Dynamically Efficient Self-Supervised Speech Pre-Training with Mixture-of-Depths

Qinyi Li

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2025

Abstract

The self-supervised pre-training fine-tuning paradigm has been widely adopted for the development of state-of-the-art speech models, due to its ability to effectively leverage unlabelled data to improve model performance. However, such models usually require large amounts of resources and extensive time during both training and inference, especially during the self-supervised pre-training stage. This causes slow iteration time for researchers, and makes it impossible to deploy such models to edge devices with constrained resources.

I propose to alleviate this problem by using a dynamic neural architecture at the self-supervised pre-training stage to facilitate both efficient training and inference. Specifically, I explore Mixture-of-Depths (Raposo et al., 2024), a method that learns to dynamically skip select layers conditioned on the input. Applying this technique effectively during the pre-training stage instead of during fine-tuning avoids the need of having to train the dynamic neural architecture separately for each downstream task the model is applied to and improves development efficiency.

A comprehensive evaluation of efficiency for the proposed approach is conducted during the self-supervised pre-training stage, measuring both theoretical efficiency and practical runtime reductions across different settings. Then, the performance of the approach is evaluated on two downstream tasks, investigating the capability of pre-trained model to effectively extract useful content and speaker information from speech.

Experiments find that applying Mixture-of-Depths during self-supervised pre-training in speech significantly improves efficiency while only incurring minor impacts on downstream task performance, achieving an effective efficiency-performance trade-off. The most efficient Mixture-of-Depths model achieves 43.66% theoretical time reduction over the baseline and up to 41.65% training time reduction and 41.30% inference time reduction in practice. Compared to the baseline, the same model achieves an increase of 1.54% in Phone Error Rate during phone classification and a decrease of 2.60% in classification accuracy during speaker identification.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Qinyi Li*)

Acknowledgements

First of all, I would like to thank my supervisor, Hao Tang, for all his guidance and mentorship over the past two years. Working on my two projects with him has truly been among the best and most rewarding experiences of my undergraduate studies. I think one's understanding of the enigmatic nature of research is shaped largely by the person who first introduces them to it. More than anything, I'm truly glad that it's this entrancing perspective I was introduced to at the beginning of my research journey and saw through for the past two years.

Thank you to my friends, for being my anchors and partners in crime through my greatest coming-of-age adventures far from home. For even when you didn't know, you have been speckles of ember lighting my way, kindling my courage to continue forward.

Thank you to the music school piano practice rooms, which have surely witnessed all the highs and lows of my undergraduate studies and granted me utopia between hammer, string, and notes. Thank you to everyone who was a part of my beginner table tennis journey this past year, for encouraging my passion to another hobby that I will love through victory and defeat. Thank you to Bach, whose polyphony is the voice of calmness on my most restless days, and to aespa, whose dissonance resonates with my headspace during the most stormy ones.

Lastly, to my family, for whom my gratitude may be best expressed not in words but as a feeling: an inherent comfort that, no matter the distance, home is always right behind should I ever stumble or stray.

Table of Contents

Chapter 1

Introduction

Given the nature of speech as the most natural communication medium, speech systems such as automatic speech recognition and voice synthesis have become vital to facilitating seamless interaction between humans and computers. With the rapid rise of deep learning over the past decade, it has become state-of-the-art to leverage deep learning techniques for speech recognition (Gulati et al., 2020; Radford et al., 2023; Communication et al., 2023). The self-supervised pre-training fine-tuning paradigm has been widely adopted across the training of large speech models (Baevski et al., 2020; Hsu et al., 2021; Chen et al., 2022), due to their ability to leverage the vast amount of unlabelled data towards improving model performance, countering the problem of sparse labelled data in speech. As state-of-the-art self-supervised speech models requiring large amounts of compute resources to both train and deploy, improving their efficiency and reducing resource requirements has become a topic of focus in speech research (Ahlawat et al., 2025).

1.1 Motivations

In the current day and age, mobile devices such as laptops and smartphones are the most common access point to speech systems. However, most of them lack the compute power required to run these large state-of-the-art speech models, requiring either compression in order to develop locally hosted solutions, or the compromise for a cloud hosted solution, which induces both higher latency unacceptable for critical applications, as well as privacy concerns. Furthermore, scaling laws (Sun et al., 2017; Hestness et al., 2017; Kaplan et al., 2020) dictates that the larger the model and/or the training dataset - the more compute resources expended in training - the better its performance. Thus arises the contradiction researchers have been striving to navigate: balancing model efficiency while maintaining performance and accuracy.

Under the widely adopted self-supervised pre-training fine-tuning paradigm, much of this bottleneck in efficiency lies in the pre-training part of model training, where the model undergoes representation learning that can transfer across to various downstream tasks. Hence, improving efficiency on the pre-training side while maintaining matching downstream performance is an important problem many have been aiming to tackle.

There has been much effort contributed towards the efficient training and inference of large neural network models in general, and there are many methods to which such problems can be alleviated: optimisations of the input and output data, of the model architecture, or at the system level, an example being distributed training (Zhou et al., 2024). In this thesis, I investigate optimising for efficient training and inference in the model architecture model with dynamic neural networks.

1.2 Preceding Work (MInf Part 1)

Both Part 1 and Part 2 of my MInf project work towards the purpose of alleviating the problem described above. Progressing from Part 1 to Part 2, I maintain the same objective, investigating efficient self-supervised speech pre-training by adapting modifications to the pre-trained model architecture, aiming to improve both training and inference efficiency while preserving performance in downstream tasks.

While both Part 1 and Part 2 work towards the same objective, they differ in the exact approaches explored. In Part 1 of my MInf project (Li, 2024), I leverage the stochastic drop-path mechanism from the FractalNet (Larsson et al., 2017) to improve pre-training efficiency. Training efficiency was achieved through the drop-path mechanisms of the FractalNet, by stochastically choosing a subset of layers from the network to update for each training batch. At inference time, FractalNet was able to adapt to different resource availabilities by selecting a suitable subnetwork from its collection of diverse subnetworks with varying depths. All subnetworks achieved an effective efficiency-performance trade-off compared to respective baselines.

The FractalNet is efficient during training, and able to achieve varying degrees of inference efficiency by choosing subnetworks of different depths. However, it applies different computations to inputs during training stochastically and applies uniform computation during inference, and I believe a more effective efficiency-performance trade-off can be achieved if the network learns to dynamically adjust its architecture depending on the input. Hence, in Part 2, I explore Mixture-of-Depths (Raposo et al., 2024). Where training and inference efficiency were optimised by probabilistic selection of layers to update in Part 1, it is now improved by learned routing of individual frames in an utterance through the model. A similar evaluation procedure is followed in this thesis: the model is evaluated for efficiency during pre-training, and subsequently for performance in downstream tasks that evaluates the effective extraction of content and speaker information from speech.

1.3 Contributions

In this thesis, I explore the application of Mixture-of-Depths (Raposo et al., 2024) in self-supervised speech pre-training, and its effectiveness in achieving efficiency while preserving performance. The contributions made are as follows:

1. First application of a layer-skipping dynamic neural architecture to enable efficient self-supervised pre-training in speech. First adoption of the Mixture-of-Depths

(Raposo et al., 2024) method to speech.

2. Discovers that the sigmoid router activation can be optimised from Mixture-of-Depths. Removing the sigmoid router activation provided better performance for all except the largest capacity factor.
3. Analyses both theoretical and practical efficiency of Mixture-of-Depths operating with a dynamic computation graph on non-uniform sequence lengths. I specifically find that the efficiency of Mixture-of-Depths is highly dependent on specific hardware in the environment
4. Verifies that Mixture-of-Depths preserves both content and speaker information in its latent features, and investigates the effect of capacity on performance.
5. Show that Mixture-of-Depths learns a specialised parameter distribution and reveals patterns in frames processed and skipped by each Mixture-of-Depths layer
6. While Mixture-of-Depths can perform zero-resource generalisation to different capacity factors at inference, removing the router altogether is infeasible.

1.4 Thesis Outline

Proceeding from this chapter, the thesis will be structured as follows:

- **Chapter 2: Background.** I will provide context on two of the main relating fields of research: self-supervised learning and efficient neural networks.
- **Chapter 3: Methodology Overview.** I detail the 2 main approaches I adapt from literature in this chapter. Then I specify the Baseline and Mixture-of-Depths model sets I leverage in experiments, and present the 2 main experiments, each serving to evaluate one side of the efficiency-performance trade-off I investigate.
- **Chapter 4: Pre-Training Experiment.** I present a comprehensive examination on the efficiency of Mixture-of-Depths across differing capacity factors. I analyse the theoretical efficiency, and measure practical efficiency across different setups. Most notably, I find the runtime reduction Mixture-of-Depths achieves in practice is highly correlated with the hardware environment.
- **Chapter 5: Downstream Tasks Experiment.** Extending upon my results on efficiency in the previous chapter, I move on to investigate the performance end of the efficiency-performance trade-off. I explore performance of Mixture-of-Depths on standard benchmark tasks, analysing two important and orthogonal dimensions of speech: content and speaker (Yang et al., 2021).
- **Chapter 6: Ablation studies.** I select a few intriguing aspects from Chapter 4 and Chapter 5 conduct further investigations. I analyse how routing decisions made by Mixture-of-Depths layers correspond to acoustic and linguistic characteristics, explore zero-resource generalisability of Mixture-of-Depths at inference time to other capacity factors, and examine the effectiveness of different Mixture-of-Depths model configurations.

Chapter 2

Background

To provide a comprehensive context for my investigation into efficient self-supervised pre-training, this chapter builds a foundation for closely related topics in the current research landscape. I will start by discussing speech representations, the transition from engineered acoustic features to self-supervised features extracted from foundation models, and why improving the efficiency of such models has become a focus for researchers. Next, I move on to consider approaches that facilitate efficiency in neural network models, reviewing current approaches and presenting the current rising direction in dynamic neural networks. Finally, I identify gaps in the current literature and lay the foundation for the research question my work aims to address.

2.1 Speech Representations: From Engineered to Self-Supervised

This section presents a progression of speech representations used in speech processing systems, discussing how cutting-edge methods have evolved from leveraging acoustic to self-supervised latent features, and the latest challenges in improving the efficiency of feature extraction from speech foundation models.

2.1.1 Engineered Acoustic Features

The physical expression of speech comprises variations in air pressure in a continuous manner. To digitise such a continuous acoustic signal, changes in air pressure over time are captured by measuring the force it exerts on the diaphragm of a microphone, converting it into electrical signals that can subsequently be processed by computers.

The digitisation of speech involves two fundamental settings: sampling rate and bit depth. As capturing all measurements in a continuous interval over time is infinite and computationally intractable, the continuous acoustic signal must be discretised by sampling at a predetermined frequency, namely the sampling rate. Each recorded discrete electrical signal must be stored in binary format, and the number of bits assigned to express the value of each signal is the bit depth (Wayland, 2018).

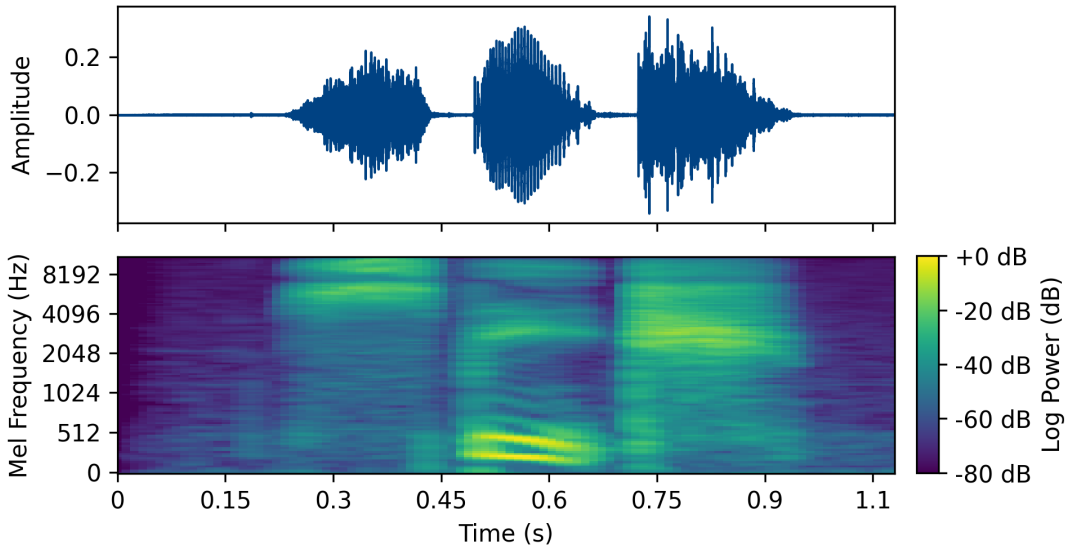

Figure 2.1: **Waveform and log-mel spectrogram.** Sampled waveform and the extracted log-mel spectrogram of a recording of the word “speech”. The waveform is a representation of variation in air pressure over time, and the spectrogram shows mel frequencies obtained from STFT against time. The colorbar to the right of the spectrogram shows the relation between the log power in dB and the colormap used.

After digitisation, the speech signal is now a discrete waveform representing variations in air pressure over time (top of Figure 2.1). However, during early research in speech processing, such waveforms were not favourable representations of speech. When it comes to speech recognition, the waveform representation is dense and contains much redundant information irrelevant towards the discrimination required for speech recognition tasks (Hinton et al., 2012). These waveforms were usually transformed from the time domain into the frequency domain to formulate a more compact representation. Irrelevant information such as phase is discarded while useful patterns on the frequency spectrum are exposed. Two of the most frequently utilised representations are Mel-frequency cepstral coefficients (MFCCs) and log-mel spectrograms (bottom of Figure 2.1), both of which result from a transformation of the waveform into the frequency domain via short-term Fourier transform (STFT).

Traditionally, after the feature extraction process described above, a label is applied to each acoustic feature, and GMM-HMM-based models, or, as is the convention more recently, deep neural network models, are employed to learn the underlying knowledge required to correctly generate these labels by maximising log likelihood. This process of learning a predictive mapping between input features and their corresponding output labels is named supervised learning, and it has enabled many applications in speech, including speech recognition, synthesis, and emotion recognition.

2.1.2 Self-Supervised Latent Features

The problem that underlies the supervised learning approach in speech is its strict requirement for labelled data, while the speech domain has always faced the problem of having sparse labelled data.

Annotating speech utterances - the unlabelled input data - is complex and time-consuming. Phonetic transcriptions require specific domain knowledge in speech, and even when such knowledge is not required in annotations such as text transcription, challenges still arise from the inherent characteristics of speech. Applying a label to speech is often ambiguous due to the continuous nature of speech frames and their dependence upon neighbouring frames, and it is difficult to distinguish the exact start and end of a specific label (Mohamed et al., 2022). This makes annotating unlabelled data cumbersome and time-consuming, leading to a lack of such data compared to other domains.

The lack of labelled data poses a bottleneck to the performance of speech models. The size of the training dataset is logarithmically related to the scaling of model performance (Sun et al., 2017), and this is a significant limitation for low-resource language applications, where the scarcity of labelled data is amplified.

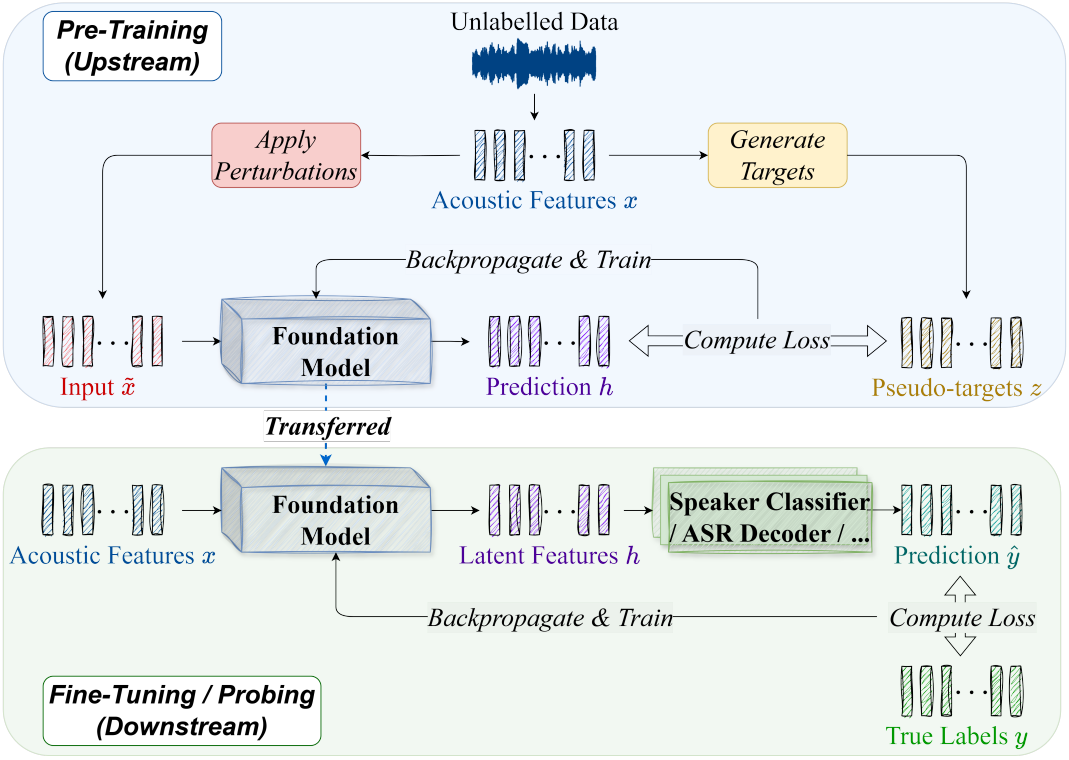

Figure 2.2: **An illustration of the self-supervised pre-training fine-tuning pipeline.** *Top: Pre-Training.* The same unlabelled data is leveraged to generate input and targets used train a foundation model, which learns useful internal structures of speech. *Bottom: Fine-Tuning/Probing.* The pre-trained foundation model is transferred to various downstream tasks. Latent features extracted from the foundation model is passed to a downstream model, and the foundation model could be trained jointly with the downstream model (fine-tuning), or leveraged as a frozen feature extractor (probing).

