with rifles: Multi-channel continuous speech separation with early exit transformer". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6139–6143.

[14] Sanyuan Chen et al. "Wavlm: Large-scale self-supervised pre-training for full stack speech processing". In: *IEEE Journal of Selected Topics in Signal Processing* 16.6 (2022), pp. 1505–1518.

[15] Zhourong Chen et al. "You look twice: Gaternet for dynamic filter selection in cnns". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9172–9180.

[16] Zewen Chi et al. "XLM-E: Cross-lingual language model pre-training via ELECTRA". In: *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*. 2022, pp. 6170–6182.

[17] Leigh Clark et al. "The state of speech in HCI: Trends, themes and challenges". In: *Interacting with computers* 31.4 (2019), pp. 349–371.

[18] Robert A Cohen, Hyomin Choi, and Ivan V Bajić. "Lightweight compression of neural network feature tensors for collaborative intelligence". In: *arXiv preprint arXiv:2105.06002* (2021).

[19] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "Binaryconnect: Training deep neural networks with binary weights during propagations". In: *Advances in neural information processing systems* 28 (2015).

[20] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[21] Zhen Dong et al. "Hawq-v2: Hessian aware trace-weighted quantization of neural networks". In: *Advances in neural information processing systems* 33 (2020), pp. 18518–18529.

[22] Zhen Dong et al. "Hawq: Hessian aware quantization of neural networks with mixed-precision". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 293–302.

[23] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[24] Mostafa Elhoushi et al. "Layerskip: Enabling early exit inference and self-speculative decoding". In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2024, pp. 12622–12642.

[25] Daniele Falavigna et al. "DNN adaptation by automatic quality estimation of ASR hypotheses". In: *Computer Speech & Language* 46 (2017), pp. 585–604.

[26] Angela Fan, Edouard Grave, and Armand Joulin. "Reducing Transformer Depth on Demand with Structured Dropout". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=SylO2yStDr.

[27] Takashi Fukuda et al. "Efficient knowledge distillation from an ensemble of teachers." In: *Interspeech*. 2017, pp. 3697–3701.

[28] David Genova, Philippe Esling, and Tom Hurlin. "Keep what you need: extracting efficient subnetworks from large audio representation models". In: *ICASSP*. 2025.

[29] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*. Vol. 159. Springer Science & Business Media, 2012.

[30] John J Godfrey, Edward C Holliman, and Jane McDaniel. "SWITCHBOARD: Telephone speech corpus for research and development". In: *Acoustics, speech, and signal processing, ieee international conference on*. Vol. 1. IEEE Computer Society. 1992, pp. 517–520.

[31] Ben Gold, Nelson Morgan, and Dan Ellis. *Speech and audio signal processing: processing and perception of speech and music*. John Wiley & Sons, 2011.

[32] Yuan Gong et al. "AST: Audio Spectrogram Transformer". In: *Interspeech*. 2021, pp. 571–575. DOI: 10.21437/Interspeech.2021-698.

[33] Jan Gorodkin et al. "A quantitative study of pruning by optimal brain damage". In: *International journal of neural systems* 4.02 (1993), pp. 159–169.

[34] Alex Graves et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *ICML*. 2006, pp. 369–376.

[35] A Gray and J Markel. "Quantization and bit allocation in speech processing". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 24.6 (2003), pp. 459–473.

[36] Robert M. Gray and David L. Neuhoff. "Quantization". In: *IEEE transactions on information theory* 44.6 (2002), pp. 2325–2383.

[37] Anmol Gulati et al. "Conformer: Convolution-augmented Transformer for Speech Recognition". In: *Interspeech 2020*. 2020, pp. 5036–5040. DOI: 10.21437/Interspeech.2020-3015.

[38] Song Han, Huizi Mao, and William J Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding". In: *arXiv preprint arXiv:1510.00149* (2015).

[39] Song Han et al. "Learning both weights and connections for efficient neural network". In: *Advances in neural information processing systems* 28 (2015).

