Dynamic Neural Architectures for Efficient Self-Supervised Speech Representation Learning

Josué Fleitas Sánchez

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2024

Abstract

This dissertation investigates the practical feasibility of adapting and applying dynamic neural architectures designed for deep residual CNNs to the self-supervised speech representation learning task. In particular, I explore Stochastic Depth and FractalNet macro-architectures on both Long Short-Term Memory and Transformer-based models as these commonly appear in state-of-the-art architectures for this task. The goal is then to determine whether the novel application of these techniques to the speech domain is able to reduce pre-training time requirements without incurring a large compromise in performance.

I conduct experiments using the Autoregressive Predictive Coding (APC) objective as a representative self-supervised loss. Furthermore, I assess the impact of the techniques by considering both their effect on the pre-training dynamics, the APC validation loss and their performance on two sample downstream tasks (Phone Classification and Speaker Verification). Experiments on downstream task performance are meant to evaluate the acoustic and phonetic information loss (if any) relative to their baselines.

I find that both approaches proposed provide a significant reduction in pre-training times and, upon careful hyperparameter tuning, are capable of remaining competitive relative to their baselines in every criterion. Furthermore, I empirically compare the two approaches and conclude that Fractal macro-architectures consistently outperform Stochastic Depth networks whereas the latter enable a greater reduction in training time. Finally, I make some smaller contributions assessing the impact of Fractal Block Width on time saving potentials and demonstrating the sensitivity of Stochastic Depth models to hyperparameters in this new domain.

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.

(*Josué Fleitas Sánchez*)

Acknowledgements

Firstly, I would like to thank my supervisor, Dr. Hao Tang, for all his invaluable guidance and feedback throughout this project. Without his input and suggestions, this dissertation would not have been possible.

Secondly, I would like to thank my dissertation project peers (Paul Martin, Qinyi Li and Jinghui Wang) for being so supportive throughout this project. The end result would also not have been possible without our mutual help and support.

Finally, I want to thank my family and friends for all their support and Grace for keeping me going through the hardest parts.

Table of Contents

Chapter 1

Introduction

Self-Supervised Learning has become a dominant framework for developing deep learning digital speech processing solutions. This is especially true for speech models of low-resource languages such as Urdu or Italian where supervised solutions cannot be applied without a significant investment in data collection and annotation. The success of these deep architectures have long phased out traditional Automatic Speech Recognition (ASR) approaches such as Hidden Markov Model (HMM)-based systems [Jelinek, 1976] and continue to excite with their potential to better human communication and improve human-computer interaction. However, while the success of the SSL paradigm has democratised digital speech technologies in recent years, it still suffers from the common failings of modern deep learning architectures, particularly in terms of hardware and time requirements as well as environmental impact.

1.1 Motivations

Due to the large data and training requirements of the SSL framework, many of the most popular speech representation models (wav2vec 2.0 [Baevski et al., 2020], HuBERT [Hsu et al., 2021], etc.) are restricted to being developed on extremely power-hungry state-of-the-art GPU clusters for several days at a time in order to achieve competitive results. On the other hand, pre-training also provides an opportunity for some leeway in accuracy provided the learned representations are still useful to the downstream tasks. This makes it a prime candidate for heuristic training methods which could reduce these unrealistic requirements.

1.2 Contributions

In this paper, I attempt to explore the potential of different training strategies and model architectures to reduce the time and hardware requirements on self-supervised speech representation learning. In particular, I study techniques that have found success and different degrees of adoption in the Computer Vision field and try to determine their practicality and effectiveness in the speech domain. My specific contributions are listed below.

- I evaluate the impact of Stochastic Depth [Huang et al., 2016] and FractalNet [Larsson et al., 2016] architectures on the Autoregressive Predictive Coding (APC) pre-training objective. (Chapters 5 and 6)
- By probing the pre-trained networks on a set of downstream tasks, I determine the phonetic and speaker-specific information lost through the use of these techniques (if any). (Chapter 7)
- I also explore the sensitivity of Stochastic Depth to the choice of hyperparameters as well as the effect of Fractal Block Width on pre-training in this new domain.

1.3 Structure

This dissertation is structured as follows. In Chapter 2, I present and explain some related works that this dissertation builds upon before discussing, in detail, the two techniques I will be focusing on in this project in Chapter 3. Chapter 4 covers the general methodology for the whole project along with the set of baselines, datasets and hypotheses used. Chapters 5 and 6 finally discuss the effect of the Stochastic Depth and FractalNets respectively on the APC objective. Chapter 7 then evaluates these models on the two downstream tasks being considered. Finally, Chapter 8 discusses conclusions, limitations and future avenues of research.

Chapter 2

Background

In this chapter, I will present the context required to understand the work carried out in this dissertation. I will begin by outlining some of the common strategies used to digitize speech inputs for Deep Learning tasks and discussing the foundational models being considered in this project. I will then present the self-supervised learning paradigm and the Autoregressive Predictive Coding objective I use throughout. In the final section, I also provide some criticism of related works by previous authors.

2.1 Digital Representations of Speech

There are two dominant digital representations of speech that are commonly used in the literature. The first, and perhaps the simplest, is the waveform representation, where we map changes in amplitude (corresponding to changes in air pressure) of the sound wave in the time domain. Because sound is continuous in nature, capturing it in a computer involves discretizing it by sampling the amplitude of the signal at regular intervals. The number of samples taken each second is typically referred to as the sampling rate of the audio and is often 16 kHz (16000 samples per second) for speech data. Many popular self-supervised networks use waveforms as their input, including Meta's wave2vec 2.0 [Baevski et al., 2020].

Because of the remarkable sensitivity of our auditory system in detecting differences in frequency, it often makes sense to transform waveforms into a frequency domain for ASR and other speech tasks. This can be accomplished by applying the Discrete Fourier Transform on a sliding time window with a given hop or stride. The extracted frequency slices can then be plotted against time to get a spectrogram. One final transformation that is typically performed is to convert the frequencies into (log) Mel-scale [Stevens et al., 1937]. This scale is preferred as human perception is more sensitive to lower frequencies. Other papers in the literature use these spectrograms as input, including the APC paper this work is based on and several CNN-based approaches to speech representation learning [Yang et al., 2022, Kriman et al., 2019].