The self-supervised pre-training fine-tuning paradigm is a breakthrough for tackling this challenge in that it is able to effectively leverage the vast amount of unlabelled data available in speech to improve model performance, by building a foundation model that is generalisable across different downstream tasks, and subsequently only a small set of labelled data is required to further fine-tune performance for each task. It effectively sidesteps the challenge of sparse labelled data in speech. Semi-supervised learning is another technique that leverages unlabelled data towards improving performance of speech models, however, it is usually not task-agnostic and renders improvement towards one specific task instead of multiple downstream tasks as is possible in self-supervised learning (Mohamed et al., 2022).

Early self-supervised methods in speech mainly consisted of generative and contrastive approaches (Mohamed et al., 2022). Generative approaches aim to reconstruct the data using the foundation model, taking as input perturbed versions of the data. For example, Autoregressive Predictive Coding (APC) by Chung et al. (2019) aims to predict future frames based frames in the past, and Masked Predictive Coding (MPC) by Jiang et al. (2019) applies a zero-mask to random frames, and aims to reconstruct the original speech utterance from the masked version. Contrastive approaches such as Contrastive Predictive Coding (CPC) (van den Oord et al., 2018) and wav2vec 2.0 (Baevski et al., 2020) are characterised by their loss function, which maximises similarity between the prediction of the foundation model with a "positive" sample, while maximising dissimilarity with "negative" samples. In recent years, predictive approaches have become popular. Predictive approaches such as HuBERT (Hsu et al., 2021) and WavLM (Chen et al., 2022) use a learned model to generate pseudo-targets. wav2vec 2.0 (Baevski et al., 2020) and HuBERT (Hsu et al., 2021) in particular were especially successful, both countering the problem of speech being continuous and the prediction may be trivial by discretising the pseudo-targets.

2.1.3 Challenges in Efficient Self-Supervised Learning

The breakthrough of the Transformer network architecture, initiated by Vaswani et al. (2017), led to models whose performance could be successfully scaled with model size. The speech research community has also largely taken to the Transformer model, due to its effectiveness in handling long-range dependencies compared to RNN-like architectures, which struggle to maintain dependencies over longer distances, due to sequential processing and information loss. Specifically, it is the most prominent architecture employed in self-supervised foundation models today.

However, such monolithic models are composed of millions and billions of parameters. For example, WavLM Large has 316.63M parameters (Chen et al., 2022), while HuBERT X-Large has 1B (Hsu et al., 2021). They consume huge amounts of memory and compute, establishing the need for efficient models during both training and inference. On the training side, hardware required for training SOTA models is no longer manageable for many smaller corporations or research institutions. Additionally, on the inference side, the deployment of such models to consumer devices is impossible due to the resources they consume.

With the large impact and domination of the self-supervised pre-training paradigm,

detailed previously in Section 2.1.2, improving the efficiency of self-supervised pre-training approaches in speech becomes an imminent problem to tackle, and it is what researchers have shifted focus to and have been largely working towards in recent years. Developing efficient techniques for self-supervised pre-training in the speech domain can be especially challenging. Computation reduction is implemented during the self-supervised pre-training stage, where the model improves efficiency while minimising loss to a pretext task instead of a concrete downstream application. However, maintaining pretext task performance may not translate to realistic downstream applications, as downstream speech tasks usually require many different, sometimes even orthogonal characteristics (Liu et al., 2023; Mohamed et al., 2024) to be learned and extracted from latent features of the pre-trained foundation model, and this is challenging to navigate.

Despite challenges, many researchers have succeeded in simplifying and improving the efficiency of self-supervised pre-training approaches. Most of these works focus on optimising the HuBERT architecture: Lin et al. (2023) explore simplified input data representation and loss function through MelHuBERT, Chang et al. (2022) explore distillation of the HuBERT model, and Lin et al. (2024a) investigate compatibility of an early-exit technique.

2.2 Towards Efficient and Dynamic Neural Architectures

With the goal of looking for efficient self-supervised learning in mind, I divert my attention to the landscape of efficient neural network training and inference. One common approach to improve the efficiency of a model is through modifying the underlying neural network architecture, and this is the approach I will adopt in my experiments. There are a few families of techniques towards this purpose, such as quantization, pruning, knowledge distillation, and compact architecture design (Wang et al., 2024) - and more recently, the family of dynamic neural architectures.

2.2.1 Introducing Dynamic Neural Architectures

The topic of dynamic neural architectures, which is a subset of techniques within the design paradigm of dynamic neural networks (Han et al., 2022), has garnered much research interest over the years. It has been most extensively researched in vision, followed by language, with fewer applications extending to speech. Despite the recent surge in interest, this idea has a deep-rooted history within the field of deep learning, with concepts of Mixture-of-Experts first proposed by Jacobs et al. (1991) in 1991, while Bengio (2013); Bengio et al. (2015) re-introduced the concept of conditional computation more recently.

In the context of dynamic neural architectures, most neural network models would be considered static: they apply the same set of transformations from each layer to all inputs, whereas not all inputs may require all transformations to make an accurate prediction. Considering the speech domain, an utterance recorded in a professional recording booth would require less computation compared to an utterance recorded in a noisy environment such as a shopping mall - the latter would require specific speech enhancement processing, and applying this to the former is not only redundant, but may

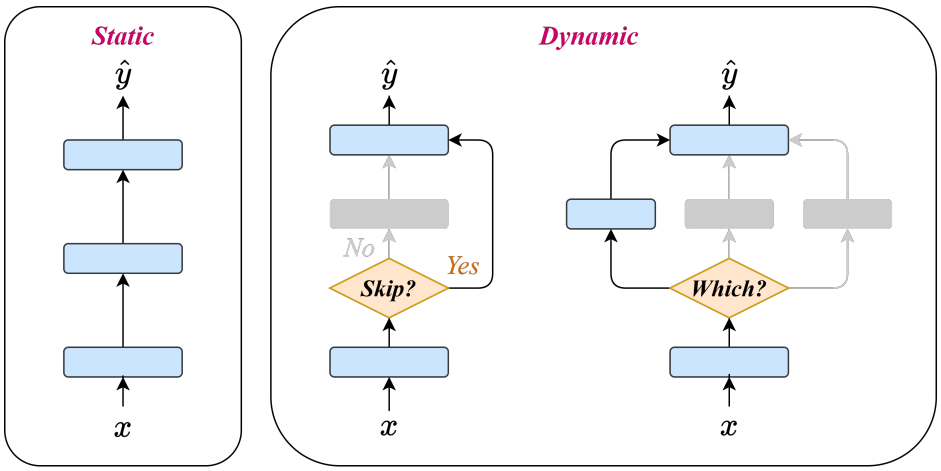

Figure 2.3: **Demonstration of a static neural architecture vs. possible dynamic neural architectures.** *Left: Static.* The same set of layers is applied to all inputs. *Right: Dynamic.* Depending on the input, the network may decide to skip the next layer, or choose from a set of layers for the current sample.

also degrade performance. Additionally, different inputs could benefit from different and more specialised transformations.

The design of dynamic neural architectures is a family of techniques that trains models designed to adapt their structure, or more specifically their computation graph, to each input during inference. Usually, only a subset of the model is activated, depending on the input. For example, a model could decide to skip a certain layer, or choose from a set of layers, as depicted in Figure 2.3.

This approach provides many benefits, and the ability to effectively navigate the trade-off between efficiency and accuracy is one of the most notable. With adaptive and dynamic architectures, models can recognise when certain transformations are not required and improve efficiency, as well as when they are essential, preserving accuracy. Additionally, a dynamic model can learn to recognise different classes within the input space that benefit from different sets of transformations and route these inputs to specialised processing units to maximise performance. For example, speech with a noisy background might require a specialised enhancement unit, while speech recorded with the speaker far away from the microphone may require amplification. As only a subset of the model is activated for each input, this gives dynamic neural architectures better representation power and a larger parameter space compared to their static counterparts.

Furthermore, the actual implementation of such techniques is straightforward and accessible. Dynamic neural architectures are compatible with most existing static architectures, and can easily generalise to a wide range of applications. Most techniques proposed are high-level approaches that add lightweight decision components to the architecture to enable dynamic behaviour, such as a gating network, and they can be easily applied to similar architectures. Additionally, deciding how the model architecture adapts to each input can be either learned from training or adjusted with tuning, making them generalisable to different training objectives and different tasks.

2.2.2 Dynamic Depth for Training Efficiency

While inference efficiency is achieved for all categories of dynamic neural architectures: dynamic depth, dynamic width, and dynamic routing (Han et al., 2022), in practice, often only dynamic depth methods are used to facilitate efficient training.

Dynamic depth techniques generally consist of two designs: early exit or layer-skipping, illustrated in Figure 2.4.

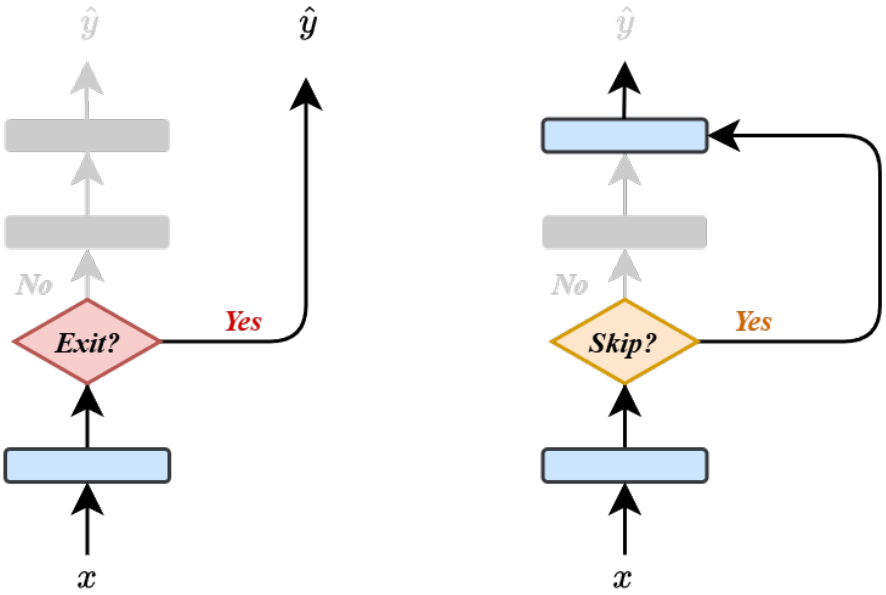

Figure 2.4: **Demonstration dynamic depth methods: early exit and layer-skipping.** *Left: Early Exit.* At designated exit points, the network decides whether to terminate computation and output the current prediction, or to continue. *Right: Layer-Skipping.* At a decision point, the network makes a decision between whether to continue or skip the next layer.

In models with early exit designs, multiple exit points are implemented in the model, similar to the one depicted on the left of Figure 2.4. Through training, the model is encouraged to learn at each exit point the best choice between continuing to deeper layers, or terminating computation and outputting the current intermediate representation. The principle behind this design is that by enabling the choice to terminate model inference at hidden layers, computation can be saved on data that does not require the additional transformations, and declines in performance caused by redundant complexity or noise introduced in later transformations (Kaya et al., 2019) could be avoided. The exit points could be placed after each layer in a network (Zhou et al., 2020), or between each network in a cascade of networks (Bolukbasi et al., 2017). Generally, early exit techniques can be efficiently trained, as decisions at each exit point are lightweight and threshold-based: if a computed confidence score for the current intermediate representation exceeds a pre-defined threshold, the model will decide to output a prediction at that exit point. There exists 2 different approaches to compute the confidence scores: (i) using a fixed function, such as entropy (Teerapittayanon et al., 2016), or (ii) using

a learned linear classifier (Xin et al., 2021). However, the output of linear classifiers is susceptible to small perturbations (Jiang et al., 2018), hence, some methods use a combination of confidence scores from multiple linear classifiers to determine the exit of an intermediate representation (Zhou et al., 2020).

In layer-skipping, usually a gating network is trained at each layer to decide whether the input will proceed to the next layer, or skip the next layer. The implementation the gating network usually dictates whether efficient training is facilitated. In many layer-skipping approaches, the training of the gating network occurs after the training of the underlying model, incurring more training cost (Wang et al., 2018). Usually efficient training for layer-skipping methods is achieved either through using a confidence-based criteria like is usually employed in early exit techniques, or by employing a differentiable gating unit, where the gating network is trained alongside the underlying model.

In my experiments, I leverage a layer-skipping technique with a differentiable gating network: Mixture-of-Depths proposed by Raposo et al. (2024).

2.3 Gaps in Existing Literature

Despite demonstrating a potential to improve both training and inference efficiency, there have only been a few studies exploring the adaptation of dynamic neural architectures to speech, and even fewer that investigate applying such methods during the self-supervised pre-training stage. Particularly, to the best of my knowledge, there have been no similar attempts to adapt a layer-skipping technique such as Mixture-of-Depths (Raposo et al., 2024) to self-supervised pre-training in speech.

Out of the existing literature, only DAISY (Lin et al., 2024a) applies a dynamic depth technique to self-supervised pre-training in speech. Lin et al. (2024a) demonstrates that early exit techniques can be trained during the self-supervised pre-training stage, and successfully enable subsequent generalisation to a variety of downstream tasks. Most work investigates applying dynamic approaches during fine-tuning for actual downstream tasks. The problem that underlies such approaches is that because the dynamic neural architecture is adapted during downstream fine-tuning, separate training of the dynamic neural architecture is required for each downstream task, incurring extra training cost, compared to training the dynamic technique during pre-training, which is more efficient. HuBERT-EE (Yoon et al., 2022) applies the early exit technique to improve efficiency during fine-tuning of the pre-trained HuBERT (Hsu et al., 2021) model. You et al. (2021) applies Mixture-of-Experts to train a speech recognition model, but the purpose for application of such a dynamic neural architecture is not for efficiency.

To bridge this gap, in this thesis, I propose to adapt a layer-skipping technique, Mixture-of-Depths (Raposo et al., 2024), to the self-supervised pre-training stage, and demonstrate that it is able to improve both efficiency during pre-training, and be effective when the model is transferred to different downstream tasks that require different characteristics to be extracted by the model, without further training of the dynamic neural architecture required.

Chapter 3

Methodology Overview

Now that the current research landscape has been established, I outline my overarching methodology and experimental design in this chapter for all experiments hereafter. First, I provide a detailed description of methods I adopt from literature. Next, I introduce the baseline for my experiments and the experiment models for which I will make comparisons. Then, I motivate the two different evaluations I will be making through two different experiments in Chapters 4 and 5 respectively, and the purpose they each serve. Finally, I will set up the hypotheses I aim to test throughout the rest of this thesis.

3.1 Proposed Approaches

In this section, I provide an overview of the methods I draw inspiration from and adopt in my experiments. I will start by elaborating how exactly each method relates to my experiments, and then situate them using taxonomies introduced in Chapter 2. Although I make modifications to the approaches in my experiments, descriptions in this section will follow the original implementations of the methods exactly as from literature.

3.1.1 Masked Predictive Coding

Masked Predictive Coding (MPC), proposed by Jiang et al. (2019), is the pre-training task I adopt in my pre-training experiment.

As detailed in Section 2.1.2, MPC belongs to the generative family of pre-training tasks, where the pre-trained model is provided with some perturbed version of the unlabelled data and aims to recover the original uncorrupted version. In the case of MPC, the corruption is applying a zero mask to certain positions of the input utterance with probability p. This approach draws inspiration from previous Masked Language Modelling (MLM) objective of models employed in language, such as BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019).

However, as the authors of MPC discovered, as speech is naturally continuous and locally smooth, masking individual frames within an utterance with probability p provided unsatisfactory results, with no improvements over the baseline without pre-

training. To address this, the authors made modifications to the standard masking strategy used in MLM objectives for the MPC method. Frames are first downsampled in groups of 8 to prevent trivial local frame masking, and then a mask is applied probabilistically to each downsampled group of frames, facilitating a wider masking window and encouraging the model to learn non-trivial structures.

3.1.2 Mixture-of-Depths

The Mixture-of-Depths approach, proposed by Raposo et al. (2024), is the experiment model architecture I employ to facilitate efficient pre-training. I make comparisons between Mixture-of-Depths models and a baseline concerning computation efficiency during pre-training (Chapter 4), and performance during downstream tasks (Chapter 5). Mixture-of-Depths is considered the layer-skipping variant of dynamic depth techniques, where the model makes a decision at certain layers to either skip or pass through that layer, as detailed in Section 2.2.2. In Mixture-of-Depths, the gating network, or the router, is trained simultaneously with the underlying transformer decoder model, facilitating efficient training as well as inference.

The component which characterises a Mixture-of-Depths transformer (decoder) model is the incorporation of Mixture-of-Depths ***"blocks"*** within, which I will refer to as Mixture-of-Depths ***"layers"*** instead throughout this thesis. In their work, Raposo et al. (2024) find that the optimal configuration for the Mixture-of-Depths model is to intersperse a Mixture-of-Depths layer every 2 layers, and implementing a transformer network consisting only of Mixture-of-Depth layers was found to be suboptimal. Specifically, the Mixture-of-Depths layer will be the first layer of every 2.

A Mixture-of-Depths layer, illustrated in Figure 3.1, is a regular transformer decoder layer with the addition of a router at the beginning. The router determines for each token in the input sequence whether it will be processed by the Mixture-of-Depths layer, or passed through a residual connection that skips to the next layer in the model. The router Raposo et al. (2024) implement is a linear layer with no bias, a learned matrix W^ℓ mapping each input token to a scalar, followed by a sigmoid activation for the output weights.