[40] Wei Han et al. "ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context". In: *Interspeech 2020*. 2020, pp. 3610–3614. DOI: 10.21437/Interspeech.2020-2059.

[41] Abdul Hannan, Alessio Brutti, and Daniele Falavigna. "LDASR: An Experimental Study on Layer Drop using Conformer-based Architecture". In: *EUSIPCO*. 2024.

[42] Abdul Hannan, Daniele Falavigna, and Alessio Brutti. "Input Conditioned Layer Dropping in Speech Foundation Models". In: *2025 IEEE 35th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2025, pp. 1–6.

[43] Abdul Hannan et al. "An Effective Training Framework for Light-Weight Automatic Speech Recognition Models". In: *Interspeech 2025*. 2025, pp. 3613–3617. DOI: 10.21437/Interspeech.2025-1704.

[44] Abdul Hannan et al. "PAEFF: Precise Alignment and Enhanced Gated Feature Fusion for Face-Voice Association". In: *Interspeech 2025*. 2025, pp. 2710–2714. DOI: 10.21437/Interspeech.2025-268.

[45] Babak Hassibi, David G Stork, and Gregory J Wolff. "Optimal brain surgeon and general network pruning". In: *IEEE international conference on neural networks*. IEEE. 1993, pp. 293–299.

[46] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[47] Neil He et al. "Hyperbolic deep learning for foundation models: A survey". In: *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 2025, pp. 6021–6031.

[48] François Hernandez et al. "TED-LIUM 3: Twice as much data and corpus repartition for experiments on speaker adaptation". In: *SPECOM*. 2018, pp. 198–208.

[49] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[50] Wei-Ning Hsu et al. "Hubert: Self-supervised speech representation learning by masked prediction of hidden units". In: *IEEE/ACM transactions on audio, speech, and language processing* 29 (2021), pp. 3451–3460.

[51] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[52] Gao Huang et al. "Deep networks with stochastic depth". In: *European conference on computer vision*. Springer. 2016, pp. 646–661.

[53] Itay Hubara et al. "Quantized neural networks: Training neural networks with low precision weights and activations". In: *journal of machine learning research* 18.187 (2018), pp. 1–30.

[54] Sohei Itahara, Takayuki Nishio, and Koji Yamamoto. "Packet-loss-tolerant split inference for delay-sensitive deep learning in lossy wireless networks". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2021, pp. 1–6.

[55] Fumitada Itakura. "A statistical method for estimation of speech spectral density and formant frequencies". In: *Electro. Comm. Japan, A* 53.1 (1970), pp. 36–43.

[56] Fumitada Itakura. "Minimum prediction residual principle applied to speech recognition". In: *IEEE Transactions on acoustics, speech, and signal processing* 23.1 (2003), pp. 67–72.

[57] Benoit Jacob et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2704–2713.

[58] Frederick Jelinek. "Continuous speech recognition by statistical methods". In: *Proceedings of the IEEE* 64.4 (2005), pp. 532–556.

[59] Hyuk-Jin Jeong et al. "Computation offloading for machine learning web apps in the edge server environment". In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2018, pp. 1492–1499.

[60] Xiaoqi Jiao et al. "Tinybert: Distilling bert for natural language understanding". In: *Findings of the association for computational linguistics: EMNLP 2020*. 2020, pp. 4163–4174.

[61] B-H Juang. "Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains". In: *AT&T technical journal* 64.6 (1985), pp. 1235–1249.

[62] Biing-Hwang Juang and Lawrence R Rabiner. "Automatic speech recognition–a brief history of the technology development". In: *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara* 1.67 (2005), p. 1.

[63] Bing-Hwang Juang, Stephene Levinson, and M Sondhi. "Maximum likelihood estimation for multivariate mixture observations of markov chains (corresp.)" In: *IEEE Transactions on Information Theory* 32.2 (1986), pp. 307–309.