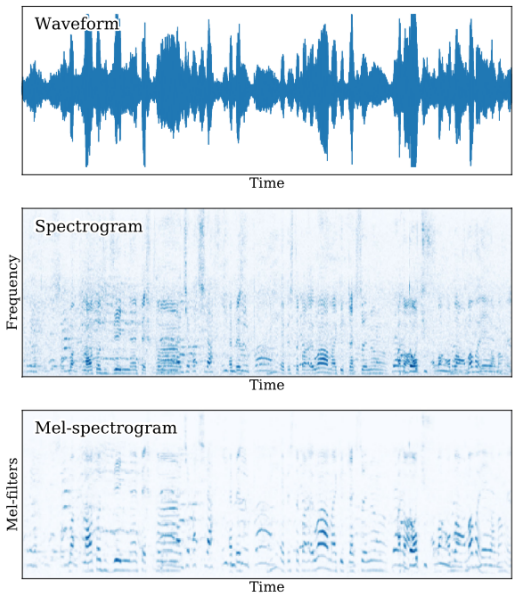


Figure 2.1: An illustration of the three most common representations of digitized speech. Diagram sourced from De Benito-Gorron et al. [2019]

2.2 LSTMs and Transformer Encoders

In this project, I consider experiments on both LSTMs and Transformer Encoders. These models feature extensively in the literature and many SOTA speech representation learning models.

2.2.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network that attempts to mitigate the vanishing gradient problem encountered by traditional recurrent architectures [Hochreiter and Schmidhuber, 1997]. In order to accomplish this, they extend the regular RNN unit with a Memory Block that contains gates to control the flow of information through the network. Every LSTM Memory Block features a cell state which persists information from earlier inputs as well as three separate gates: a forget gate, an output gate and an input gate. The general flow of information through each block is then as follows:

1. The ”forget gate” first decides what information to delete from the current cell state. This gate consists of a linear layer with a sigmoid activation that decides whether to keep or drop each number in the cell state C_{t-1}.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

2. The ”input gate” then decides what new information from the current data and previous hidden states is to be added to the cell state. These values then get transformed into a vector of potential cell state values $\tilde{C}_t$

$$i_t = \sigma(W_i[h_{t-1},\, x_t] + b_i), \qquad \tilde{C}_t = \tanh(W_c[h_{t-1},\, x_t] + b_C)$$

3. These last two values get combined in order to update the current cell state

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

4. Finally, the output of the current cell is determined. This proceeds in two steps. First, we use a linear layer to decide what parts of the cell state to output. Then, this is multiplied with a tanh activation of the current cell state.

$$o_t = \sigma(W_o[h_{t-1},\, x_t] + b_o), \qquad h_t = o_t * \tanh(C_t)$$

The memory mechanism at use here enables the networks to learn from distant inputs, making them useful to encode sequential information into a lower dimensional representation (or embedding). In particular, I consider networks that stack these LSTM Memory Blocks to form deep networks and extract the embedding from the final layer of this deep stack. These networks are trained using the Backpropagation Through Time (BPTT) algorithm proposed in Mozer [1989].

2.2.2 Transformer Encoders

Transformer Encoders are another popular network used to embed sequential information into a lower dimensional representation [Vaswani et al., 2017]. Unlike LSTMs and other recurrent architectures, Transformers process input data globally instead of incrementally, which makes them easier to parallelize. Their main objective is then to augment each element in a sequence with contextual information from surrounding elements using self-attention. In speech recognition, the networks aim to add acoustic context to each element such as surrounding phones and prosodic information. The Transformer Encoder architecture consists of a positional encoding layer, a multi-head self-attention unit and a feed-forward neural network to provide non-linearity.

Self-attention. A single self-attention operation decides which tokens to use as additional context for each input element. This is done by taking the input state vector z_i and performing a search query on a set of key and value pairs $(k_j,\, v_j)$. The queries, keys and values are all derived from the state vectors using learnable weights in the following manner.

$$\mathbf{q}_i = \mathbf{z}_i W_Q \qquad \mathbf{k}_j = \mathbf{z}_j W_K \qquad \mathbf{v}_j = \mathbf{z}_j W_V$$

For each key, we then compute an attention weight by taking the dot product of the ith query and the jth key.

$$\mathbf{w}_j = \mathbf{q}_i^\top \mathbf{k}_j$$

These attention weights are then normalised and turned into a probability distribution over all values using the softmax operator. This distribution then represents the likelihood of each value being context to the current state vector

$$\mathbf{c}_i = \sum_{j=1}^{N} \alpha_j \mathbf{v}_j \qquad \text{where} \quad \alpha_j = \frac{\exp(w_j)}{\sum_{l=1}^{N} \exp(w_l)}$$

In practice, transformer networks use multi-head attention, where multiple such self-attention layers are used in parallel and their weighted average is computed using learned weights as the final attention output. This is preferrable, as it allows each head to focus on different parts of the sequence simultaneously, which allows for more complex relationships and dependencies to be captured. In this work, I use 2 self-attention heads for simplicity but it is more common to see architectures with 8 or more attention heads.

Positional Encoding. Since transformers do not process information sequentially, temporal information in the context is completely lost. To circumvent this, positional information needs to be explicitly added to each of the input state vectors z_i before computing self-attention. In this work, I use the same sinusoidal function proposed by Vaswani et al..

$$PE_{(i,2j)} = \sin(i/10000^{2j/d_{model}}) \tag{2.1}$$

$$PE_{(i,2j+1)} = \cos(i/10000^{2j/d_{model}}) \tag{2.2}$$

where j represents the location in the position encoding vector for the ith state vector z_i. d_{model} is the dimension of the layer's output vector.

2.3 Self-Supervised Learning

Many of the recent successes in deep learning research have come from supervised learning where labeled data is used to guide the network's training. The effectiveness of these techniques, however, is limited to the availability of such labelled datasets, which is often a problem for many applications of industrial interest. As such, so-called unsupervised learning techniques have been developed which function by discovering useful patterns in raw, unlabelled training data [Mohamed et al., 2022]. These patterns can then be used to make predictions on further test data.