Raposo et al. (2024) applies Mixture-of-Depths to a dataset of sequences with uniform length n. For every Mixture-of-Depths layer in the model, a uniform capacity $k < n$ is determined before training begins, where the capacity k is the number of tokens from each sequence that will be routed through every Mixture-of-Depths layer.

At the beginning of every Mixture-of-Depths layer, router weights are computed based for each input. Suppose there is a Mixture-of-Depths layer at hidden layer ℓ, where $x^\ell = (x_1^\ell \ldots x_n^\ell)$ is the input to layer ℓ with n tokens, and each token x_i^ℓ is a d-dimension embedding, such that $x^\ell \in \mathbb{R}^{n \times d}$. A sigmoid scalar router weight is computed for each token in the sequence through a learned linear transformation $W_\theta^\ell \in \mathbb{R}^d$ by:

$$r^\ell = \text{sigmoid}(x^\ell W_\theta^\ell) \tag{3.1}$$

The scalar router weights is subsequently used to determine which tokens will be

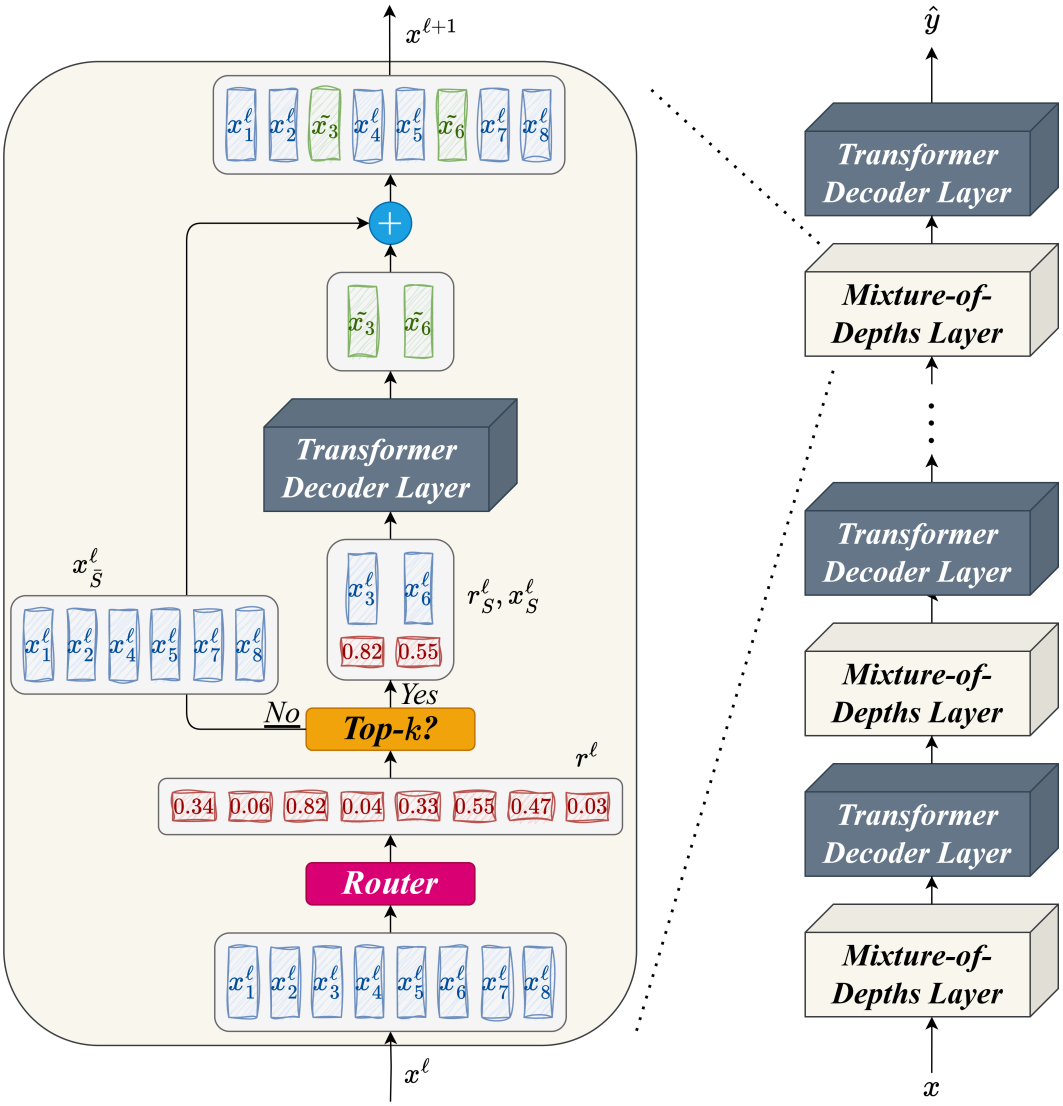

Figure 3.1: **Mixture-of-Depths model architecture.** *Left: Mixture-of-Depths Layer.* Shows the path of input x^ℓ through a Mixture-of-Depths layer. x^ℓ first passes through the Router, generating r^ℓ. Then, r^ℓ is used to determine whether each x_i^ℓ is processed by the current layer. At the end, the processed and unprocessed tokens are composed back together in their original order. *Right: Mixture-of-Depths Model.* A Mixture-of-Depths Layer is interspersed every two layers in the Mixture-of-Depths model, and it occurs first out of every two layers.

processed by the transformer decoder layer at layer ℓ, denoted f. Tokens x_i^ℓ with corresponding scalar router weight r_i^ℓ that is within the Top-k largest router weights is processed by the Mixture-of-Depths layer, denoted by $x_S^\ell = \{x_i^\ell : r_i^\ell \in \text{Top-}k(r^\ell)\}$. Otherwise, x_i^ℓ will be routed through a residual connection that passes around the transformer decoder layer f at Mixture-of-Depths layer ℓ. In the end, the output $x^{\ell+1} = (x_1^{\ell+1} \ldots x_n^{\ell+1})$ will be:

$$x_i^{\ell+1} = \begin{cases} r_i^\ell f(x_S^\ell) + x_i^\ell & \text{if } r_i^\ell \in \text{Top-}k(r^\ell) \\ x_i^\ell & \text{otherwise} \end{cases} \tag{3.2}$$

Related Work

There has been previous work applying a similar routing approach on the token level within a transformer. For example, as Raposo et al. (2024) mention in their paper, CoLT5 (Ainslie et al., 2023) also leverages conditional computation within a transformer. In CoLT5, at the attention and feed-forward components of every layer, a router decides whether a token requires extra computation: if the token should pass through a "light" attention/feed-forward component, or both a "light" and "heavy" feed-forward component. In contrast to CoLT5, Mixture-of-Depths implements a simplified approach that offers the option to either pass through or skip a transformer layer.

Regarding the application of the Mixture-of-Depths technique, there has been related work adapting Mixture-of-Depths to vision and multimodal models (Lin et al., 2024b), but no approaches yet for speech. Applying Mixture-of-Depths to speech require some adjustments, as speech utterances don't have a uniform length, requiring dynamic capacity adjustments. I will detail these adjustments in the following section.

3.2 Experiment Models: Baseline and Mixture-of-Depths Modifications

As Mixture-of-Depths is the experiment architecture I aim to investigate, in this section, I describe the exact Mixture-of-Depths models I will evaluate and the baseline I will make comparisons to through experiments in Chapter 4 and Chapter 5. Additionally, I explain a few modifications I make to the original Mixture-of-Depths setup described in Section 3.1.2.

Base Architectural Configurations For both the Baseline and Mixture-of-Depths model, I employ a 12-layer transformer encoder, with architectural hyperparameters $d_{model} = 256$, $d_{feedforward} = 2048$, and $d_{head} = 4$.

Mixture-of-Depths Modifications In original Mixture-of-Depths, Raposo et al. (2024) applied the model to a dataset consisting of text sequences of uniform length. I apply Mixture-of-Depths to speech, however, and utterances in a speech dataset are always non-uniform due to their continuous nature. Therefore, in my experiments, it is not possible to have a fixed capacity k. In this case of dynamic sequence lengths, I maintain instead a fixed ***capacity factor*** c and calculate a dynamic top-k. I calculate a dynamic capacity k for each batch by $k = n \cdot c$, based on the fixed capacity factor c and the longest sequence/utterance length n in the batch, and choose top-k frames from each utterance in that batch to process. All datasets used in my experiments are sorted by length and grouped into batches by batch size. The batches are then shuffled and randomly sampled during training. Batching by sorted sequence length ensures that a reasonable dynamic

capacity is applied to each sequence in the batch, while robust training is maintained by shuffling the batches.

Exploring Activations and Offsets Raposo et al. (2024) originally employs a Mixture-of-Depths model with sigmoid router weight activation, which will be referred to throughout this thesis as ***"activation"***. A Mixture-of-Depths layer applied every 2 layers is found to be the best configuration, and Raposo et al. (2024) apply the Mixture-of-Depths layer first out of every 2 layers. I name this configuration setting ***"offset"***. If a Mixture-of-Depths layer occurs first every 2 layers, it is defined to have a Mixture-of-Depths offset of 0; if it occurs at the second of every 2 layers, it has offset 1.

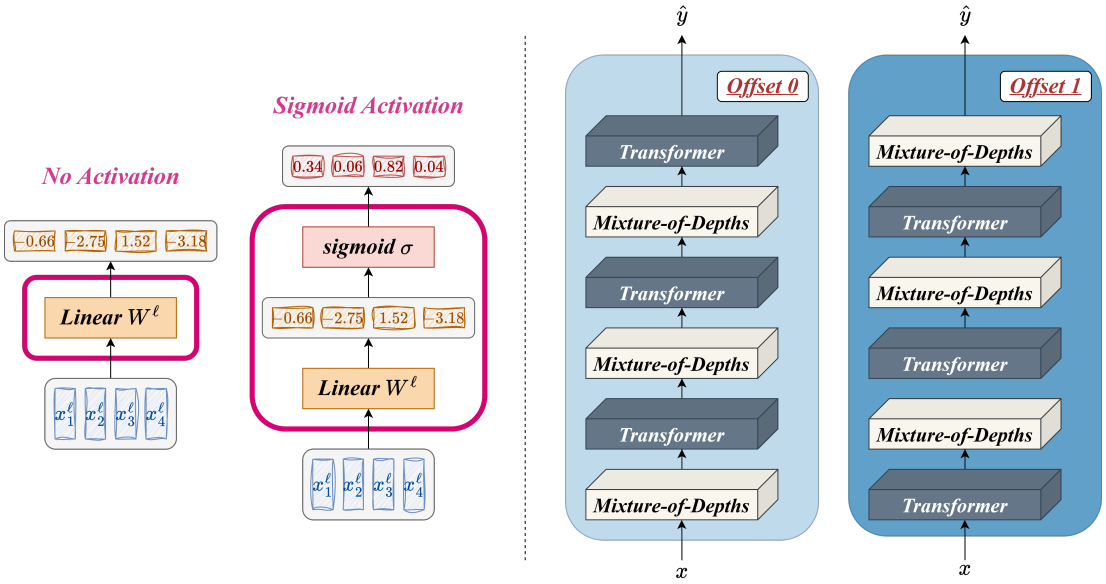

Figure 3.2: **Different "Activation" and "Offset" settings experimented.** *Left:* Two different router activations settings employed, with and without the nonlinear sigmoid. *Right:* Two different offset settings employed, "Offset 0" puts the Mixture-of-Depths layer first out of every 2 layers, while "offset 1" puts the Mixture-of-Depths layer second.

I experimented with varying the activation and offset configurations for the Mixture-of-Depths models, resulting in 4 different sets:

1. Sigmoid router activation, offset 0 ***(implementation in Raposo et al. (2024))***
2. Sigmoid router activation, offset 1
3. No router activation, offset 0
4. No router activation, offset 1 ***(Best Set)***

In each set, I also explore the effect of different capacity factors, to facilitate a more fine-grained analysis on different capacity factor settings in Mixture-of-Depths. I choose 0.125 (the best capacity factor reported by Raposo et al. (2024)), 0.25, 0.5, and 0.75. For a consistent comparison, I compare the baseline to one set of Mixture-of-Depths models throughout the pre-training and downstream experiments. Performance varies provided different configurations and capacity factors, but provided with the aim to improve

efficiency in this thesis, I identify a "Best Set" that is the most efficiency-effective: it is most effective at smaller capacity factors. A comparison of performance between the "Best Set" and all 4 different sets, especially the configuration proposed by Raposo et al. (2024), can be found in Section 6.3.

3.3 Evaluations: Pre-Training and Downstream

Through upcoming experiments, there are two main questions I aim to investigate:

(1) **How much efficiency does Mixture-of-Depths provide?** Towards answering this question, I evaluate pre-training efficiency in Chapter 4. In the pre-training experiment, I aim to investigate both theoretical efficiency gains and wall-clock runtime savings in practice, and use the pre-training loss only as an indicative measure of whether models are effectively training. Theoretical efficiency will be evaluated by counting FLOPs on all of the models, and as theoretical efficiency does not take into account many real-world factors, including parallel execution in PyTorch, practical efficiency will be measured using wall-clock runtime. Specifically, I measure both training and inference times, across 3 different training environments, to gain a more comprehensive understanding of the impact of hardware on the practical runtime of Mixture-of-Depths.

(2) **What is the trade-off in performance for these efficiency gains?** As pre-training loss is not an accurate measurement of performance, I investigate the performance of models by evaluating their accuracy on downstream tasks. Specifically, I evaluate for the preservation of orthogonal characteristics in speech (Mohamed et al., 2022) with two tasks: phone classification for local phonetic information in speech, and speaker identification for global speaker information. In the downstream tasks, I aim to conduct a layer-wise analysis, comparing the hidden representation extracted at each layer from Mixture-of-Depths models to each layer from the Baseline model.

3.4 Hypotheses

Towards my experiments, I make the following hypotheses:

- **Hypothesis 1 (H1):** ***Efficiency.*** *Lower capacity factors will lead to both higher theoretical and practical efficiency. Practical efficiency may be tapered due to additional operations in the router and can vary depending on hardware.*
- **Hypothesis 2 (H2):** ***Performance.*** *Mixture-of-Depths models have comparable performance to the Baseline across downstream tasks. Higher capacity factor will lead to better models.*
- **Hypothesis 3 (H3):** ***Parameter Distribution.*** *The parameter distribution learned by Mixture-of-Depths models and the Baseline will be fundamentally different.*
- **Hypothesis 4 (H4):** ***Inference-Time Generalisability.*** *Mixture-of-Depths will adapt well to different capacity factors during inference.*

Chapter 4

Pre-Training: Evaluation of Computation Efficiency

This chapter details the pre-training experiment I conduct, aiming to evaluate the computation efficiency achieved by Mixture-of-Depths models from the best set with differing capacity factors. I detail the exact setup of my experiment, briefly discuss the pre-training loss, then evaluate efficiency from two perspectives: theoretical efficiency using FLOPs, and practical efficiency using wall-clock time.

4.1 Experiment Setup

Dataset I utilise the LibriSpeech corpus (Panayotov et al., 2015) as the unlabelled dataset to pre-train all models. LibriSpeech is commonly used for self-supervised pre-training in speech, for example, in SOTA methods such as wav2vec 2.0 (Baevski et al., 2020) and HuBERT (Hsu et al., 2021). All models are trained on the "clean" pool from the 100-hour subset in LibriSpeech (`train-clean-100`). As the 100-hour and 360-hour subsets from LibriSpeech are disjoint, I use a sample extracted from the clean pool of the 360-hour subset (`train-clean-360`) as the validation set. The training set consists of 28,539 utterances, and the validation set consists of 2,854 utterances.

Data pre-processing For each utterance sample in the dataset, I extract 40-dimension log-mel spectrograms with a 25ms window and 10ms shift, and apply global normalisation, in line with the approach of Yang et al. (2022). I compute the mean and variance across all utterances for each of the training and validation sets, and each utterance is normalised by the mean and variance of their respective sets. I adopt the downsampling optimization from MelHuBERT by Lin et al. (2023), concatenating every 2 contiguous frames sequentially and resulting in 80-dimension spectrogram-based feature frames and an effective 20ms shift between each feature frame.

Pre-training task settings The pre-training task I adopt is Masked Predictive Coding (Jiang et al., 2019), as previously detailed in Section 3.1.1. The model is encouraged to reconstruct the original utterance from zero-masked corruptions. Masked prediction

pre-training tasks are frequently adopted in SOTA self-supervised pre-trained models in speech (Baevski et al., 2020; Hsu et al., 2021). I adopt the masking configurations from work by Lin et al. (2023). Whereas Jiang et al. (2019) downsampled frames in an utterance in groups of 8 to prevent learning local trivial solutions, I employ a masking strategy where a decision is made at each frame to mask the next 5 frames with a 14% probability. This mask strategy is implemented with the `compute_mask_indices` function from Ott et al. (2019).

Models I make comparisons of the baseline to the "best set" of Mixture-of-Depths models in this experiment, as detailed in Section 3.2. The baseline model is a 12-layer transformer encoder, while the set of Mixture-of-Depths models (no activation, offset 1) consists of 4 models, investigating capacity factors 0.125, 0.25, 0.5, and 0.75 applied to the Mixture-of-Depths layers. I train all models for 200 epochs on learning rate $1e^{-4}$, with a batch size of 8 and apply dropout with probability 0.1. I manually set the same random seed for each of PyTorch, NumPy and Python's random library at the beginning of all experiments.

Evaluation metrics This experiment evaluates both the theoretical and practical efficiency of the Mixture-of-Depths set of models during pre-training. Theoretical efficiency of the model set measured in FLOPs, using the `FlopsCountAnalysis` function from the `fvcore` library. Practical efficiency is surveyed by measuring wall-clock training and inference time across 3 different hardware environments. Pre-training loss is used as an indication of effective training in the models rather than an accurate performance measure, and Mean-Squared Error (MSE) loss is used. The pre-training task encourages the model to learn useful relations within speech and does not possess grounded realistic applications, and evaluation of model performance will be conducted in Chapter 5, where I measure performance on downstream tasks.