[64] Leonid V Kantorovich. "Mathematical methods of organizing and planning production". In: *Management science* 6.4 (1960), pp. 366–422.

[65] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. "Shallow-deep networks: Understanding and mitigating network overthinking". In: *International conference on machine learning*. PMLR. 2019, pp. 3301–3310.

[66] Sehoon Kim et al. "Learned token pruning for transformers". In: *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2022, pp. 784–794.

[67] Taehyeon Kim et al. "Comparing Kullback-Leibler Divergence and Mean Squared Error Loss in Knowledge Distillation". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. Main Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 2628–2635. DOI: 10.24963/ijcai.2021/362. URL: https://doi.org/10.24963/ijcai.2021/362.

[68] Young Jin Kim and Hany Hassan. "FastFormers: Highly Efficient Transformer Models for Natural Language Understanding". In: *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. Ed. by Nafise Sadat Moosavi et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 149–158. DOI: 10.18653/v1/2020.sustainlp-1.20.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[70] Taku Kudo and John Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing". In: *EMNLP*. Brussels, Belgium, Nov. 2018, pp. 66–71. DOI: 10.18653/v1/D18-2012.

[71] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[72] Yann LeCun, John Denker, and Sara Solla. "Optimal brain damage". In: *Advances in neural information processing systems* 2 (1989).

[73] Stephen E Levinson, Lawrence R Rabiner, and M Mohan Sondhi. "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition". In: *Bell System Technical Journal* 62.4 (1983), pp. 1035–1074.

[74] Andong Li et al. "Learning to inference with early exit in the progressive speech enhancement". In: *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE. 2021, pp. 466–470.

[75] Guangli Li et al. "Auto-tuning neural network quantization framework for collaborative inference between the cloud and edge". In: *International Conference on Artificial Neural Networks*. Springer. 2018, pp. 402–411.

[76] Jianhua Lin. "Divergence measures based on the Shannon entropy". In: *IEEE Transactions on Information theory* 37.1 (2002), pp. 145–151.

[77] Richard P Lippmann. "Review of neural networks for speech recognition". In: *Neural computation* 1.1 (1989), pp. 1–38.

[78] Chao Liu, Zhiyong Zhang, and Dong Wang. "Pruning deep neural networks by optimal brain damage." In: *Interspeech*. Vol. 2014. 2014, pp. 1092–1095.

[79] Xuan Liu, Di Cao, and Kai Yu. "Binarized lstm language model". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 2113–2121.

[80] Yiping Lu et al. "Understanding and improving transformer from a multi-particle dynamic system point of view". In: *arXiv preprint arXiv:1906.02762* (2019).

[81] Loren Lugosch et al. "Speech Model Pre-Training for End-to-End Spoken Language Understanding". In: *Interspeech*. ISCA, 2019, pp. 814–818. DOI: 10.21437/INTERSPEECH.2019-2396. URL: https://doi.org/10.21437/Interspeech.2019-2396.

[82] Rao Ma, Qi Liu, and Kai Yu. "Highly efficient neural network language model compression using soft binarization training". In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE. 2019, pp. 62–69.

[83] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. "Split computing and early exiting for deep learning applications: Survey and research challenges". In: *ACM Computing Surveys* 55.5 (2022), pp. 1–30.

[84] Warren S McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[85] Maximillian Nickel and Douwe Kiela. "Poincaré embeddings for learning hierarchical representations". In: *Advances in neural information processing systems* 30 (2017).

[86] Daniele Jahier Pagliari et al. "Crime: Input-dependent collaborative inference for recurrent neural networks". In: *IEEE Transactions on Computers* 70.10 (2020), pp. 1626–1639.

[87] Panayotov et al. "Librispeech: an asr corpus based on public domain audio books". In: *ICASSP*. IEEE. 2015.

[88] Daniel S. Park et al. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: *Interspeech 2019*. 2019, pp. 2613–2617. DOI: 10.21437/Interspeech.2019-2680.