One subcategory of unsupervised learning that has enjoyed much success in speech and audio processing is that of self-supervised learning (SSL). This paradigm entails learning useful representations by extracting information from the training data itself and using this as labels. An example would be masking parts of the input data and asking the model to recover the masked data. The representation learned can then be used in several downstream tasks, eliminating the need to train an entirely new model for each of them. In relation to downstream applications, this means that SSL training is made up of two stages as indicated in Figure 2.2:

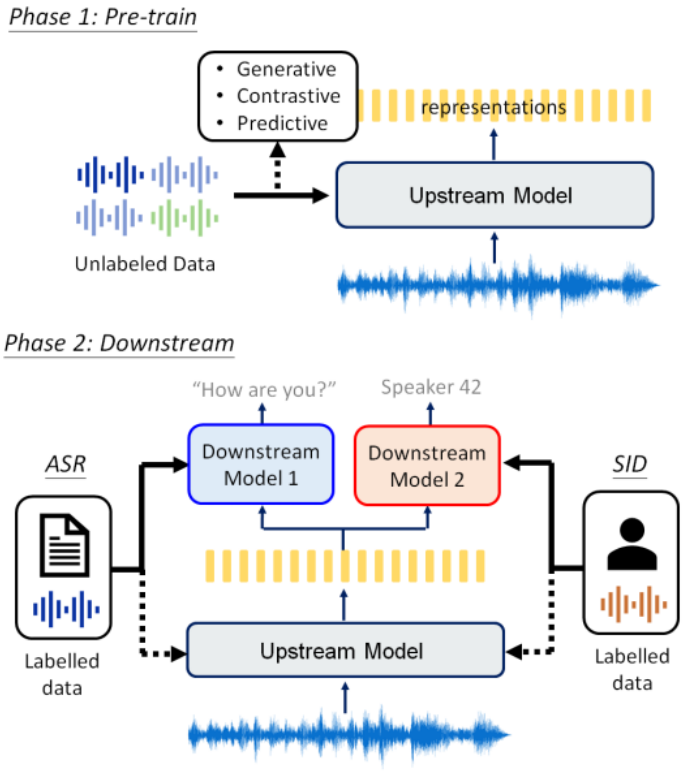


Figure 2.2: Self-Supervised Learning framework application on downstream tasks. Top is the pre-training phase where representation is learned. Bottom is the fine-tuning supervised stage on downstream task. Note that 'Upstream Model' is equivalent to 'Foundation Model'. Diagram sourced from Mohamed et al. [2022]

1. Use SSL to pre-train a 'foundation model' that learns a useful representation
2. Fine-tune the pre-trained model in a supervised fashion on the downstream task of interest

2.4 Autoregressive Predictive Coding (APC)

Autoregressive Predictive Coding (APC) is a simple objective function that can be used in self-supervised training to learn sequential information in unlabelled input data. In the context of speech representation learning, APC simply tries to predict speech frames k time steps into the future using a simple loss. In this project, I use a Mean-Squared Error (MSE) Loss in accordance with Yang et al.'s methodology but an l_1 norm is also possible. Formally,

$$\frac{1}{2}\sum_{t=1}^{T-k} \|x_{t+k} - Wh_t\|_2^2 \tag{2.3}$$

In the context of this project, the sequence $x_1, \ldots, x_T$ is a sequence of log Mel features and $h_t = f(x_{1:t})$ for an arbitrary neural network f. The weight matrix W represents a

linear layer's parameters that is tasked with predicting x_{t+k} using h_t. The goal of the objective is to obtain a neural network whose hidden representations h_t can be used as embeddings of the original speech utterance. The hyperparameter k controls the difficulty of the objective, with lower assignments being easier to learn than larger ones. An assignment of $k = 1$ would be equivalent to language modelling but Yang et al. explain this results in a task that is too simple to learn. This is largely due to the temporal smoothness in the spectrogram which is not a concern in regular language modeling. I use a setting of $k = 5$ in this project.

Other self-supervised losses exist that guide training in other non-autoregressive manners. For example, contrastive losses such as InfoNCE [Oord et al., 2018] aim to maximise the agreement between representations of positive samples (similar data instances), while minimising it between representations of negative samples (dissimilar data instances).

2.5 Criticism of Previous Work

This section motivates the position of this work in the general field by considering the limitations of other related work.

Self-Supervised Pre-Training Cost. Many of the SOTA models in speech processing applications leverage the recent advances in self-supervised learning methods in order to attain their impressive results. Due to the potential and relative infancy of these methods, most of the research in this area has been attempting to design novel training objectives without providing much discussion or analysis about the high computational costs of these networks. Attempting to reduce these costs could not only facilitate model deployment in commercial applications but could also help to democratize research in this area by reducing the barrier of entry which could in turn speed up development of these methods.

Dynamic Architectures in Speech SSL. A more recent paper by Lugo and Vielzeuf raised the issue of improving self-supervised speech representation model efficiency. However, many of their proposed methods of achieving this consist of transfer learning approaches (such as Prompt Tuning [Lester et al., 2021] and Bitfit [Ben Zaken et al., 2022]) or consider attempts at simplifying the self-attention computation by either approximating it (Reformers [Kitaev et al., 2020]) or optimizing its calculations (LSRA [Wu et al., 2020]). While they also discuss some alternative networks that could help reduce costs (such as LiGRUs [Ravanelli et al., 2018]) they fail to discuss or demonstrate the potential for the dynamic architectures proposed. To the best of my knowledge, no prior work has therefore attempted to adapt these techniques to the speech domain, despite these models having been developed 5+ years ago and enjoying moderate success in many Computer Vision tasks.

Chapter 3

Proposed Approach

In this chapter, I will present and explain the deep learning architectures and training strategies that I will be leveraging in this project.

3.1 Stochastic Depth

The Stochastic Depth [Huang et al., 2016] training procedure aims to achieve deep networks during testing but short ones during training on architectures with residuals/skip connections. Part of the appeal of this idea lies in its simplicity, since the main mechanism involves randomly dropping some layers during training in favour of the residual/identity connection according to some probability rule. In particular, the original paper exemplifies the potential of this framework on the ResNet architecture [He et al., 2016] demonstrating how it enables practical scaling of this model beyond 1000 layers. The authors also draw parallels between this technique and dropout, which reduces co-adaptation of hidden nodes in a network by randomly dropping some nodes in each layer [Hinton et al., 2012]. More specifically, it is explained that in both cases the techniques can be interpreted as training an ensemble of networks with smaller capacity, where stochastic depth considers networks of varying depths whereas dropout considers 'thinner' networks with less hidden units.