4.2 Pre-Training Loss

I briefly evaluate the training and validation loss across the Baseline and Mixture-of-Depths models to confirm that the models are training effectively using the MPC pre-training task.

The overall trend of all models across 200 epochs is presented in Figure 4.1. From the figure, all Mixture-of-Depths models match the pre-training loss of the Baseline almost exactly throughout the 200 epochs in both training and validation. The Mixture-of-Depths models with lower capacity factors ($c = 0.125$ and $c = 0.25$) can be observed to have higher losses during the first epochs, but all models quickly converge.

The exact final training and validation loss at the end of the 200 epochs is detailed in Table 4.1. Once again, it can be observed that all of the values align very closely, and there appears to be a trend of lower loss with higher capacity factors. More notably, the Mixture-of-Depths model with $c = 0.75$ improves over both the training and validation loss observed on the baseline, while the Mixture-of-Depths model with $c = 0.5$ improves over the validation loss of the baseline.

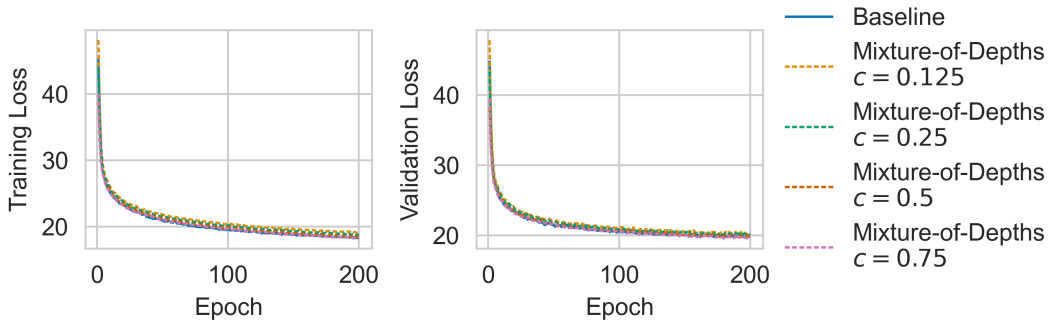

Figure 4.1: **Training and validation loss of Baseline and Mixture-of-Depths models across 200 epochs.** *Left:* Training loss. *Right:* Validation loss.

Model	Training Loss	Validation Loss
Baseline	18.30	19.89
Mixture-of-Depths $c = 0.125$	19.08	20.23
Mixture-of-Depths $c = 0.25$	18.76	20.05
Mixture-of-Depths $c = 0.5$	18.43	19.86
Mixture-of-Depths $c = 0.75$	18.29	19.70

Table 4.1: **Training and validation loss at epoch 200 (final epoch) across Baseline and Mixture-of-Depths models.** All values are rounded to 2 decimal places.

4.3 Theoretical Efficiency via FLOPs

FLOPs Computation I measure theoretical efficiency in FLOPs on the training set. For all models, the total FLOPs expended in one epoch iterated over the training set is computed, and then divided by the total number of frames across all utterances in the dataset to obtain a measurement of FLOPs expended per frame.

4.3.1 FLOPs Results

Model	FLOPs Per Frame	Reduction (%)
Baseline	15,803,333	0%
Mixture-of-Depths $c = 0.125$	8,903,357	43.66%
Mixture-of-Depths $c = 0.25$	9,889,318	37.42%
Mixture-of-Depths $c = 0.5$	11,861,182	24.95%
Mixture-of-Depths $c = 0.75$	13,829,912	12.49%

Table 4.2: **Measured FLOPs per frame for each model, and percentage reductions achieved by Mixture-of-Depths models compared to the Baseline.** All values are rounded to 2 decimal places.

The exact FLOPs measurement and the percentage reduction are presented in Table 4.2. All Mixture-of-Depths models exhibit considerable FLOPs compute reduction over the

baseline, ranging from a 43.66% reduction in FLOPs expended per frame for the model with the smallest capacity factor $c = 0.125$, to 12.49% for capacity factor $c = 0.75$.

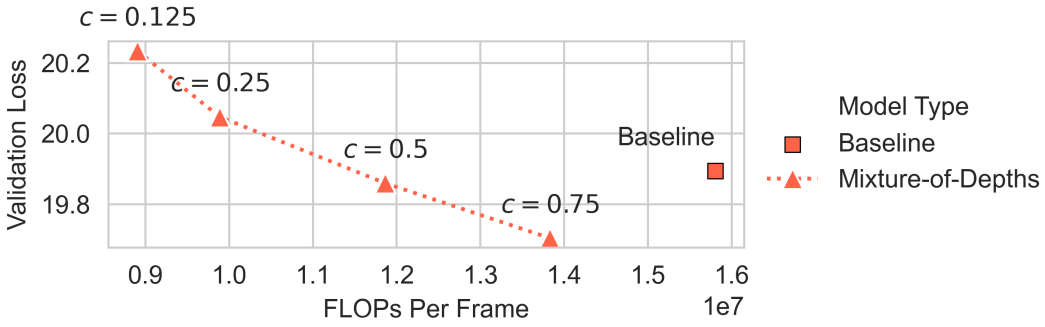

Figure 4.2: **Visualisation of FLOPs reductions across Mixture-of-Depths and Baseline models.** The validation loss at the end of 200 epochs is shown along the y-axis.

More interesting observations can be made from Figure 4.2. Here, the FLOPs expended per frame exhibits an inversely linear relation to the final validation loss at the end of 200 epochs. It is made clear again from this figure, that Mixture-of-Depths models with $c = 0.5$ and $c = 0.75$ learns the pre-training task better than the Baseline with lower FLOPs, while models with $c = 0.125$ and $c = 0.25$ may not match the validation loss of the Baseline, but they're incredibly efficient in the loss-FLOPs trade-off achieved. Furthermore, the relation between the capacity factor and FLOPs reduction gained in this experiment closely resembles $1 - \frac{6c+6}{12}$, shown in Figure 4.3, strongly suggesting a linear relation between the capacity factor and the theoretical compute reduction.

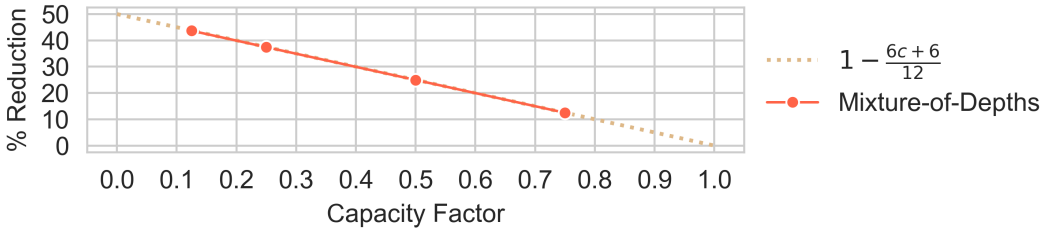

Figure 4.3: **Visualisation of the hypothesised relation between capacity factor and the % reduction achieved in theory.** The hypothesis appears to overlap the relation observed exactly.

4.3.2 Limitations of FLOPs Measurement

Despite its widespread employment and theoretical soundness, the FLOPs measurement has its limitations, most arising from the fact that it does not take into account many real-world factors that could impact the actual runtime in practice. For example, the FLOPs reduction computed could only be achieved considering a sequential execution of all FLOPs operations, and doesn't take into account the asynchronous and parallel executions of PyTorch. Additionally, it only measure the compute expended on the

forward pass of the models, and does not measure differences during backpropagation, which could be different due to the difference in model architectures employed. Furthermore, FLOPs operations are distributed in practice across both the CPU and GPU of the device, and depending on the exact hardware setup, this may also have impact on the practical runtime.

Therefore, in the next section, I provide a complementary perspective on the efficiency of Mixture-of-Depths models by measuring their wall-clock runtime in practice. Considering all of the factors mentioned above, the runtime across different hardware environments is measured for a more detailed analysis.

4.4 Practical Efficiency via Wall-Clock Time

4.4.1 Hardware Environments

I report the wall-clock runtime on 3 different hardware environments in this experiment, namely **Cluster**, **Desktop**, and **Laptop**.

Warm-up Process For each model, a uniform warm-up process is implemented, as the Baseline model has been observed to be heavily dependent on a sufficient warm-up. For example, when running in the Laptop environment, the Mixture-of-Depths model with $c = 0.125$ shows a 90% reduction in runtime during the first epoch compared to the Baseline model without warm-up, while it is about 40% after warm-up. This also aligns with practical training scenarios, where the model is trained over a larger number of epochs. Before recording runtimes, the model first iterates once over the unshuffled training dataset, then, it is trained and I measure runtime over multiple epochs.

Environment Setup The exact setting of each measurement is detailed below:

(1) **Cluster.** I measure the runtime on the university's MLP compute cluster, a network of compute nodes each equipped with multiple CPUs and GPUs. The runtime reported for each model is the average over 200 epochs of training.

(2) **Desktop.** The second measurement is conducted on a desktop computer. The runtime on this setup is more stable as there is more control over environmental factors, hence an average over 10 epochs of training is reported.

(3) **Laptop.** Finally, I measure runtime on a GPU-equipped laptop. For this setup I also report the averaged runtime over 10 epochs.

The CPU and GPU specifications for each environment are detailed in Table 4.3, as they are variables that have an impact on the efficiency in practice.

In summary, ranking the environments by CPU capability, the ordering is

$$\text{Laptop} > \text{Desktop} \gg \text{Cluster}$$

Ranking by GPU capability, the exact reverse ordering is obtained

$$\text{Cluster} \gg \text{Desktop} > \text{Laptop}$$

	CPU Specs		GPU Specs		
	Model	Cores/Threads	Model	Memory	TFLOPs
Cluster	Xeon E5-2603 v4	6/6	A6000	48GB	38.71
Desktop	Core i5-14500	14/20	RTX 4060	8GB	15.11
Laptop	Core i9-13900HX	24/32	RTX 4060 Mobile	8GB	11.61

Table 4.3: **CPU and GPU specifications across training environments Cluster, Desktop, and Laptop.**

4.4.2 Training Wall-Clock Time

The relation between model runtimes during training and final validation loss across different training environments is depicted in Figure 4.4. I observe largely the same trend in the practical measurements as before when theoretical efficiency was analysed in Section 4.3.1. Across all 3 environments, all Mixture-of-Depths models achieve higher efficiency compared to the Baseline model. Comparing different environments, it is evident that the Cluster setup had the fastest runtimes, followed by Desktop, followed by Laptop. This aligns with the order of GPU capability across the environments and is expected as neural network training is largely dependent on GPU computations. And it only makes sense that the absolute runtime reductions are smallest on the Cluster, followed by Desktop and Laptop.

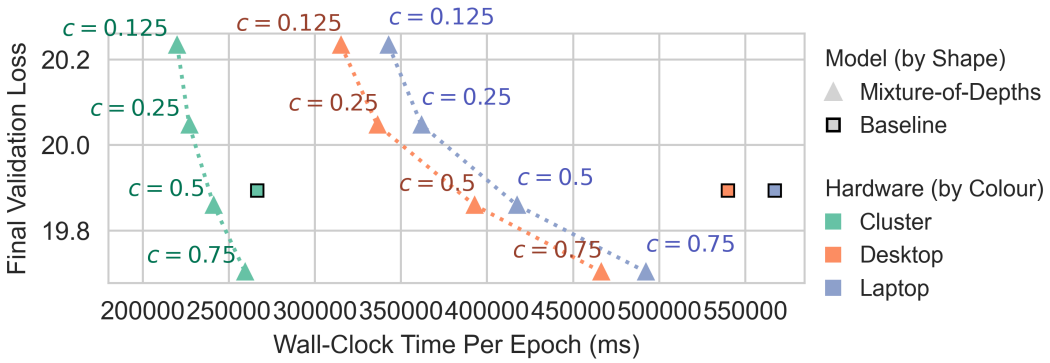

Figure 4.4: **Visualisation of Wall-Clock Training reductions achieved across different Mixture-of-Depths capacity factors.**

Looking at Table 4.4, however, it is discovered that the relative runtime reductions achieved by the Mixture-of-Depths model over the Baseline also follows a similar pattern to the absolute runtime reduction. Models under the Desktop environment shows reductions matching most closely to the theoretical reductions detailed in Section 4.3.1. The reductions under the Laptop setting follows, while the reduction of Mixture-of-Depths models from the Baseline on the Cluster environment is less than half of those achieved on the Laptop across all capacity factors, with a very small reduction by $c = 0.75$.

Model	Cluster (%)	Desktop (%)	Laptop (%)
Baseline	0%	0%	0%
Mixture-of-Depths $c = 0.125$	17.47%	41.65%	39.57%
Mixture-of-Depths $c = 0.25$	14.82%	37.72%	36.18%
Mixture-of-Depths $c = 0.5$	9.49%	27.27%	26.41%
Mixture-of-Depths $c = 0.75$	2.65%	13.63%	13.21%

Table 4.4: **Training reductions (%) achieved by Mixture-of-Depths models over the Baseline model, across Cluster, Desktop, and Laptop environments.**

4.4.3 Inference Wall-Clock Time

Comparing across Table 4.4 and Table 4.5, which details the reductions achieved during training and inference respectively, it can be observed that overall less efficiency is gained during inference by employing Mixture-of-Depths models, compared to during training. From Table 4.5, all models under the Desktop and Laptop setting show competitive inference efficiency over the Baseline, while the efficiency is more subdued on the Cluster setup. The Mixture-of-Depths model with $c = 0.75$ actually has a longer inference time compared to the Baseline. Looking at Figure 4.5, it is confirmed that it shows the same trend as observed in Figure 4.4.

Model	Cluster (%)	Desktop (%)	Laptop (%)
Baseline	0%	0%	0%
Mixture-of-Depths $c = 0.125$	8.13%	41.30%	36.43%
Mixture-of-Depths $c = 0.25$	7.07%	36.55%	32.69%
Mixture-of-Depths $c = 0.5$	3.72%	25.14%	22.59%
Mixture-of-Depths $c = 0.75$	-2.31%	11.92%	9.22%

Table 4.5: **Inference reductions (%) achieved by Mixture-of-Depths models over the Baseline model, across Cluster, Desktop, and Laptop environments.**

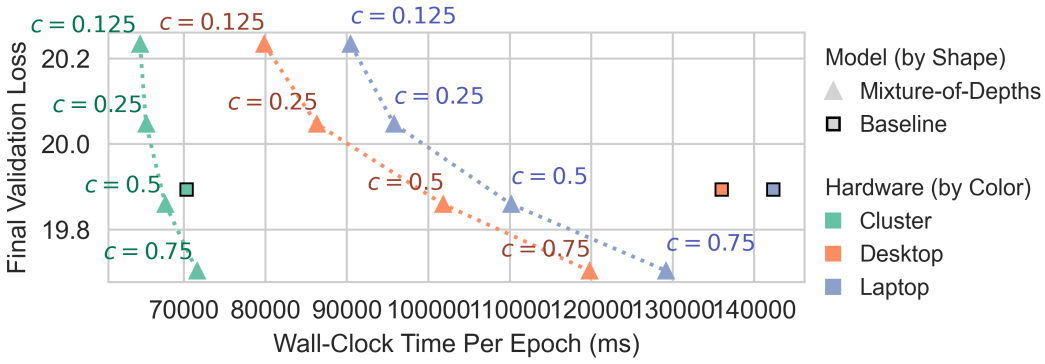

Figure 4.5: **Visualisation of Wall-Clock Inference reductions achieved across different Mixture-of-Depths capacity factors.**

4.4.4 Investigating Discrepancy Across Environments

From the measurements of practical efficiency across different hardware environments, large discrepancies in the percentage reductions achieved between Cluster, Desktop, and Laptop are observed during both training and inference. This is clearly shown in Tables 4.4 and 4.5. Furthermore, the inference time of the Mixture-of-Depths model with capacity factor $c = 0.75$ in fact has longer runtime compared to the Baseline.

I hypothesise that this discrepancy is due to a weak CPU on the Cluster environment compared to others, and I profile the execution times on both CPU and GPU within the Cluster and Laptop environments. The profiled runtimes is used as a broad guide to reveal trends rather than the ground truth, as (1) profiling introduces (unknown) additional overhead costs which add to the actual runtime (2) profiled runtimes assumes sequential execution, whereas in practice PyTorch executes instructions in parallel.

		Forward		Backpropagation	
	Model	CPU Time	GPU Time	CPU Time	GPU Time
Cluster	Baseline	5.447s	3.679s	10.572s	195.463µs
	$c = 0.125$	8.126s	2.215s	11.958s	194.696µs
Laptop	Baseline	10.660s	13.687s	29.540s	31.382s
	$c = 0.125$	10.555s	12.449s	28.041s	28.263s

Table 4.6: **CPU and GPU runtimes across Cluster and Laptop setups for Baseline and Mixture-of-Depths model with $c = 0.125$.** Values are rounded to 3 decimal places.

In Table 4.6, it is observed that runtime is uniformly reduced across all operations for the Laptop environment from the Baseline to the Mixture-of-Depths model. However, on the Cluster environment, runtime is decreased on the GPU but increased on the CPU. Both the Forward and Backpropagation operations are bottlenecked by CPU runtime on the Cluster, significantly so during Backpropagation. Considering the hardware specifications from 4.4.1, the weak CPU is the main cause for the subdued efficiency of Mixture-of-Depths models on the Cluster, and it is important to note that the CPU hardware is important to obtain maximum efficiency on Mixture-of-Depths models.

4.5 Summary

In this chapter, the efficiency gains from applying Mixture-of-Depths models to a dataset of speech utterances with varying lengths is thoroughly investigated, and the model efficiency is robustly evaluated through both theoretical and practical perspectives. There is significant theoretical runtime reductions to be gained from applying a Mixture-of-Depths model. The same holds in practice across all but one hardware environment evaluated. I additionally uncover that achieving the theoretical efficiency in practice depends on the hardware of the training environment, specifically the CPU hardware.