[89] Sree Hari Krishnan Parthasarathi and Nikko Strom. "Lessons from building acoustic models with a million hours of speech". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 6670–6674.

[90] David Peer et al. "Greedy-layer pruning: Speeding up transformer models for natural language processing". In: *Pattern Recognition Letters* 157 (2022), pp. 76–82.

[91] Yifan Peng, Jaesong Lee, and Shinji Watanabe. "I3D: Transformer architectures with input-dependent dynamic depth for speech recognition". In: *ICASSP*. IEEE. 2023, pp. 1–5.

[92] Yifan Peng et al. "Structured pruning of self-supervised pre-trained models for speech recognition and understanding". In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[93] Cal Peyser et al. "Towards disentangled speech representations". In: *arXiv preprint arXiv:2208.13191* (2022).

[94] Karol J. Piczak. "ESC: Dataset for Environmental Sound Classification". In: *Proceedings of the 23rd Annual ACM Conference on Multimedia*. Brisbane, Australia, Oct. 13, 2015. ISBN: 978-1-4503-3459-4. DOI: 10.1145/2733373.2806390. URL: http://dl.acm.org/citation.cfm?doid=2733373.2806390.

[95] Antonio Polino, Razvan Pascanu, and Dan Alistarh. "Model compression via distillation and quantization". In: *arXiv preprint arXiv:1802.05668* (2018).

[96] Yan-min Qian and Xu Xiang. "Binary neural networks for speech recognition". In: *Frontiers of Information Technology & Electronic Engineering* 20.5 (2019), pp. 701–715.

[97] L Rabiner et al. "Speaker-independent recognition of isolated words using clustering techniques". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 27.4 (2003), pp. 336–349.

[98] Mohammad Rastegari et al. "Xnor-net: Imagenet classification using binary convolutional neural networks". In: *European conference on computer vision*. Springer. 2016, pp. 525–542.

[99] Hassan Sajjad et al. "On the effect of dropping layers of pre-trained transformer models". In: *Computer Speech & Language* 77 (2023), p. 101429.

[100] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).

[101] Simone Scardapane et al. "Why should we add early exits to neural networks?" In: *Cognitive Computation* 12.5 (2020), pp. 954–966.

[102] Jonghyeon Seon et al. "Stop or forward: Dynamic layer skipping for efficient action recognition". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 3361–3370.

[103] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[104] Daniel Soudry, Itay Hubara, and Ron Meir. "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights". In: *Advances in neural information processing systems* 27 (2014).

[105] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[106] Ryoichi Takashima, Li Sheng, and Hisashi Kawai. "Investigation of sequence-level knowledge distillation methods for CTC acoustic models". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 6156–6160.

[107] Raphael Tang et al. "Temporal early exiting for streaming speech commands recognition". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 7567–7571.

[108] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. "Branchynet: Fast inference via early exiting from deep neural networks". In: *2016 23rd international conference on pattern recognition (ICPR)*. IEEE. 2016, pp. 2464–2469.

[109] Hugo Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[110] Leonid Nisonovich Vaserstein. "Markov processes over denumerable products of spaces, describing large systems of automata". In: *Problemy Peredachi Informatsii* 5.3 (1969), pp. 64–72.

[111] Ashish Vaswani et al. "Attention is All you Need". In: *NeurIPS*. Ed. by I. Guyon et al. Vol. 30. 2017.

[112] Andreas Veit and Serge Belongie. "Convolutional networks with adaptive inference graphs". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–18.

[113] Taras K Vintsyuk. "Speech discrimination by dynamic programming". In: *Cybernetics* 4.1 (1968), pp. 52–57.

[114] Li Wan et al. "Regularization of neural networks using dropconnect". In: *International conference on machine learning*. PMLR. 2013, pp. 1058–1066.

[115] Alex Wang et al. "GLUE: A multi-task benchmark and analysis platform for natural language understanding". In: *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*. 2018, pp. 353–355.