In the context of its application to the ResNet architecture, this technique can be formally stated by introducing a Bernoulli Random Variable b_l for a given block of the network and changing the layer's output to be

$$H_l = ReLU(b_l f_l(H_{l-1}) + id(H_{l-1}))$$

where $f_l()$ is the transformation performed by the block, $id()$ is the identity function and we assume a ReLU activation [Fukushima, 1969]. In this way, if the Bernoulli Variable is 1, we get the regular update rule and otherwise we skip all transformations for this block. The probability of the random variable being 1 is then termed its 'survival probability' and we end up with a set of survival probabilities p_l as new hyperparameters in training. The authors further suggest to use a linear decay function

$$p_l = 1 - \frac{1}{L}(1 - p_L)$$

to preserve more of the earlier layer transformations. This is motivated by the knowledge that earlier layers in CNNs learn information that will be used by later layers [Fong and Vedaldi, 2018].

3.2 Fractal Networks (FractalNets)

FractalNets are proposed as an alternative to deep residual architectures, where we use a recursive expansion rule to generate fractal macro-architectures with repeating substructures as shown in Figure 3.1 [Larsson et al., 2016]. This structure results in models with interacting subpaths of varying depths and eliminates the need to learn residuals explicitly. The different columns are then combined in the architecture via a join layer which applies a simple element-wise mean to produce a single output per Fractal Block (although other approaches are possible). The paper also introduces drop-path as an architecture-specific form of regularization similar to regular dropout where entire branches get dropped stochastically during training to reduce co-adaptation of parallel paths. Two drop-path strategies are proposed: global and local (see Figure 3.2). In global drop-path, we select a single column for the entire block and drop every other layer in the block. Local drop-path is similar but instead considers dropping each input to a join layer with a specific probability while ensuring that at least one connection is maintained. The paper uses a mixture of these two approaches as their drop-path strategy which I replicate in this dissertation.

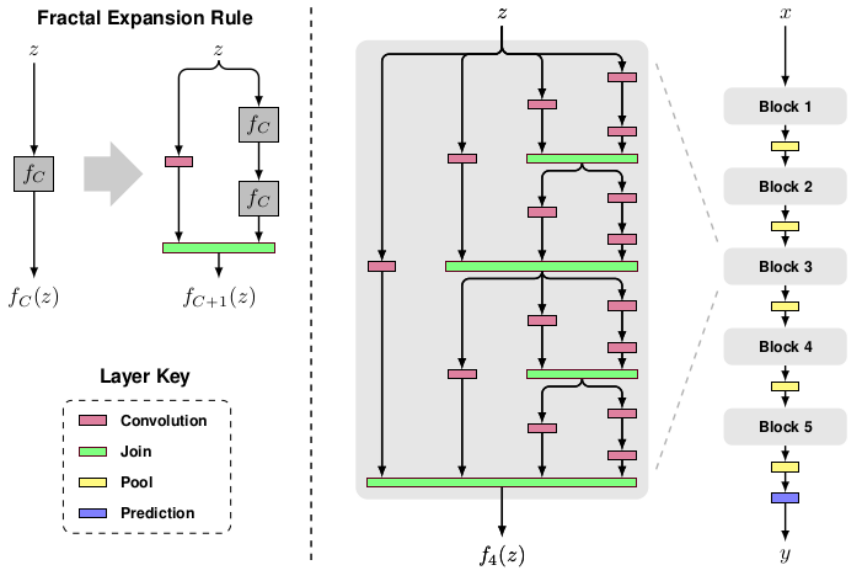


Figure 3.1: Left shows the basic expansion rule with a single layer of the desired type (in this case convolutional). Right shows an example FractalBlock arising from the expansion rule where we have 4 intertwined columns of varying depths. Diagram sourced from Larsson et al. [2016]

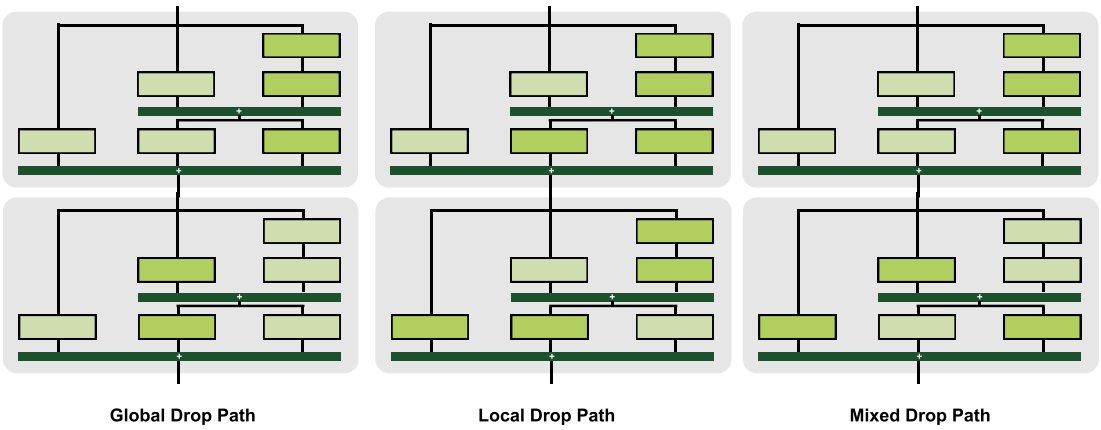


Figure 3.2: Visual representation of the three different drop-path strategies discussed in Larsson et al. [2016]. Vibrant green cells show active layers while muted green show dropped layers. Left shows global drop path where a single column is picked for each block in the architecture. Center shows local drop path where layers are randomly dropped per join layer. Right shows the strategy used here and in the original paper where both approaches are combined.

Several other benefits of this approach, compared to its contemporaries, are offered by the authors. These include a form of automatic deep supervision [Lee et al., 2015] as in ResNets and perhaps more interestingly, a form of knowledge distillation [Hinton et al., 2015] as a result of the topology. According to their results, deeper networks in this arrangement seem to guide and improve the learning of the shallower and faster networks higher up in the fractal structure. This in turn imbues these networks with an 'anytime property' where the shallow subnets can provide a useful but faster answer and deeper ones a slower, more accurate one.