Next chapter I turn my focus to the performance of Mixture-of-Depths models on downstream tasks, and investigate the other side of the efficiency-performance trade-off.

Chapter 5

Downstream Tasks: Evaluations of Performance

As mentioned at the end of Chapter 4, since now both the theoretical and practical efficiency are established under different capacity factor settings in Mixture-of-Depths, I shift my focus to the performance end of the efficiency-performance trade-off to be investigated.

Performance is investigated through two tasks: the extraction of phonetic information through phone classification, and the preservation of speaker information through speaker identification. I choose one task each from a select two categories of the commonly employed SUPERB benchmark (Yang et al., 2021) to evaluate two fundamental aspects of speech: content and speaker. As different acoustic and linguistic information are typically found to be concentrated at different layers in a self-supervised pre-trained model (Pasad et al., 2021; Chiu et al., 2025), I conduct a layer-wise analysis in all of the downstream experiments, evaluating the usefulness of hidden representations extracted at each layers as latent features input to a downstream model.

5.1 Phone Classification: Frame-Level Evaluation

5.1.1 Experiment Setup

Dataset I use the Wall Street Journal (WSJ) dataset (Paul and Baker, 1992) for phone classification, another common corpus used in speech. The speech content of this dataset originates from articles published in the Wall Street Journal newspaper. I use the `si284` training set for both training and development, splitting `si284` in a 9:1 ratio for training and development respectively. The `dev93` test set is used to report test-time accuracy. I use the force alignments identical to those developed by Yang et al. (2022) as the phone labels for each frame to train my models on. For this classification task, there are 42 possible label classes for each phone, including a silence label.

Data processing Following a similar pre-processing approach as detailed in the pre-training experiment (Chapter 4), the input to the models are extracted 40-dimensional

log-mel spectrograms with a 25ms window and 10ms shift, downsampled to 20ms by concatenation of every 2 contiguous frames. However, this causes a mismatch in resolution, as the labels extracted by Yang et al. (2022) are aligned with a 10ms shift. To pacify this mismatch in temporal resolution, each frame of all hidden representations extracted from the pre-trained models are repeated in an interleaving fashion, i.e. $(x_1, x_2, \ldots, x_T)$ becomes $(x_1, x_1, x_2, x_2, \ldots, x_T, x_T)$, and if the original utterance extracted with a 10ms shift had an odd number of frames, the last frame is dropped, and the extracted latent feature becomes $(x_1, x_1, x_2, x_2, \ldots, x_T)$.

Models Again, report results on the "Best Set" of Mixture-of-Depths models and compare to the Baseline model, as described in Section 3.2. Specifically, in line with the setup designed for the SUPERB benchmark (Yang et al., 2021), I freeze each pre-trained model and train a separate linear classifier to extract hidden representations from each layer of the model as latent feature inputs to the linear classifier.

Hyperparameters All linear classifiers are trained for 10 epochs with a batch size of 8 using the Adam optimizer. A learning rate search was conducted, and linear classifiers are trained with the optimal learning rate $2.5e^{-4}$.

Evaluation Metrics All models are evaluated by phone error rate (PER), which is defined as the proportion of incorrect predictions made by the trained linear classifiers.

5.1.2 Phone Classification Results

Model	Best PER Across Layers (%)	Optimal Layer
Baseline	26.49%	8
Mixture-of-Depths $c = 0.125$	28.03%	8
Mixture-of-Depths $c = 0.25$	28.88%	7
Mixture-of-Depths $c = 0.5$	26.67%	8
Mixture-of-Depths $c = 0.75$	26.73%	8

Table 5.1: **Best PER (↓) and optimal layer for Baseline and Mixture-of-Depths models (No Activation, Offset 1).** Lower values represent better performance.

Best Performance Table 5.1 shows the best PER achieved by all models for the phone classification task, and the layer where they are achieved respectively. All Mixture-of-Depths models achieve very close PER to the Baseline: the difference between the Baseline and the best-performing Mixture-of-Depths model ($c = 0.5$) is only 0.18%, while the difference between the Baseline and the worst-performing Mixture-of-Depths model ($c = 0.25$) is 2.39%. All models reach optimal performance at similar layers: the Mixture-of-Depths model with $c = 0.125$ at Layer 7, while all other Mixture-of-Depths models and the Baseline at Layer 8. Broadly, Mixture-of-Depths models trained with a larger capacity factor achieve better performance, though it may not hold when the difference in capacity factor is small (for example, $c = 0.5$ and $c = 0.75$).

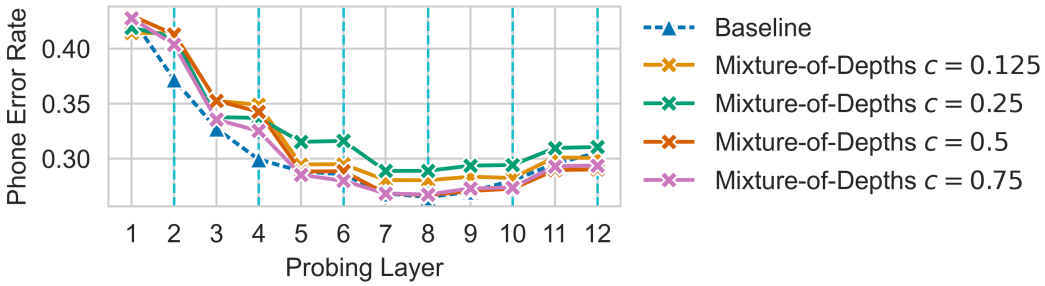

Figure 5.1: **Layer-wise PER (↓) reported for Baseline and Mixture-of-Depths models (No Activation, Offset 1).** Mixture-of-Depths layers are indicated through vertical dashed blue lines in the figure, specifically at layers 2, 4, 6, 8, 10, and 12. Lower values represent better performance.

Layer-wise Trends Figure 5.1 depicts results from probing each layer of all pre-trained models. Overall, Mixture-of-Depths models all follow a similar trend to the Baseline model. Performance is strongest at around Layer 8, and deteriorates towards earlier and later layers. Before the optimal layer (layer 8), it can be observed that Mixture-of-Depths models have worse performance at Mixture-of-Depths layer, likely because only a certain proportion of frames were processed at these layers. Performance quickly picks up again at regular transformer encoder layers following these Mixture-of-Depths layers until reaching matching or almost matching best performances to the Baseline. This indicates that the router at these layers learn effective specialised transformations for selected frames at each layer. However, after the optimal layer 8, it can be observed that Mixture-of-Depths models maintain optimal performance better than the Baseline model. This is especially evident when looking at the models with the best performance out of the Mixture-of-Depths set, $c = 0.5$ and $c = 0.75$. At the Mixture-of-Depths layers (layers 10 and 12) after the optimal layer 8, they both show better performance compared to the baseline, suggesting that the router in these Mixture-of-Depths layers also learns which frames should be processed.

Efficiency-Performance Trade-offs Considering the efficiency analysed in Chapter 4, the Mixture-of-Depths model with $c = 0.5$ would the best option if the aim is to maximise the preservation of performance while achieving efficiency gains. On the other hand, if efficiency were prioritised, the Mixture-of-Depths model with $c = 0.125$ is extremely effective in its efficiency-to-performance trade-off achieved.

5.2 Speaker Identification: Utterance-Level Evaluation

5.2.1 Experiment Setup

Dataset For speaker identification, I adopt the VoxCeleb1 dataset (Nagrani et al., 2020) which is widely used for speaker identification. It consists of speech utterances from 1,251 celebrity speakers, extracted from YouTube videos. I use the `dev` set for training and development, extracting 138,361 utterances for training and 8,251 for

development. It is ensured that there are utterances from each of the 1,251 speakers in each of the training and development splits. I use the `test` set to report my results.

Data processing The same pre-processing approach as detailed in the phone classification experiment is applied (Section 5.1.1). 40-dimensional log-mel spectrograms with a 25ms window and 10ms shift were extracted and downsampled to 20ms. In this case, as a single speaker embedding is desired representing multiple frames across the entire utterance, the extracted latent feature from the model is first averaged, then passed to a trainable linear classifier to obtain speaker information relevant to the task.

Models Results on the "Best Set" of Mixture-of-Depths models are reported and compared to the Baseline model, exactly as described in Section 5.1.1. Again, I freeze each pre-trained model and train a separate linear classifier to obtain the necessary speaker information from the latent features extracted from each hidden layer of the pre-trained model.

Hyperparameters All linear classifiers are trained for 10 epochs with a batch size of 8 using the Adam optimizer. A learning rate search was performed, and results are reported with optimal learning rate $1e^{-3}$.

Evaluation Metrics All models are evaluated by classification accuracy, which is the proportion of speakers the linear classifier at each layer correctly identifies.

5.2.2 Speaker Identification Results

Model	Best Acc Across Layers (%)	Optimal Layer
Baseline	40.81%	7
Mixture-of-Depths $c = 0.125$	38.21%	7
Mixture-of-Depths $c = 0.25$	38.15%	7
Mixture-of-Depths $c = 0.5$	38.37%	5
Mixture-of-Depths $c = 0.75$	39.15%	9

Table 5.2: **Best classification accuracy (↑) and corresponding optimal layer for Baseline and Mixture-of-Depths models (No Activation, Offset 1).** Higher values respresent better performance.

Best Performance Table 5.2 displays the best classification accuracy achieved by all models for the speaker identification task, and the layer where they are achieved respectively. Similar to what was observed in the phone classification experiment, all models achieve very similar best performance as the Baseline: the smallest difference in best performance was 1.66% ($c = 0.75$), while the largest difference was 2.66% ($c = 0.25$). The Mixture-of-Depths models with capacity factors $c = 0.125$ and $c = 0.25$ reach optimal performance at the same layer as the Baseline (Layer 7), while the model with $c = 0.5$ reaches optimal performance at an earlier layer (Layer 5) and the model

with $c = 0.75$ at a later layer (Layer 9). Again, overall a larger capacity factor achieves better performance, though the difference between models with $c = 0.125$ and $c = 0.25$ is an exception.

Layer-wise Trends Figure 5.2 shows the test accuracy of linear classifiers trained at each layer. There is no largely uniform trend for the behaviour of Mixture-of-Depths layers across all models, but models with $c = 0.125$ and $c = 0.25$ appear follow a similar trend and exhibit similar behaviour at Mixture-of-Depths layers as from phone classification experiment. Before reaching their respective optimal layer for performance, the Mixture-of-Depths layers incur a smaller increase from its previous layer in comparison to regular transformer encoder layers. However, after their respective optimal layers, Mixture-of-Depths layers become better at preserving performance from the previous layer compared to regular transformer encoder layers.

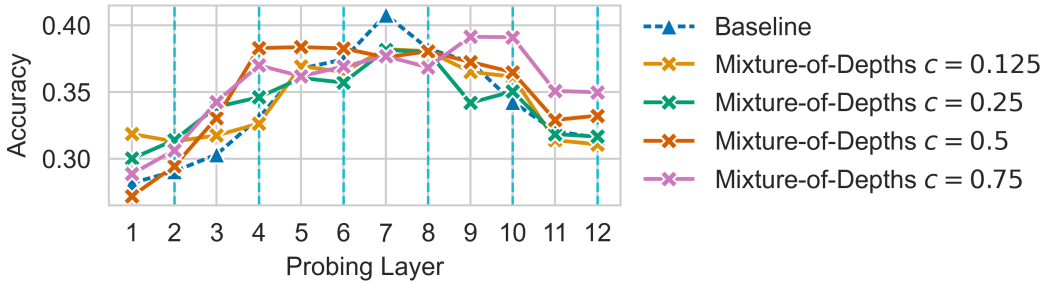

Figure 5.2: **Layer-wise classification accuracy (↑) reported for Baseline and Mixture-of-Depths models (No Activation, Offset 1).** Mixture-of-Depths layers are indicated through vertical dashed blue lines in the figure, specifically at layers 2, 4, 6, 8, 10, and 12. Higher values represent better performance.

Efficiency-Performance Trade-offs Considering the efficiency-performance trade-off for the speaker identification task, the model with capacity factor $c = 0.75$ is the best option if preservation of performance was the priority. However, efficiency gains for this setting can vary depending on the exact hardware environment, as detailed in Section 4.4 of the pre-training experiment. If efficiency were instead prioritised, the Mixture-of-Depths model with $c = 0.125$ would still be a very effective choice, with only a 2.60% reduction in classification accuracy.

5.3 Summary

In this chapter, the performance of the pre-trained Mixture-of-Depths models were evaluated on downstream phone classification and speaker identification tasks, investigating the trade-offs of the efficiency Mixture-of-Depths achieves to their effectiveness in extracting content and speaker information from speech. All models showed very similar performance to the Baseline across all tasks, and generally larger capacity factors $c = 0.5$ and $c = 0.75$ facilitate slightly better performance compared to smaller capacity factors of $c = 0.125$ and $c = 0.25$.

Chapter 6

Ablation Studies: What Does Mixture-of-Depths Learn?

In this chapter, I extend upon my explorations and findings from experiments during Chapter 4 and Chapter 5, and conduct several ablation studies.

In Section 6.1, I investigate the parameter distribution of Mixture-of-Depths models compared to the Baseline, and whether Mixture-of-Depths can generalise to a different capacity factor at inference without additional training. In Section 6.2, I examine in detail the top-k routing decisions made at Mixture-of-Depths layers, revealing acoustic properties that causes frames to be prioritised or skipped. And finally, in Section 6.3, I present results obtained from the different Mixture-of-Depths model sets with different activation and offset configurations (defined in Section 3.2) on experiments from Chapter 5, and provide insight into the best settings.

6.1 Cross-Capacity Zero-Resource Inference

In this section, I explore the ability of each Mixture-of-Depths model to operate at and generalise to capacity factors that are different to what they're trained with, without any additional training with the new capacity factor assigned. Specifically, I investigate two questions: (i) Is it possible to recover a regular transformer encoder network with the same architecture as the Baseline, by setting the capacity factor to 1 at inference, or does Mixture-of-Depths learn fundamentally different parameter distributions? (ii) Could a Mixture-of-Depths model train on a lower capacity factor and be able to scale to a higher capacity factor at inference without further training, and acquire better performance? Could it train on a higher capacity factor, but generalise to lower capacity factors without significant impact to its performance?

I investigate these questions on the phone classification experiments setup proposed in Section 5.1. To test different inference-time capacity factors on a pre-trained Mixture-of-Depths model, the capacity factor for each top-k router is modified. Each pre-trained model is frozen during the experiment, and linear classifiers are trained on the latent features extracted from each hidden layer of the model.

6.1.1 Inference $c = 1.0$: Studying Parameter Distribution

I experiment with two different approaches to study whether the parameter distribution in Mixture-of-Depths models align with the Baseline model. For Approach 1, I remove all router components from the forward computation graph of each Mixture-of-Depths model, replacing each Mixture-of-Depths layer by the transformer encoder layer that lies within. Results from this approach are shown at the top of Figure 6.1. For Approach 2, only the top-k routing logic is removed. Router weights r^{ℓ} are still computed and multiplied to all inputs, as is the original Mixture-of-Depths approach. This is shown at the bottom of Figure 6.1.

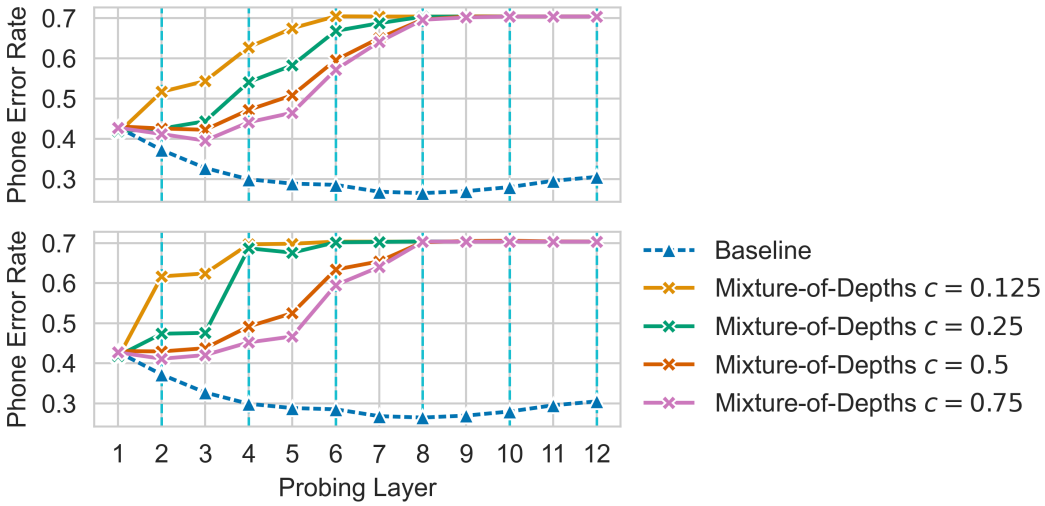

Figure 6.1: **PER (↓) of phone classification experiments after removing Mixture-of-Depths top-k routing from the computation graph (No Activation, Offset 1).** *Top: Approach 1.* Replaces each Mixture-of-Depths layer by the transformer encoder layer within. *Bottom: Approach 2.* Replaces Mixture-of-Depths layers with transformer encoder layers as outlined in Approach 1. In addition, the output from each Mixture-of-Depths layer is multiplied by their respective router weights.