[116] Changhan Wang et al. "VoxPopuli: A Large-Scale Multilingual Speech Corpus for Representation Learning, Semi-Supervised Learning and Interpretation". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 993–1003. DOI: 10.18653/v1/2021.acl-long.80. URL: https://aclanthology.org/2021.acl-long.80/.

[117] Kuan Wang et al. "Haq: Hardware-aware automated quantization with mixed precision". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8612–8620.

[118] Xiaofei Wang et al. "Convergence of edge computing and deep learning: A comprehensive survey". In: *IEEE communications surveys & tutorials* 22.2 (2020), pp. 869–904.

[119] Xin Wang et al. "Skipnet: Learning dynamic routing in convolutional networks". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 409–424.

[120] Pete Warden. "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition". In: *CoRR* abs/1804.03209 (2018). arXiv: 1804.03209. URL: http://arxiv.org/abs/1804.03209.

[121] Shinji Watanabe et al. "Student-teacher network learning with enhanced features". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 5275–5279.

[122] Zeyuan Wei, Li Hao, and Xueliang Zhang. "Model Compression by Iterative Pruning with Knowledge Distillation and Its Application to Speech Enhancement." In: *INTERSPEECH*. 2022, pp. 941–945.

[123] George August Wright et al. "Training dynamic models using early exits for automatic speech recognition on resource-constrained devices". In: *arXiv preprint arXiv:2309.09546* (2023).

[124] Zhanghao Wu et al. "Lite transformer with long-short range attention". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. 2020. URL: https://openreview.net/forum?id=ByeMPlHKPH.

[125] Zhaofeng Wu et al. "Dynamic sparsity neural networks for automatic speech recognition". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6014–6018.

[126] Zuxuan Wu et al. "Blockdrop: Dynamic inference paths in residual networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8817–8826.

[127] Qiao Xiao et al. "Dynamic data pruning for automatic speech recognition". In: *arXiv preprint arXiv:2406.18373* (2024).

[128] Ji Xin et al. "BERxiT: Early exiting for BERT with better fine-tuning and extension to regression". In: *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: Main Volume*. 2021, pp. 91–104.

[129] Ji Xin et al. "DeeBERT: Dynamic early exiting for accelerating BERT inference". In: *arXiv preprint arXiv:2004.12993* (2020).

[130] Jingjing Xu et al. "Dynamic encoder size based on data-driven layer-wise pruning for speech recognition". In: *arXiv preprint arXiv:2407.18930* (2024).

[131] Junhao Xu et al. "Mixed precision low-bit quantization of neural network language models for speech recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3679–3693.

[132] Ji Won Yoon, Beom Jun Woo, and Nam Soo Kim. "Hubert-ee: Early exiting hubert for efficient speech recognition". In: *arXiv preprint arXiv:2204.06328* (2022).

[133] Ji Won Yoon et al. "Inter-KD: Intermediate knowledge distillation for CTC-based automatic speech recognition". In: *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2023, pp. 280–286.

[134] Ji Won Yoon et al. "Tutornet: Towards flexible knowledge distillation for end-to-end speech recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 1626–1638.

[135] Kai Yu et al. "Neural network language model compression with product quantization and soft binarization". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), pp. 2438–2449.

[136] Li Yuan et al. "Revisiting knowledge distillation via label smoothing regularization". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 3903–3911.

[137] Salah Zaiem et al. "Fine-tuning strategies for faster inference using speech self-supervised models: a comparative study". In: *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*. IEEE. 2023, pp. 1–5.

[138] Ke Zhang et al. "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks". In: *IEEE access* 4 (2016), pp. 5896–5907.

[139] Minjia Zhang and Yuxiong He. "Accelerating training of transformer-based language models with progressive layer dropping". In: *Advances in neural information processing systems* 33 (2020), pp. 14011–14023.

[140] Kai Zhen et al. "Sparsification via compressed sensing for automatic speech recognition". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 6009–6013.