3.3 Application to Speech Representation Learning

Given the fact that both of these approaches were designed with Computer Vision (CV) tasks in mind and utilizing convolutional architectures, they need to be adapted to the speech representation learning domain. In particular, I consider the following decisions going forward:

1. **Handling Recurrent Architectures with no Residuals**: recurrent architectures such as LSTMs and GRUs are often much better suited to learn insights from sequential data such as speech. However, these models lack residuals that are necessary for the successful implementation of Stochastic Depth, meaning I will need to incorporate these into my models as a baseline (see Section 4.2)
2. **Stochastic Depth Probability Rule**: layer-wise analysis of LSTM and Transformer models might be able to find a probability rule that would better suit these networks. However, in the interest of enabling direct comparison with the original paper's results, I limit myself to the same linear-decay and constant probability rules proposed by Larsson et al..

Chapter 4

Experimental Setting

This chapter will outline my methodology and provide some motivation for the specific experiments I carry out and present in Chapters 5 through 7. In addition to discussing the data used and the baseline models I will be considering for the pre-training task, I also provide some justification and details for a further batch of experiments on downstream objectives covered in Chapter 7. Finally, I provide some preliminary hypotheses about the performance of the techniques on both APC and the downstream tasks.

4.1 Datasets

All of my pre-training experiments will be carried out on the 100 hour clean training subset of the Librispeech dataset (LS100) [Panayotov et al., 2015]. This dataset consists of speech sourced from LibriVox audio books and is also the dataset used in the APC paper. However, contrary to Yang et al., I decide to use the 100 hour subset in favour of the larger 360 and 960 hour counterparts. The reasoning for this is two-fold:

1. Since pre-training can be seen as a form of parameter initialization [Hinton and Salakhutdinov, 2006], I expect that a larger dataset will only have an impact on the performance of the models on the downstream tasks. Given my objective is not to achieve competitive results on these tasks but rather to provide an analysis of the impact of these techniques on self-supervised learning, the size of the dataset used should not alter my analysis.

2. Due to the large number of networks I am interested in comparing, it makes sense to use less data to have a faster turn-around of results.

Furthermore, two other datasets will be considered when probing the pre-trained models on the downstream tasks (see 4.3). These are the Wall Street Journal (WSJ) dataset for phone classification [Paul and Baker, 1992] and the Voxceleb1 (Vox1) dataset for speaker verification [Nagrani et al., 2017]. The Wall Street Journal dataset includes about 80 hours of journalist-read news whereas Vox1 is composed of 352 hours of speech extracted from YouTube videos of 1,251 different celebrities. Experiments on WSJ involve training on the `si284` subset and testing on the `eval92` and `dev93` subsets.

In terms of data preparation, I once again follow Yang et al.'s paper by extracting 40-dimensional log Mel features with a 25 ms window and a 10 ms hop. These features are also normalized using global mean and variance computed on the relevant training sets. Similarly, I also use their same forced phone alignments on WSJ as the ground truth for the phone classification objective. These are sourced from a speaker-adaptive GMM-HMM based on Kaldi's `tri3b` recipe [Yang et al., 2022, Povey et al., 2011].

4.2 Baseline Models

In order to evaluate how the proposed techniques scale to deeper models, I will be considering both LSTM-based and Transformer Encoder-based approaches since sequential processing in recurrent architectures limits their ability to capture long-range dependencies independent of depth [Vaswani et al., 2017]. As such, deep transformer models are much more common in the literature and feature in many more deep learning architectures, including many state-of-the-art (SOTA) models [Baevski et al., 2020]. Moreover, as previously discussed, LSTM models require a residual counterpart in order to make Stochastic Depth applicable to them. These requirements lead me to the following three baselines:

- LSTM Baseline: 6 layer unidirectional LSTM with 256 hidden dimension.
- LSTM Skip Baseline: 6 layer unidirectional LSTM with 256 hidden dimension *and skip connections*. This enables a fair point of comparison for LSTM-based stochastic depth approaches.
- Transformer Encoder Baseline: 10 layer Transformer Encoder with 2 attention heads and 256 hidden dimension. In order to prevent these models from attending to future frames and therefore 'cheating' the APC objective, I also apply a causal attention mask to these models.

4.3 Probing on Downstream Tasks

In order to assess the information quantity and availability in the representations learned during pre-training, I will be probing the APC networks on a set of downstream tasks. Although fine-tuning is more common in the literature, it would introduce an additional confounding factor to my analysis of the learned representations, which is why I favour the probing approach. For each of the following tasks, I will therefore be freezing the APC network layer-wise and training a linear classifier head which takes the APC network's output as input and produces whatever the downstream task requires. In particular, I will explore:

1. **Phone Classification (PC) on WSJ**: this first task will attempt to learn the phonetic information of each of the utterances from the representations learned. The original APC paper demonstrates that a regular LSTM can learn a latent space where the necessary phonetic and timing information required to achieve a low error on PC is preserved. I will therefore be exploring whether the techniques in Chapter 3 will replicate this same success. The phone set used for this task

consists of 39 phones including a token for silence and spoken and non-spoken noise. I also replicate their choice of optimizer and hyperparameters, training the linear layer using a learning rate of 10^{-3} and a batch size of 16 for 10 epochs with the Adam optimizer [Kingma and Ba, 2014].

2. **Speaker Verification (SV) on Vox1**: this second task will evaluate the usefulness of the APC representations in capturing speaker-independent characteristics (vocal tract length, speaking style, accent, etc.). As in Yang et al.'s paper, I compute a frame-wise mean of the LSTM or Transformer output and use it as the speaker embedding without any additional training. I then use these embeddings and the Vox1 verification dataset to compute Equal Error Rates (EERs) for each model layer-wise. The choice of hyperparameters also remains largely unchanged, with a learning rate of 10^{-3} using a batch size of 16 and an Adam optimizer for 10 epochs.

4.4 Hypotheses

Finally, as motivation for the coming sections and based on the related work and my preliminary understanding, I proceed with the following set of hypotheses.