From Figure 6.1, it is clear that all Mixture-of-Depths models have drastically different parameter distributions within their transformer encoder layers compared to the Baseline, regardless of Approach 1 or 2. It is not feasible to employ Mixture-of-Depths models with capacity factor $c = 1.0$, equivalent to just extracting a stack of the transformer encoder layers within, under any approach. Using the difference in Phone Error Rate between the Mixture-of-Depths models and the Baseline model from each of Approach 1 and Approach 2 as a proxy for deviation in parameter distributions, it appears that Approach 2 deviates further from the parameter distribution of a non-dynamically-trained model (Baseline). In addition, the lower the capacity factor, the more bottlenecked the Mixture-of-Depths model is at its Mixture-of-Depths layers and the further the parameter distributions deviate from the non-dynamically-trained model. However, despite this deviation from the Baseline parameter distribution, these Mixture-of-Depths models displayed in Figure 6.1 achieved very similar performance to the Baseline in the phone classification downstream experiment from Section 5.1. This indicates

that the Mixture-of-Depths models successfully learned specialised transformations required only for a subset of frames in each utterance in each Mixture-of-Depths layer, optimising efficiency without sacrificing performance.

6.1.2 Inference $c < 1.0$: Studying Generalisability

I now investigate the capability of Mixture-of-Depths models to generalise to different inference-time capacity factors less than 1, without additional training. Specifically, I choose to explore inference-time capacity factors $[0.125, 0.25, 0.5, 0.75]$, the same set of train-time capacity factor values I explored as described in Section 3.2. I leverage this to conduct a structured grid-like analysis.

For each model tested with a different inference-time capacity factor, the best Phone Error Rate across all layers probed is reported in Figure 6.2. I also include results from models with the same capacity factor at both training and inference as reference, displayed along the bottom-left to top-right diagonal of the heatmap.

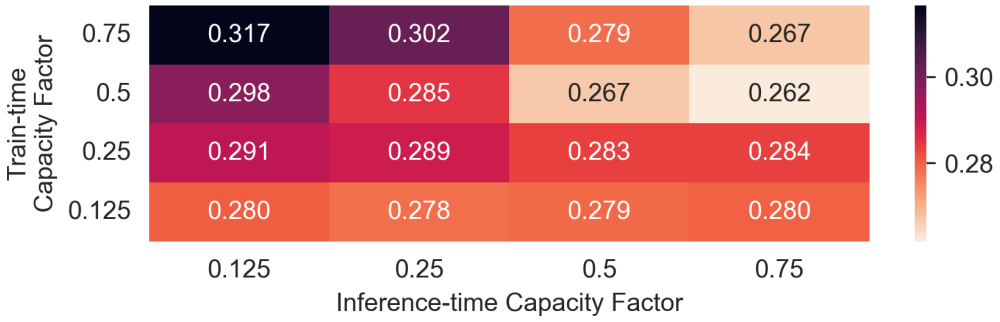

Figure 6.2: **Visualisation of frames routed through all 12 layers in Mixture-of-Depths model with $\mathbf{c = 0.75}$ (No Activation, Offset 1).** Lighter colour indicates lower Phone Error Rate, which corresponds to better performance.

A few notable patterns can be identified from Figure 6.2:

- All models are able to generalise across different inference-time capacity factors. There is no significant impact to performance by changing capacity factor at inference time.
- The higher the train-time capacity factor, the larger the performance gains achieved through increasing the inference-time capacity factor - although the improvements are small in absolute value. For example, increasing inference-time capacity factor does not provide much improvement in performance for the model trained on capacity factor $c = 0.125$, while for the model with $c = 0.5$, changing its inference-time capacity factor to $c = 0.75$ actually provides better results compared to the model trained on $c = 0.75$.
- The higher the train-time capacity factor, the more the performance deteriorates through decreasing the inference-time capacity factor. This is especially demonstrated through the model trained on capacity factor $c = 0.75$, providing a PER of

0.317 when configured with inference-time capacity factor $c = 0.125$, the highest PER out of all values in the heatmap.

- The model trained on capacity factor $c = 0.125$ (bottom row, corresponding to train-time capacity factor 0.125 on the y-axis of the heatmap) appears to learn a very specialised distribution. The quality of information extracted, reflected by the performance, is not affected as the inference-time capacity factor increases.

Additionally, it is worth noting that the best result from Figure 6.2 actually exceeds the Baseline result. This result is from the model trained on capacity factor $c = 0.5$, and configured to $c = 0.75$ at inference without further training. A comparison of the PER layer-wise between the model with this best configuration and the Baseline model is shown below in Figure 6.3.

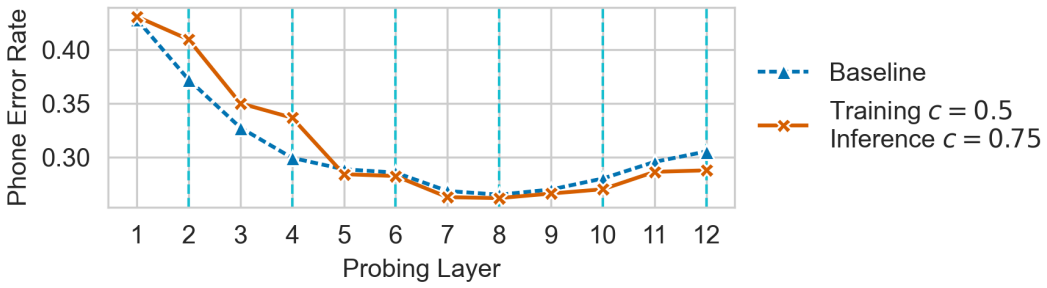

Figure 6.3: **Visualisation of frames routed through all 12 layers in Mixture-of-Depths model with c = 0.75 (No Activation, Offset 1).**

6.2 Priority of Phones: Visualising Routed Frames

Through experiments in Chapter 4 and Chapter 5, it is discovered that Mixture-of-Depths models are able to significantly improve efficiency while maintaining performance on downstream tasks evaluating for both content and speaker information in speech. The findings in Section 6.1.1 suggests that Mixture-of-Depths models learns specialised transformations required only for a subset of frames in each utterance. In this experiment, I aim to characterise properties of the subset of frames each Mixture-of-Depths layer chooses to process.

To investigate this, I use Mixture-of-Depths models from the Best Set, pre-trained on the `train-clean-100` subset of LibriSpeech for 200 epochs as detailed in the pre-training experiment in Chapter 4. Specifically, I examine two Mixture-of-Depths models, with capacity factors $c = 0.125$ and $c = 0.75$. In the analysis of the model with $c = 0.125$, I focus on which frames are prioritised by each Mixture-of-Depths layer, and I turn my attention to the frames which are skipped for the Mixture-of-Depths model with $c = 0.75$. I select one utterance from the Wall Street Journal dataset (Paul and Baker, 1992) and visualise the trajectory of this utterance through each Mixture-of-Depths model. The routing decisions at each layer are analysed in the context of the spectrogram and the force-alignment phonetic labels for the utterance. Each analysis

considers routing decisions made from two perspectives: a phone-wise analysis and a layer-by-layer analysis.

6.2.1 What is Prioritised the Most?

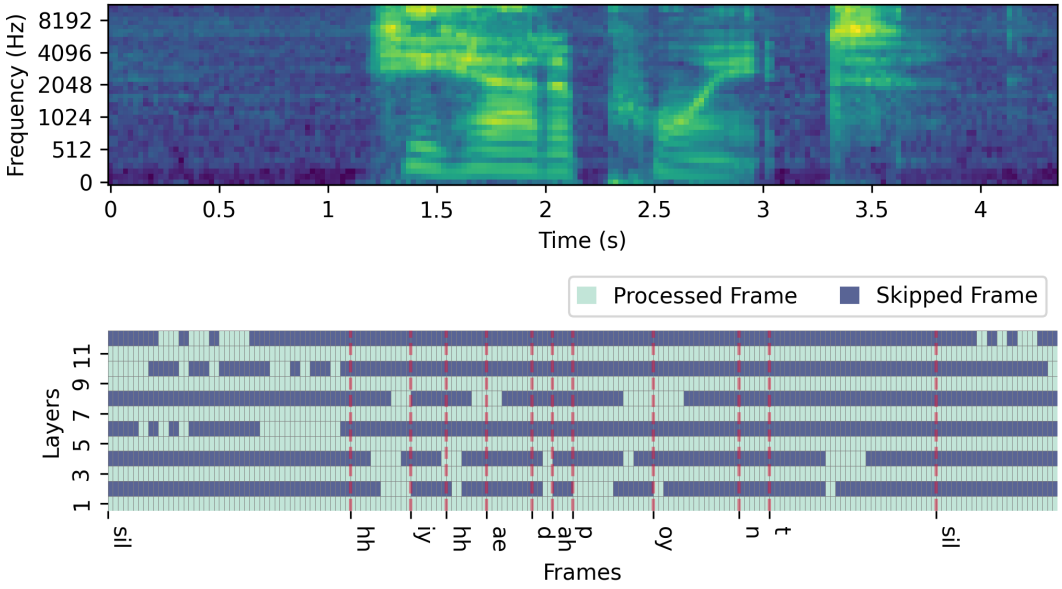

Figure 6.4: **Aligned spectrogram and heatmap, visualising frames routed through layers in the Mixture-of-Depths model with $\mathbf{c = 0.125}$ (No Activation, Offset 1).** *Top:* The log-mel spectrogram of the utterance processed. Darker colours represent regions of lower energy, and brighter colours represent higher energy. *Bottom:* Visualisation of selection of routed frames through all 12 layers of the model. The phonetic transcription of the utterance is denoted at the bottom: 'sil' 'hh' 'iy' 'hh' 'ae' 'd' 'ah' 'p' 'oy' 'n' 't' 'sil', corresponding to the text transcription "He had a point" with silence at the start and end.

Phone-wise analysis: each phone is processed where? Figure 6.4 shows the spectrogram features and routing decisions made for each frame of the utterance throughout the Mixture-of-Depths model with $c = 0.125$. Routing decisions are visualised at the bottom of the figure, and the regular transformer encoder layers in the model are labelled (Layers 1, 3, 5, 7, 9, 11). In this model, frames in the utterance corresponding to silence ('sil') are processed the most. They're processed at later layers, specifically from Layer 6 onwards. Frames corresponding to fricatives ('hh') and plosives ('d', 'p', 't') are also prioritised during Mixture-of-Depths layers, though their processing happens at earlier layers in the model. It appears that voiced vowels 'iy', 'ae', 'oy' are processed by some Mixture-of-Depths layer, but it can be observed that the specific frames processed lies at the boundaries between phones. For example, the boundary between 'iy' and 'hh' at Layer 4. This will be discussed in more detail in the layer-wise analysis next.

Layer-by-layer analysis: each layer processes which phones? Cross-referencing routing decisions with the corresponding spectrogram features in Figure 6.4, Layers 2

and 4 can be identified to process mostly fricatives and plosives, or transitions between these sounds and a voiced vowel. Observing the spectrogram, the processed frames generally correspond to features with low energies at lower frequencies that are not silence. Layer 6 identifies frames within the period of silence preceding speech. At Layer 8, it is evident that periods of transition from voiceless fricatives/plosives to voiced vowels are identified. Looking at the spectrogram, these periods all contain a change of energy in the lower frequencies. Layers 10 and 12 again identify silence, at both the start and the end of the utterance.

6.2.2 What is Skipped the Most?

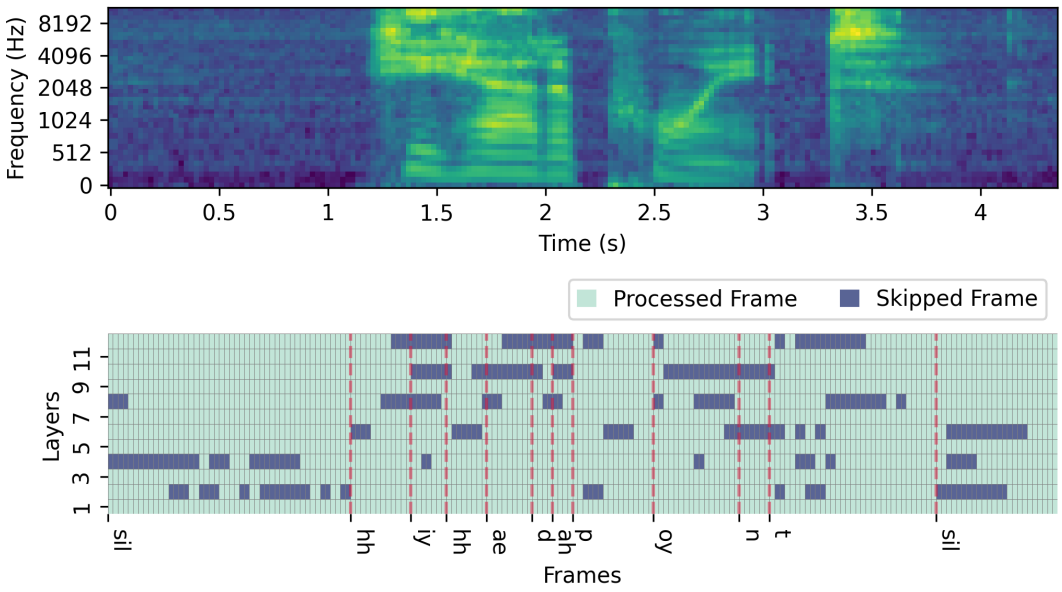

Figure 6.5: **Visualisation of frames routed through all 12 layers in Mixture-of-Depths model with $\mathbf{c = 0.75}$ (No Activation, Offset 1).** *Top:* The log-mel spectrogram of the utterance processed by the model, where darker colours represent regions of lower energy, and brighter colours represent higher energy. *Bottom:* Visualisation of selection of routed frames through all 12 layers of the model. The phonetic transcription of the utterance is denoted at the bottom: 'sil' 'hh' 'iy' 'hh' 'ae' 'd' 'ah' 'p' 'oy' 'n' 't' 'sil', corresponding to the text transcription "He had a point".

Phone-wise analysis: each phone is processed where? For the Mixture-of-Depths model with $c = 0.75$, the analytical focus shifts to the frames which are skipped. The decisions are visualised in Figure 6.5 above. A similar pattern that was previously observed emerges: periods of silence ('sil') are skipped in earlier layers and processed at later layers (Layer 6 onwards), while non-silence sounds generally show an opposite trend, prioritised in earlier layers and skipped at later layers. It can be observed that frames corresponding to the transition from silence to speech, and vice versa, are processed across all Mixture-of-Depths layers. In the spectrogram at the top of Figure 6.5, this corresponds to the region along the horizontal time axis between 1s and 1.5s where

the features transition from low energy across all frequencies (corresponding to silence 'sil') to high energy at high frequencies (voiceless fricative 'hh'), and the region between 1.5s and 4s where features transition from high energy at high frequencies (voiceless plosive 't') to low energy across all frequencies (silence 'sil').

Layer-by-layer analysis: each layer processes which phones? Consider the routing decisions layer-by-layer, Layer 2 identifies and skips frames corresponding to periods silence before, after, and during speech. When looking at the force-alignment labels, it appears that a few frames corresponding the plosives 'p' and 't' are processed, but looking at the corresponding features in the spectrogram reveals that these frames are periods with low energy across all frequencies where no sound is produced. Layer 4 recognises similar characteristics to Layer 2, and in addition skips regions within voiced vowels that are not near phonetic boundaries. At Layer 10, vowel sounds 'iy', 'ae', 'ah' and the nasal sound 'n' is neglected. No clear pattern emerges for Layers 6, 8, and 12, though silence frames are processed more.

6.2.3 Summary of Findings

Findings in this section suggest that voiceless sounds and silence - spectrogram features with relatively lower energy across the frequency spectrum, as well as transition boundaries between phones, are prioritised by Mixture-of-Depths models and identified as important to speech processing. Specifically, silence is skipped at earlier layers and prioritised towards later layers, while it is the reverse for fricatives and plosives.

These findings, considered in context with the results from Chapter 5, reveal more meaning. In the phone classification experiment in Section 5.1, the best Phone Error Rate achieved for both models analysed here is at Layer 8. Performance gradually tapers off afterward for the Mixture-of-Depths models, though not as sharply as the Baseline model. The prior observations can be connected here. Processing fricative and plosive sounds are key to the optimal performance at Layer 8, and allocating compute to silence frames and neglecting speech sounds slows the performance decrease. Looking at the speaker identification experiment from Section 5.2 instead, the Mixture-of-Depths models here with $c = 0.125$ and $c = 0.75$ reach optimal accuracy at Layer 8 and Layer 10 respectively. For both models, it appears the first layer allocating more processing compute to silence frames provides a critical boost to performance, but subsequent silence processing layers have a negative impact instead.

6.3 Exploring Configurations: Activations and Offsets

As part of my methodology from Chapter 3, I proposed different sets of Mixture-of-Depths models by varying router activation (sigmoid activation vs. no activation) and the Mixture-of-Depths layer offset (offset 0 vs. offset 1) in Section 3.2. I selected a "Best Set" based on effectiveness at smaller capacity factors, as the focus of this thesis was to improve efficiency, and analysed results from models in the "Best Set" during subsequent experiments in Chapter 4 and Chapter 5. Here, I discuss results for all different configurations on downstream tasks from Chapter 5. The phone classification

results and speaker identification results are displayed in Figure 6.6 and Figure 6.7 respectively.