H1: **Stochastic Depth will have a larger negative impact on shorter models than on deeper ones**. This follows from the understanding that deeper, overparameterized models do not need to rely on all intermediate layers to learn their objective. Moreover, earlier layers have been shown to be particularly important in residual architectures meaning a constant assignment is expected to reduce performance more than the proposed linear decay rule [Li and Papyan, 2024].

H2: **FractalNets will be more accurate than Stochastic Depth networks** both in terms of pre-training loss and downstream performance. This is due to the implicit deep supervision enabling faster training in the same number of epochs. However, due to the larger model architecture, *pre-training time will not be as greatly reduced.*

H3: **With either approach, the phonetic information is more likely to be lost than information relating to speaker identity**. I believe this is a corollary of the fact that speaker characteristics can be extremely broad and varied whereas phonetic information is much more specific and prone to be perturbed by the proposed methods. I then hypothesize that performance on the PC downstream task will be more negatively impacted than for the SV task

Chapter 5

APC Pre-Training on Stochastic Depth Networks

This chapter will present the results of pre-training LSTM and Transformer networks using the Stochastic Depth approach. I will look at each model type in turn, presenting their architectures and evaluation metrics and then proceed to offer some analysis of the results attained. In the last section, I also investigate the sensitivity of Stochastic Depth networks to the choice of hyperparameter in the speech domain.

5.1 LSTM Experiments

First, I carry out experiments relating to Stochastic Depth on LSTM-based architectures. In particular, I consider models that use a constant probability rule of 0.5 ($p = 0.5$) and the linear decay rule ($p = DECAY$) suggested in the paper with final probability of 0.5 (see Chapter 3) [Huang et al., 2016]. The choice of 0.5 as my survival probability is also motivated by Huang et al.'s original findings showing that this setting outperformed most other assignments for the networks they were testing, providing the greatest speedup with the minimal accuracy trade-off.

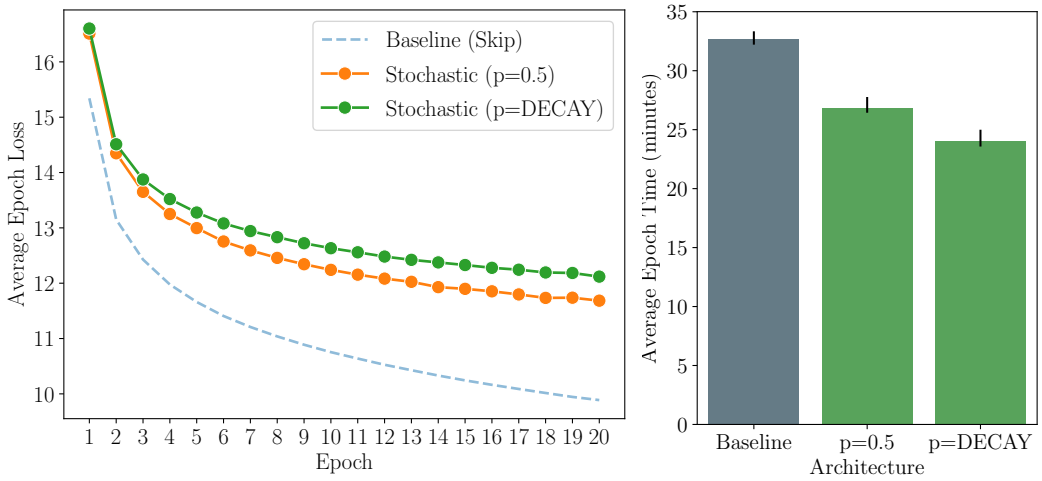
Table 5.1 shows the different stochastic models and the two baselines we use in this section alongside some summary statistics pertaining to model performance and size.

Model	Layers	E(Layers)	Theor. Speedup	Backprop (ms)
Baseline-Skip	6	6	1x	391.61
Stochastic (p=0.5)	6	3	2x	276.42
Stochastic (p=DECAY)	6	4.25	1.4x	252.44

Table 5.1: Summary Statistics for LSTM Models. Note the expected number of layers for the linear-decay rule with 0.5 final probability is $(3L-1)/4$. Backpropagation times are reported using batch size = 16 on a Titan-X GPU

I train these three models using the APC objective with a time-step of 5 frames, a learning rate of 1e-3 and a batch size of 16 over 20 epochs. To enable time comparisons,

all LSTM models are trained using a single Nvidia GTX 1060 GPU. Figure 5.1a shows the training loss curves for these models and figure 5.1b shows a barplot comparing the average epoch time in minutes for each network.



(a) LSTM APC Training Loss

(b) Avg. Epoch Times

Figure 5.1: (Left) LSTM APC Training Curves (20 epochs). As a reminder, I use k=5 frames into the future for the APC objective. The effect of stochastic depth networks is clear with a shallower curve and higher final training loss. (Right) LSTM APC Average Epoch Time with error bars showing range. The time savings provided by the method are again clear from inspection.

As expected, we can see that the probability rules chosen have a significant impact on the final APC performance, with both stochastic depth networks seeing around a 22% increase in final training loss compared to the skip baseline. Interestingly, however, the linear decay rule seems to affect APC performance more negatively than the constant assignment which could be due to the importance of the final layers to the specific objective being trained [Hermans and Schrauwen, 2013]. Moreover, we attain a considerable reduction in average training epoch time in both cases over the baseline. Per epoch, the constant probability rule results in a 19.71% time saving and the linear decay rule reduces time by 30.32%. In the current context this accumulates to a total of 2-2.5 hours saved over the skip baseline. However, it is important to note that these numbers do not exactly reflect the theoretical speedup outlined in table 5.1. This is likely due to the memory overhead in the GPU cluster used to run these experiments. It is therefore expected that with more optimal hardware, these figures will approach their theoretical values. Finally, as a means of quantifying the generalization capacity of these networks, Table 5.2 below shows the best validation loss attained after 20 epochs. The validation results give a much more optimistic picture of the effect of Stochastic Depth on the training objective with both networks performing very comparably to the baseline network.

Model	Best Validation Loss	Epoch	+/-
Baseline-Skip	12.4585	9	0.0
Stochastic (p=0.5)	12.6601	20	**+1.62%**
Stochastic (p=DECAY)	12.9041	20	+3.58%

Table 5.2: Best validation losses on the APC objective after 20 epochs of training along with epochs they are attained at. Note that the baseline model undergoes some light overfitting as the best validation loss is obtained quite early on during training. This is not the case in the stochastic networks, both obtaining their best validation results in the final epoch.

5.2 Transformer Encoder Experiments

In the following, I apply the same two probability rules explored in Section 5.1 above to a set of deeper transformer-based networks. While my methodology remains largely unchanged, it is worth noting that applying the Stochastic Depth technique to Transformer models does not require introducing any additional residual connections. Instead, I will be leveraging the existing residuals and stochastically dropping either the Multihead Attention block or the Feedforward block (or both) in each case. As before, summary statistics for the models considered are outlined in Table 5.3 below.

Model	Layers	E(Layers)	Theor. Speedup	Backprop (ms)
Baseline	10	10	1x	21.80
Stochastic (p=0.5)	10	5	2x	12.95
Stochastic (p=DECAY)	10	7.25	1.4x	12.85

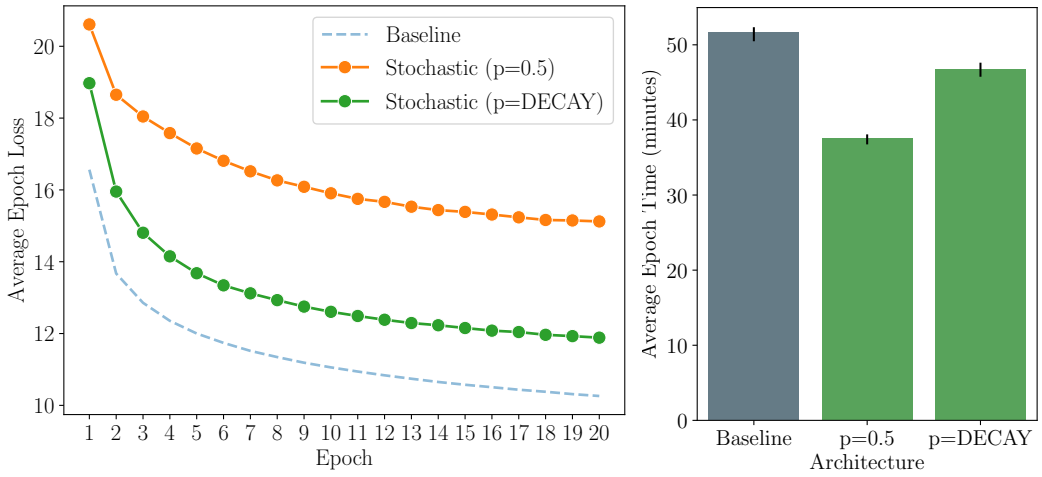
Table 5.3: Summary Statistics for Transformer Models. Backpropagation times are reported using batch size = 8 on a Titan-X GPU

I once again train these models on APC with a time-step of 5 frames for 20 epochs but use a batch size of 8 and a learning rate of 5e-4 for all transformer experiments. In order to ensure a consistent time comparison, all of these models are also trained on a single Nvidia Titan-X GPU. This results in the training loss curves and epoch time barplots depicted in figures 5.2a and 5.2b respectively.

In this instance, we can see that the linear decay rule vastly outperforms the constant assignment in terms of final pre-training loss with a mere 14.6% increase over the baseline. Notably, these results also corroborate my initial hypothesis suggesting that Stochastic Depth would be more successful of a technique for deeper networks in the speech domain. Moreover, the difference in training loss curves here is likely due to the following two observations:

1. **Later Layers become more Redundant as Depth Increases**: this is consistent with the understanding that earlier and middle layers tend to capture most acoustic information of interest in Transformer networks [Pasad et al., 2021].

2. **Dropping Sub-Layer Components Hurts Performance**: this follows from the



(a) Transformer APC Training Loss (b) Avg. Epoch Times

Figure 5.2: (Left) Transformer APC (k=5) Training Curves (20 epochs). We once again see shallower curves with higher final values but p=DECAY assignment presents itself as the better compromise. (Right) Transformer APC Average Epoch Time with error bars showing range. While less pronounced, time savings are again notable.

understanding of Stochastic Depth as a dropout-adjacent technique to reduce co-adaptation of parts of the network. In other words, independently training layer sub-components could be causing them to diverge in terms of objectives and inhibit them from cooperating with each other as intended.

On the subject of time savings, we once again notice some disparity with the original LSTM results. This time, the constant probability rule reduces the average epoch duration considerably more than the linear decay assignment as we would expect. Despite this, the more suitable linear decay rule still manages to save around 5 minutes per epoch on average resulting in a 9.53% saving (approximately 1.5 hours in total here). This could be due to the fact that self-attention computations dominate the training time in this case, giving the stochastic depth networks a greater edge over the baseline. Once again, these patterns also resurface in the APC generalization results. Table 5.4 shows the best validation losses along with the relative change over the baseline, cementing the linear decay rule's superior performance.

Model	Best Validation Loss	Epoch	+/-
Baseline	12.4213	14	0.0
Stochastic (p=0.5)	14.5262	20	+16.9%
Stochastic (p=DECAY)	12.8038	19	**+3.08%**

Table 5.4: Best validation losses on the APC objective after 20 epochs of pre-training on Transformer networks. Baseline once again undergoes some very light overfitting. The constant probability rule greatly affects the capacity of the network to generalize the APC objective.

These results highlight the importance of the probability rule of choice to the success of Stochastic Depth networks. In particular, it is clear that this hyperparameter shouldn't be fixed in isolation and should instead account for the objective, model architecture and depth being used. This setting is explored in the next section.

5.3 Exploring the Probability Rule

In an attempt to explore the sensitivity of this technique to the choice of survival probability in the speech domain, I consider a range of smaller experiments on Stochastic Depth architectures with different probability rules. In particular, the following considers a subset of 10 epochs of the APC pre-training for stochastic depth networks using both the constant probability and linear decay rules explored in Sections 5.1 and 5.2 above. Survival probability assignments are considered in intervals of 10% from p=10% to p=100% where a probability assignment of 100% means inputs don't go through any layers in the network and an assignment of 0% means the input gets passed through all layers (baseline performance). Since this hyperparameter selection aims to optimize two separate objectives (loss and epoch time), I choose to visualize the relationship of each of these in a two-dimensional scatter plot depicted in Figure 5.3. Furthermore, as I am dealing with a pre-training task, I choose the validation loss as my loss metric of choice. The plot labels Pareto optimal solutions with their specific survival probability rule.