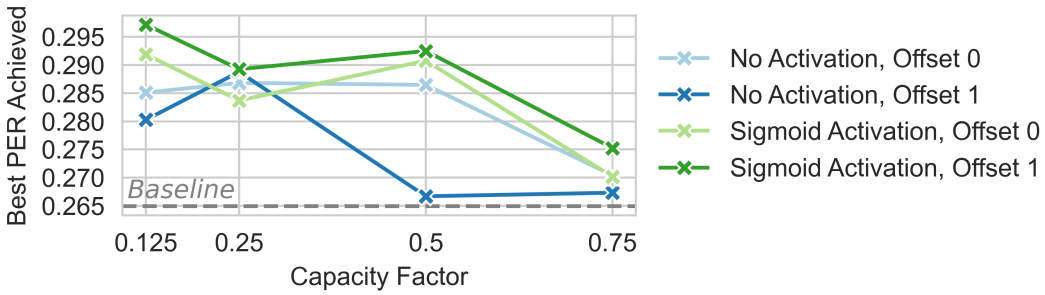

Figure 6.6: **Phone Classification Results: Best PER (↓) across different capacity factors, in all Mixture-of-Depths model sets.** (↓ indicates that better performance corresponds to a lower PER)

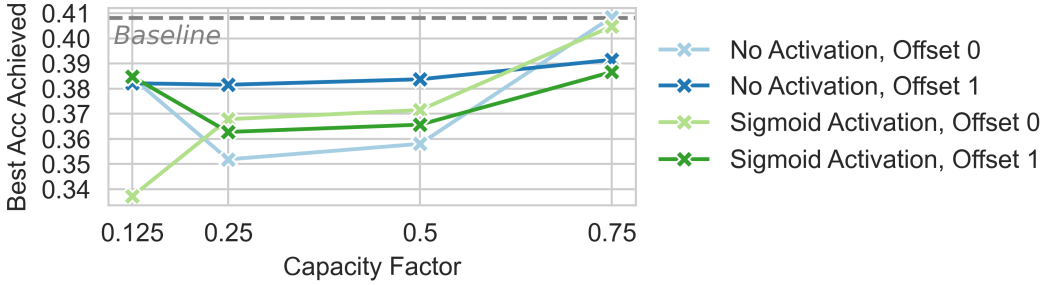

Figure 6.7: **Speaker Identification Results: Best Accuracy (↑) across different capacity factors, in all Mixture-of-Depths model sets.** (↑ indicates that better performance corresponds to a higher accuracy)

Compared to the configuration proposed by Raposo et al. (2024) (Sigmoid Activation, Offset 0), the configuration of "Best Set" (No Activation, Offset 1) is generally superior. In phone classification experiments, results from the original configuration are dominated by the "Best Set" at either the same or a smaller capacity factor. For example, even though at $c = 0.25$, the original configuration achieves lower PER than the "Best Set", both results are dominated by the model from the "Best Set" with $c = 0.125$. In speaker identification, the configuration in the main experiments dominates the original configuration in all capacity factors except for $c = 0.75$, where the least efficiency is achieved.

In general, considering the best trade-off in performance across both experiments, the "Best Set" I reported in the main experiments is the most effective when paired with smaller capacity factors 0.125, 0.25, and 0.5, that aims for better efficiency. However, while the "Best Set" is superior to other models at capacity factor 0.75 during the phone classification experiment, it falls short during the speaker identification experiment. Paired with capacity factor $c = 0.75$, the configuration with no router activation and offset 0 would have achieved the best trade-off between content and speaker information extracted at its optimal layer.

Chapter 7

Conclusions

7.1 Reviewing Results

In this thesis, I explored adapting the Mixture-of-Depths method proposed by Raposo et al. (2024) to improve both training and inference efficiency during the self-supervised pre-training of a speech model while maintaining effectiveness in facilitating effective extraction of content and speaker information within speech. Towards hypotheses I proposed in Section 3.4, the following results were found:

1. (H1) All Mixture-of-Depths models, across capacity factors 0.125, 0.25, 0.5, and 0.75, were able to achieve theoretical FLOPs reductions over the Baseline, with the $c = 0.125$ achieving a reduction of 43.66%. In practice, all Mixture-of-Depths models were able to achieve both training and inference wall-clock time reductions over the Baseline across all hardware environments, except for the Mixture-of-Depths model with capacity factor $c = 0.75$, where the model had a longer inference time compared to the Baseline. It was discovered that the practical efficiency gains achieved by Mixture-of-Depths depends heavily upon the CPU capabilities of the hardware environment.

2. (H2) When transferred to downstream phone classification and speaker identification tasks, all pre-trained Mixture-of-Depths models were able to achieve performance comparable to the Baseline, and successfully demonstrate their ability to preserve the extraction of useful content and speaker information from the raw utterance in their latent features while improving efficiency. In general, higher capacity factors facilitated improvements in performance on downstream tasks; however, the difference is still small, and the smallest capacity factor $c = 0.125$ achieves an effective efficiency-performance trade-off.

3. (H3) The parameter distributions learned by transformer encoder layers within Mixture-of-Depths models are all fundamentally different compared to the Baseline. Further investigation was conducted to find that at a smaller capacity factor of $c = 0.125$, Mixture-of-Depths layers learn to prioritise frames with relatively lower total energy across the frequency spectrum, such as silence, fricatives, plosives, as well as boundaries between fricatives/plosives and voiced vowels.

Generally, silence is processed more towards later layers, while part of the utterance that contains speech is processed during earlier layers.

4. (H4) Mixture-of-Depths models are found to adapt well if they were assigned a different capacity factor than the one the model is pre-trained with at inference time, with no additional training to the pre-trained model itself. Models trained on higher capacity factors achieve better gains in performance with a higher inference-time capacity factor. In particular, the model with train-time capacity factor 0.5 and inference-time capacity factor 0.75 exceeds Baseline performance.
5. Overall, the Mixture-of-Depths configuration that removes the sigmoid router activation and assigns Mixture-of-Depths layers to the second out of every two layers in the Mixture-of-Depths models was found to be most effective, especially in cases where the higher efficiency of a small capacity factor is desired.

7.2 Limitations

1. Despite best efforts, the wall-clock time measured on the Cluster hardware environment, as described in Chapter 4, still showed a large variance between epochs and runs. Runtime could be affected by concurrent jobs running on the same node, or the current thermal conditions of hardware on the compute node, which I could not control.
2. All models were pre-trained on the `train-clean-100` subset, which comprises 100 hours out of 960 hours of speech total in LibriSpeech, and only contains "clean" speech where the recording had higher quality and speakers had US English accents (Panayotov et al., 2015). Training on a larger dataset and with noisy utterances would facilitate more robust results.
3. Out of the 4 aspects of evaluation proposed for self-supervised pre-trained speech models in the standard benchmark SUPERB (Yang et al., 2021), only the content and speaker aspects were evaluated. The other two aspects, semantics and paralinguistics, are important to various other applications in speech.

7.3 Future Directions

While many experiments have been conducted in this thesis, there are still a few future directions that could be explored.

1. Investigate the compatibility of Mixture-of-Depths with SOTA self-supervised pre-training tasks that implement generating pseudo-targets from clustering techniques on top of masked prediction, such as wav2vec 2.0 (Baevski et al., 2020) or HuBERT (Hsu et al., 2021).
2. Develop optimisations that reduce the additional computation load router components in Mixture-of-Depths impose, relax its dependency on CPU hardware and achieve better global practical efficiency.

Bibliography

Harsh Ahlawat, Naveen Aggarwal, and Deepti Gupta. Automatic speech recognition: A survey of deep learning techniques and approaches. *International Journal of Cognitive Computing in Engineering*, 6:201–237, 2025. ISSN 2666-3074. doi: https://doi.org/10.1016/j.ijcce.2024.12.007. URL https://www.sciencedirect.com/science/article/pii/S2666307424000573.

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David C. Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. Colt5: Faster long-range transformers with conditional computation. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5085–5100. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.309. URL https://doi.org/10.18653/v1/2023.emnlp-main.309.

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-A

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *CoRR*, abs/1511.06297, 2015. URL http://arxiv.org/abs/1511.06297.

Yoshua Bengio. Deep learning of representations: Looking forward. In Adrian-Horia Dediu, Carlos Martín-Vide, Ruslan Mitkov, and Bianca Truthe, editors, *Statistical Language and Speech Processing - First International Conference, SLSP 2013, Tarragona, Spain, July 29-31, 2013. Proceedings*, volume 7978 of *Lecture Notes in Computer Science*, pages 1–37. Springer, 2013. doi: 10.1007/978-3-642-39593-2_1. URL https://doi.org/10.1007/978-3-642-39593-2_1.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70

of *Proceedings of Machine Learning Research*, pages 527–536. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/bolukbasi17a.html.

Heng-Jui Chang, Shu-Wen Yang, and Hung-yi Lee. Distilhubert: Speech representation learning by layer-wise distillation of hidden-unit bert. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*, pages 7087–7091. IEEE, 2022. doi: 10.1109/ICASSP43922.2022.9747490. URL https://doi.org/10.1109/ICASSP43922.2022.9747490.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE J. Sel. Top. Signal Process.*, 16(6):1505–1518, 2022. doi: 10.1109/JSTSP.2022.3188113. URL https://doi.org/10.1109/JSTSP.2022.3188113.

Aemon Yat Fei Chiu, Paco Kei Ching Fung, Roger Tsz Yeung Li, Jingyu Li, and Tan Lee. Probing speaker-specific features in speaker representations. *CoRR*, abs/2501.05310, 2025. doi: 10.48550/ARXIV.2501.05310. URL https://doi.org/10.48550/arXiv.2501.05310.

Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James Glass. An unsupervised autoregressive model for speech representation learning. In *Interspeech 2019*, pages 146–150, 2019. doi: 10.21437/Interspeech.2019-1473.

Seamless Communication, Loïc Barrault, Yu-An Chung, Mariano Coria Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan, John Hoffman, Christopher Klaiber, Pengwei Li, Daniel Licht, Jean Maillard, Alice Rakotoarison, Kaushik Ram Sadagopan, Guillaume Wenzek, Ethan Ye, Bapi Akula, Peng-Jen Chen, Naji El Hachem, Brian Ellis, Gabriel Mejia Gonzalez, Justin Haaheim, Prangthip Hansanti, Russ Howes, Bernie Huang, Min-Jae Hwang, Hirofumi Inaguma, Somya Jain, Elahe Kalbassi, Amanda Kallet, Ilia Kulikov, Janice Lam, Daniel Li, Xutai Ma, Ruslan Mavlyutov, Benjamin N. Peloquin, Mohamed Ramadan, Abinesh Ramakrishnan, Anna Y. Sun, Kevin Tran, Tuan Tran, Igor Tufanov, Vish Vogeti, Carleigh Wood, Yilin Yang, Bokai Yu, Pierre Andrews, Can Balioglu, Marta R. Costa-jussà, Onur Celebi, Maha Elbayad, Cynthia Gao, Francisco Guzmán, Justine Kao, Ann Lee, Alexandre Mourachko, Juan Pino, Sravya Popuri, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, Paden Tomasello, Changhan Wang, Jeff Wang, and Skyler Wang. Seamlessm4t-massively multilingual & multimodal machine translation. *CoRR*, abs/2308.11596, 2023. doi: 10.48550/ARXIV.2308.11596. URL https://doi.org/10.48550/arXiv.2308.11596.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN,*

USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. In Helen Meng, Bo Xu, and Thomas Fang Zheng, editors, *21st Annual Conference of the International Speech Communication Association, Interspeech 2020, Virtual Event, Shanghai, China, October 25-29, 2020*, pages 5036–5040. ISCA, 2020. doi: 10.21437/INTERSPEECH.2020-3015. URL https://doi.org/10.21437/Interspeech.2020-3015.

Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2022. doi: 10.1109/TPAMI.2021.3117837.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory F. Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *CoRR*, abs/1712.00409, 2017. URL http://arxiv.org/abs/1712.00409.

Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. doi: 10.1109/MSP.2012.2205597.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE ACM Trans. Audio Speech Lang. Process.*, 29:3451–3460, 2021. doi: 10.1109/TASLP.2021.3122291. URL https://doi.org/10.1109/TASLP.2021.3122291.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87, 1991. doi: 10.1162/NECO.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

Dongwei Jiang, Xiaoning Lei, Wubo Li, Ne Luo, Yuxuan Hu, Wei Zou, and Xiangang Li. Improving transformer-based speech recognition using unsupervised pre-training. *CoRR*, abs/1910.09932, 2019. URL http://arxiv.org/abs/1910.09932.

Heinrich Jiang, Been Kim, Melody Guan, and Maya Gupta. To trust or not to trust a classifier. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper$_f$*iles*/*paper*/2018/*file*/7180*cffd*6*a*8*e*829*dacfc*2*a*31*b*
Paper.pdf.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess,

Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3301–3310. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kaya19a.html.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1VaB4cex.

Qinyi Li. Efficient pre-training using fractalnet for adaptation to different resource availabilities in speech, April 2024.

Tzu-Quan Lin, Hung-Yi Lee, and Hao Tang. Melhubert: A simplified hubert on mel spectrograms. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2023, Taipei, Taiwan, December 16-20, 2023*, pages 1–8. IEEE, 2023. doi: 10.1109/ASRU57964.2023.10389700. URL https://doi.org/10.1109/ASRU57964.2023.10389700.

Tzu-Quan Lin, Hung yi Lee, and Hao Tang. Daisy: Data adaptive self-supervised early exit for speech representation models. In *Interspeech 2024*, pages 4513–4517, 2024a. doi: 10.21437/Interspeech.2024-626.

Xi Victoria Lin, Akshat Shrivastava, Liang Luo, Srinivasan Iyer, Mike Lewis, Gargi Ghosh, Luke Zettlemoyer, and Armen Aghajanyan. Moma: Efficient early-fusion pre-training with mixture of modality-aware experts. *CoRR*, abs/2407.21770, 2024b. doi: 10.48550/ARXIV.2407.21770. URL https://doi.org/10.48550/arXiv.2407.21770.

Oli Danyi Liu, Hao Tang, and Sharon Goldwater. Self-supervised predictive coding models encode speaker and phonetic information in orthogonal subspaces. In Naomi Harte, Julie Carson-Berndsen, and Gareth Jones, editors, *24th Annual Conference of the International Speech Communication Association, Interspeech 2023, Dublin, Ireland, August 20-24, 2023*, pages 2968–2972. ISCA, 2023. doi: 10.21437/INTERSPEECH.2023-871. URL https://doi.org/10.21437/Interspeech.2023-871.

Abdelrahman Mohamed, Hung-yi Lee, Lasse Borgholt, Jakob D. Havtorn, Joakim Edin, Christian Igel, Katrin Kirchhoff, Shang-Wen Li, Karen Livescu, Lars Maaløe, Tara N. Sainath, and Shinji Watanabe. Self-supervised speech representation learning: A review. *IEEE J. Sel. Top. Signal Process.*, 16(6):1179–1210, 2022. doi: 10.1109/JSTSP.2022.3207050. URL https://doi.org/10.1109/JSTSP.2022.3207050.

Mukhtar Mohamed, Oli Danyi Liu, Hao Tang, and Sharon Goldwater. Orthogonality and isotropy of speaker and phonetic information in self-supervised speech repre-

sentations. *CoRR*, abs/2406.09200, 2024. doi: 10.48550/ARXIV.2406.09200. URL https://doi.org/10.48550/arXiv.2406.09200.

Arsha Nagrani, Joon Son Chung, Weidi Xie, and Andrew Zisserman. Voxceleb: Large-scale speaker verification in the wild. *Computer Speech Language*, 60: 101027, 2020. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2019.101027. URL https://www.sciencedirect.com/science/article/pii/S0885230819302712.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015. doi: 10.1109/ICASSP.2015.7178964.

Ankita Pasad, Ju-Chieh Chou, and Karen Livescu. Layer-wise analysis of a self-supervised speech representation model. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2021, Cartagena, Colombia, December 13-17, 2021*, pages 914–921. IEEE, 2021. doi: 10.1109/ASRU51503.2021.9688093. URL https://doi.org/10.1109/ASRU51503.2021.9688093.

Douglas B. Paul and Janet M. Baker. The design for the Wall Street Journal-based CSR corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992. URL https://aclanthology.org/H92-1073.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 28492–28518. PMLR, 2023. URL https://proceedings.mlr.press/v202/radford23a.html.

David Raposo, Samuel Ritter, Blake A. Richards, Timothy P. Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *CoRR*, abs/2404.02258, 2024. doi: 10.48550/ARXIV.2404.02258. URL https://doi.org/10.48550/arXiv.2404.02258.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 843–852. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.97. URL https://doi.org/10.1109/ICCV.2017.97.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8,*

2016, pages 2464–2469. IEEE, 2016. doi: 10.1109/ICPR.2016.7900006. URL https://doi.org/10.1109/ICPR.2016.7900006.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-A

Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. Model compression and efficient inference for large language models: A survey. *CoRR*, abs/2402.09748, 2024. doi: 10.48550/ARXIV.2402.09748. URL https://doi.org/10.48550/arXiv.2402.09748.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Ratree Wayland. *Phonetics: A Practical Introduction*, chapter 7. Cambridge University Press, 2018.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. Berxit: Early exiting for BERT with better fine-tuning and extension to regression. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 91–104. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EACL-MAIN.8. URL https://doi.org/10.18653/v1/2021.eacl-main.8.

Gene-Ping Yang, Sung-Lin Yeh, Yu-An Chung, James R. Glass, and Hao Tang. Autoregressive predictive coding: A comprehensive study. *IEEE J. Sel. Top. Signal Process.*, 16(6):1380–1390, 2022. doi: 10.1109/JSTSP.2022.3203608. URL https://doi.org/10.1109/JSTSP.2022.3203608.

Shu-Wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y. Lin, Andy T. Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, Tzu-Hsien Huang, Wei-Cheng Tseng, Ko-tik Lee, Da-Rong Liu, Zili Huang, Shuyan Dong, Shang-Wen Li, Shinji Watanabe, Abdelrahman Mohamed, and Hung-yi Lee. SUPERB: speech processing universal performance benchmark. In Hynek Hermansky, Honza Cernocký, Lukás Burget, Lori Lamel, Odette Scharenborg, and Petr Motlícek, editors, *22nd Annual Conference of the International Speech Communication Association, Interspeech 2021, Brno, Czechia, August 30 - September 3, 2021*,

pages 1194–1198. ISCA, 2021. doi: 10.21437/INTERSPEECH.2021-1775. URL https://doi.org/10.21437/Interspeech.2021-1775.

Ji Won Yoon, Beom Jun Woo, and Nam Soo Kim. Hubert-ee: Early exiting hubert for efficient speech recognition. *CoRR*, abs/2204.06328, 2022. doi: 10.48550/ARXIV.2204.06328. URL https://doi.org/10.48550/arXiv.2204.06328.

Zhao You, Shulin Feng, Dan Su, and Dong Yu. Speechmoe: Scaling to large acoustic models with dynamic routing mixture of experts. In Hynek Hermansky, Honza Cernocký, Lukás Burget, Lori Lamel, Odette Scharenborg, and Petr Motlícek, editors, *22nd Annual Conference of the International Speech Communication Association, Interspeech 2021, Brno, Czechia, August 30 - September 3, 2021*, pages 2077–2081. ISCA, 2021. doi: 10.21437/INTERSPEECH.2021-478. URL https://doi.org/10.21437/Interspeech.2021-478.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper$_f$*iles*/*paper*/2020/*file*/*d*4*dd*111*a*4*fd*973394238*aca*5*c*
Paper.pdf.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. A survey on efficient inference for large language models. *CoRR*, abs/2404.14294, 2024. doi: 10.48550/ARXIV.2404.14294. URL https://doi.org/10.48550/arXiv.2404.14294.

Appendix A

Supplementing Results

A.1 Layer-Wise Phone Classification Results Across Different Configurations

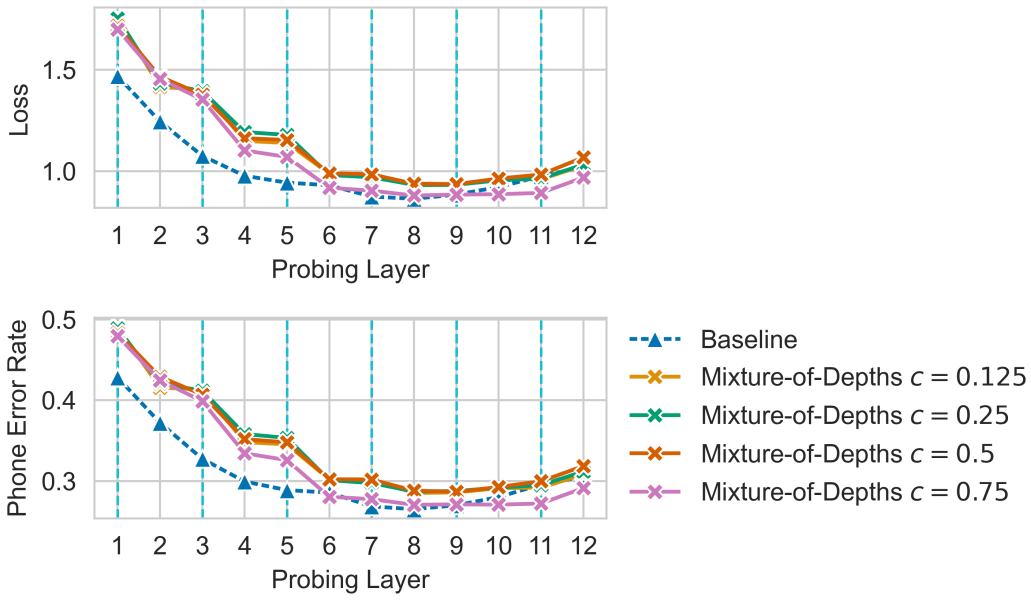

Figure A.1: **Loss and Accuracy of Mixture-of-Depths model with configuration: no router activation, offset 0.**

A.2 Layer-Wise Speaker Identification Results Across Different Configurations

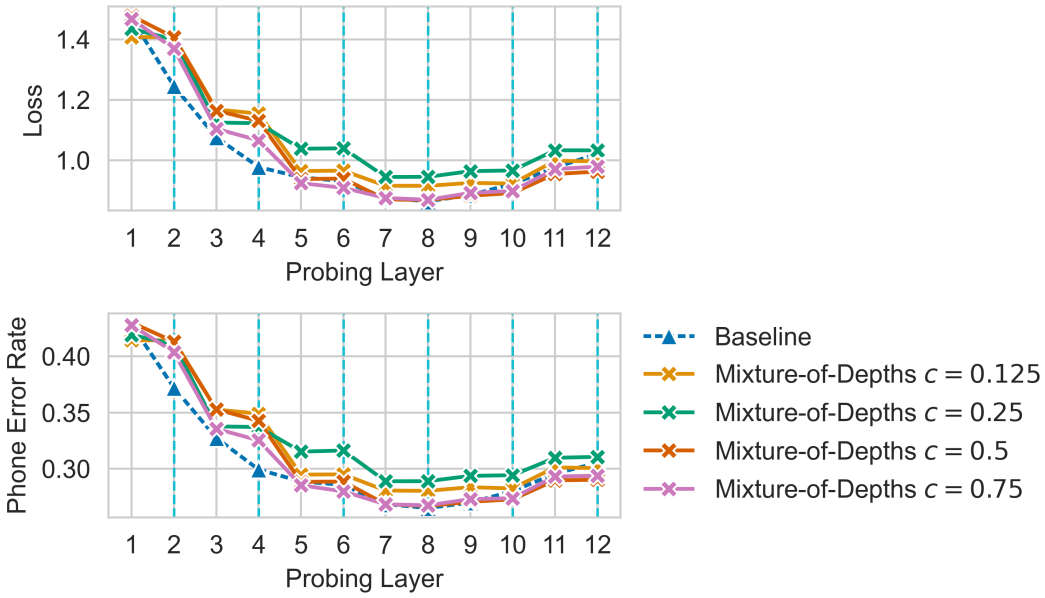

Figure A.2: **Loss and Accuracy of Mixture-of-Depths model with configuration: no router activation, offset 1.**

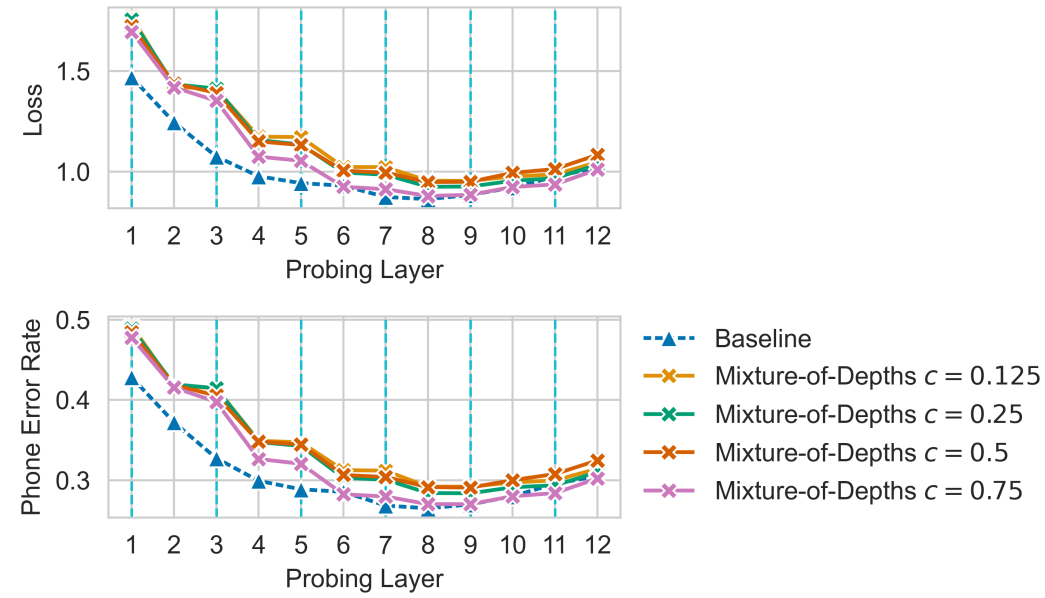

Figure A.3: **Loss and Accuracy of Mixture-of-Depths model with configuration: sigmoid router activation, offset 0.**

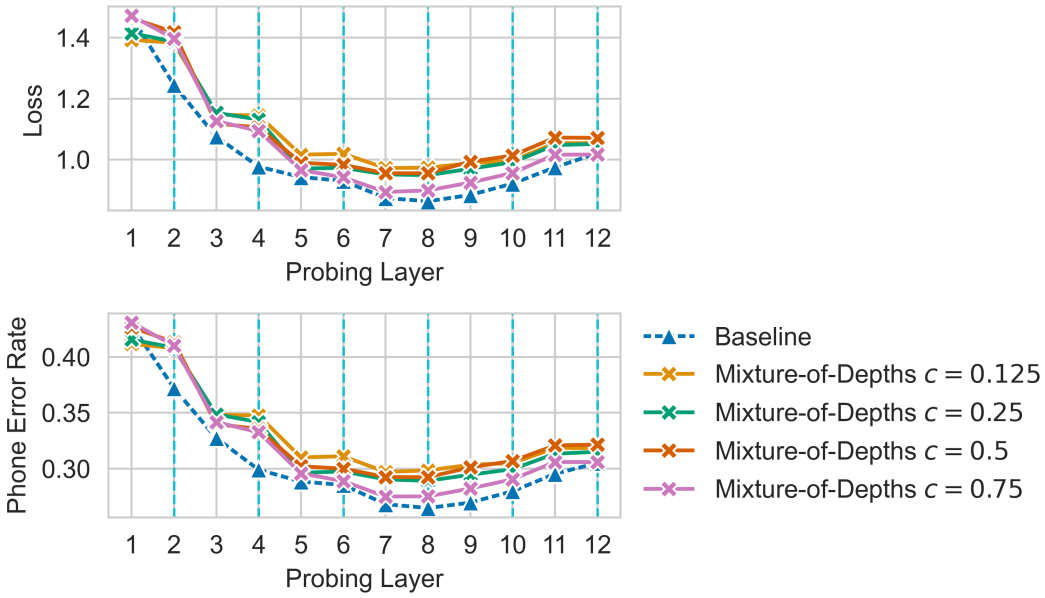

Figure A.4: **Loss and Accuracy of Mixture-of-Depths model with configuration: sigmoid router activation, offset 1.**

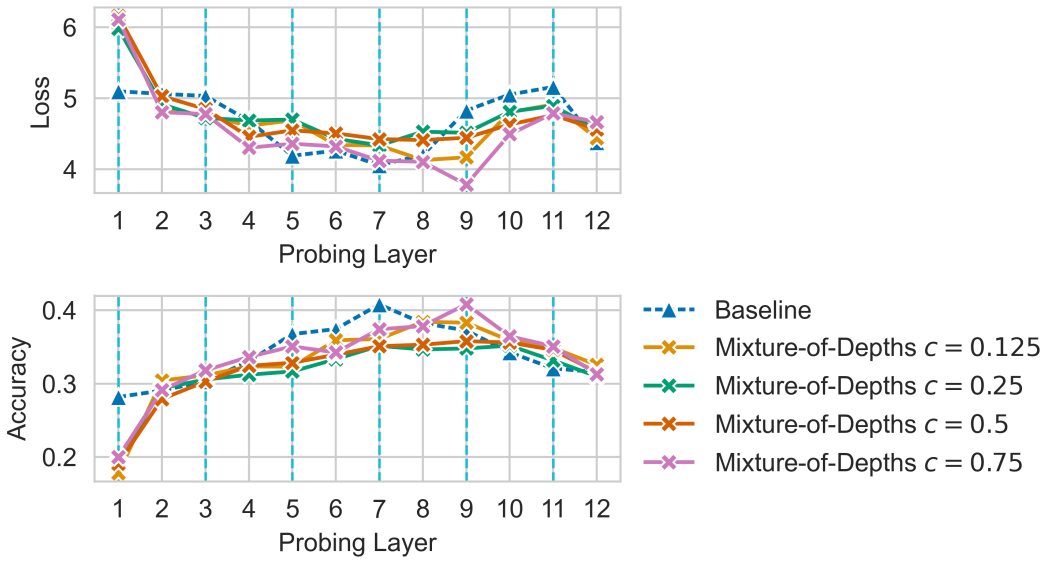

Figure A.5: **Loss and Accuracy of Mixture-of-Depths model with configuration: no router activation, offset 0.**

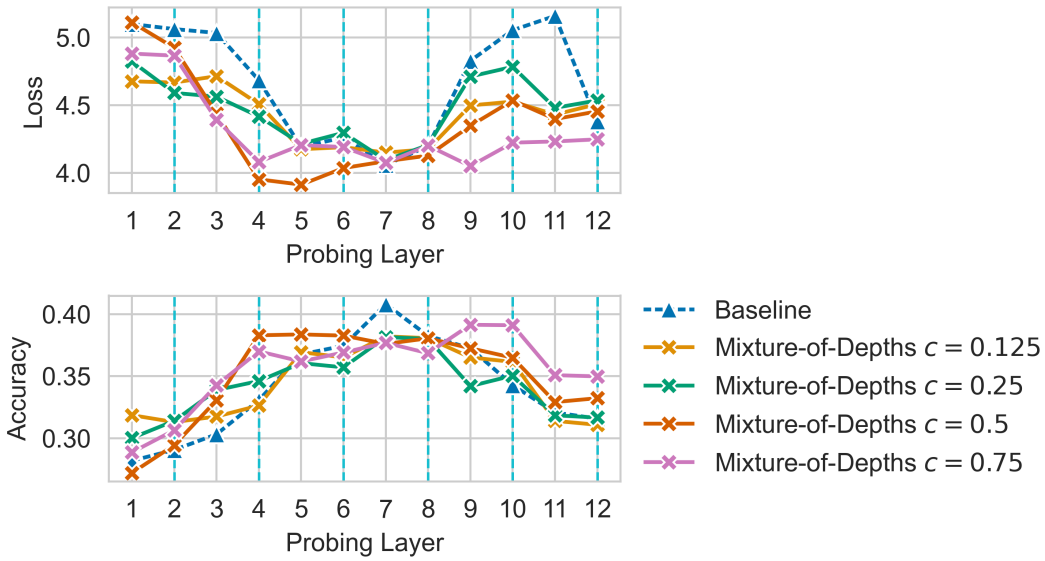

Figure A.6: **Loss and Accuracy of Mixture-of-Depths model with configuration: no router activation, offset 1.**

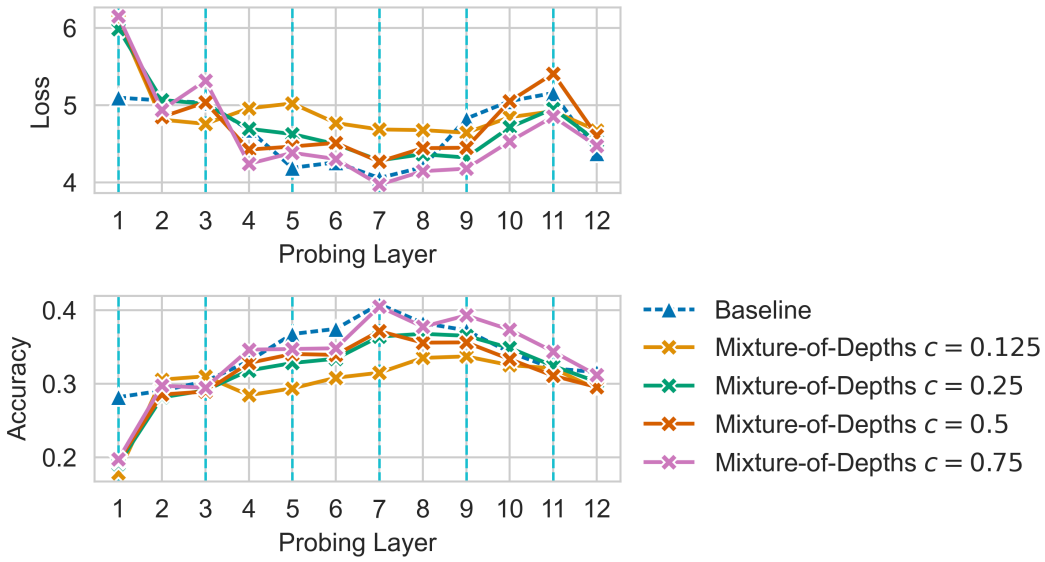

Figure A.7: **Loss and Accuracy of Mixture-of-Depths model with configuration: sigmoid router activation, offset 0.**

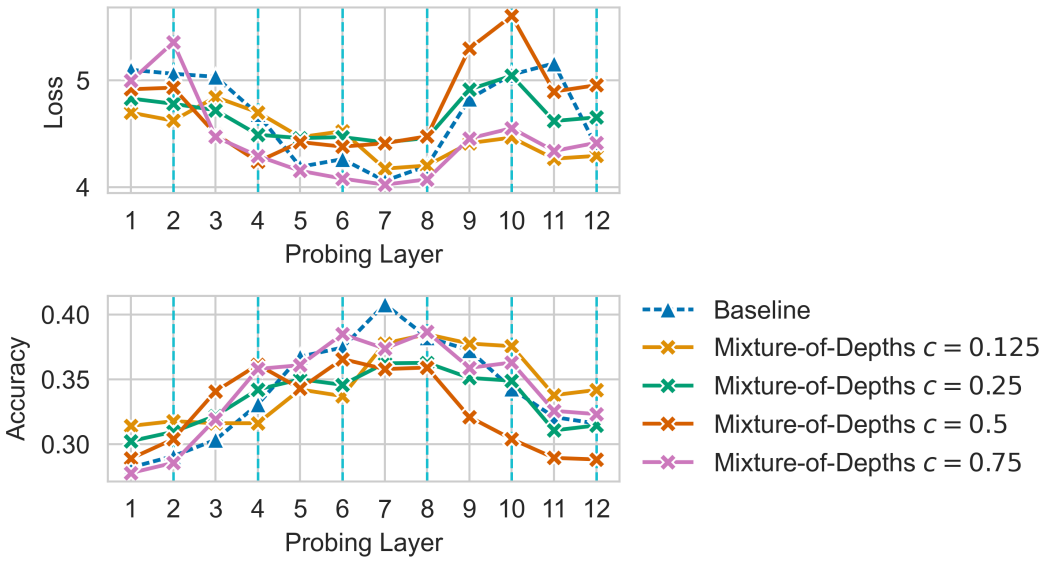

Figure A.8: **Loss and Accuracy of Mixture-of-Depths model with configuration: sigmoid router activation, offset 1.**