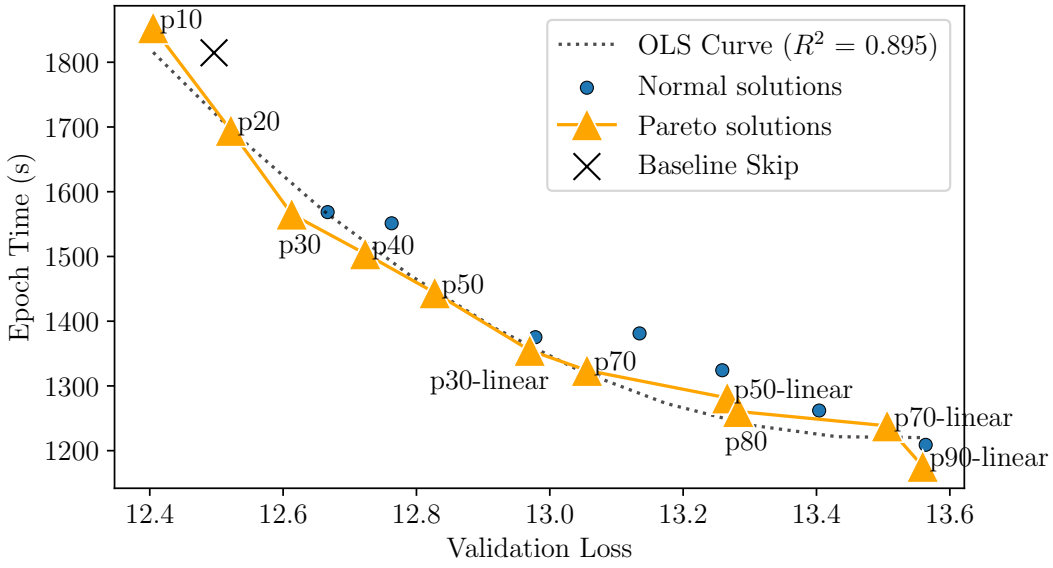


Figure 5.3: LSTM Stochastic Depth Epoch time vs APC Validation loss Trade-Off across different hyperparameter assignments. Blue circle markers indicate dominated hyperparameter settings while orange triangle markers show all non-dominated settings, constituting the Pareto front. A clear inverse quadratic relationship can be observed and is illustrated by the Ordinary Least Squares regression line. The cross marker represents the baseline skip LSTM network parameter combination which we note is a dominated solution. Further discussion in the text below

As we can see, the results clearly shows the inverse (quadratic) relationship expected between the choice of the survival probability rule and the two objectives being minimized. This suggests that smaller to moderate survival probability assignments might be better suited for this type of network as larger probability rules incur a larger increase in validation loss and a marginally diminishing reduction in epoch time. We also note that the best assignment found in Section 5.1 (constant 0.5 survival probability) features in the Pareto optimal set, providing a good balance between epoch time and APC validation loss which further justifies this setting. Finally, it is worth noting the general split observed in the Pareto-optimal probability rule type. Namely, constant probability rules appear to be better suited for smaller thresholds being used while larger probability thresholds favour from the linear decay rule proposed in Huang et al. [2016]. This relationship is to be expected as linear decay assignments tend to be more accurate in general at the cost of slower training times while the inverse is true of constant assignments.

Overall, moderate constant assignments such as p=50% or p=70% or even smaller linear decay assignments (p = 30% DECAY) appear to be best suited for LSTM models. While I expect we would obtain a similar relationship in other models, these results reinforce the need for similar explorations to be carried out in order to inform the specific ideal hyperparameter setting for other networks.

5.4 Summary

In this chapter, I presented and interpreted results from Stochastic Depth networks on the APC pre-training task. In particular, I showed how a suitable survival probability rule assignment can result in significant time savings while minimally sacrificing on the ability of the network to generalize the objective. After establishing the success of the technique in the self-supervised speech representation learning domain, I discussed and analysed some differences in the results attained between the two main architectures (LSTM and Transformer Encoder). My results corroborated my initial hypothesis that Stochastic Depth networks are more successful with deeper models, while results can greatly vary with shallower networks. This in turn motivated a further set of experiments exploring the effect of the survival probabilities on the network's learning. These results showed a clear inverse relationship between different probability assignments and the objective combination of epoch time and validation loss, consistent with my expectations. Furthermore, they showed that moderate constant survival probability rules such as p=50% provide a good balance between both objectives in LSTM networks.

In the following chapter, I will be exploring the second and final technique to speed up pre-training of deep neural networks: FractalNets.

Chapter 6

APC Pre-Training on Fractal Networks

This chapter explores the second approach to speeding up APC pre-training of LSTM and Transformer architectures, namely FractalNets. I will again be looking at the effect of this technique on each model type and analysing the results attained in either case. In the last section of this chapter, I also explore the impact of the width of each Fractal block to the overall performance and time-saving potential of the method.

6.1 LSTM Experiments

Both of my fractal architectures in this section will be using stacks of 2-column Fractal Blocks as depicted in Figure 6.1. On top of this, I will be replicating the original authors' mixed drop-path strategy, choosing to drop all but a single column (global) half of the time and randomly dropping inputs to each join layer the other half (local). In order to ensure a valid path is available in the network, local drop-path will not drop an input to a join layer with fewer than 2 connections going into it. Moreover, I will explore two threshold settings for local drop-path, 15% and 50%. The 15% rule is what Larsson et al. use in their paper, whereas the 50% rule is provided to examine the effect of a more aggressive dropout rule on the training time. This first section will use a single LSTM layer as the most basic unit and consider networks that are 3 Fractal Blocks deep, such that their deepest column is exactly 6 LSTM layers deep. More information on the different models considered and their baselines are provided in Table 6.1

Model	Total Layers	Deepest Column	E(Layers)	Backprop (ms)
Baseline	6	6	6	285.33
Fractal (p=0.15)	9	6	≈6.4	330.67
Fractal (p=0.5)	9	6	≈5.6	337.34

Table 6.1: Summary Statistics for Fractal LSTM Models. The probability p refers to the local drop-path probability used for the Fractal model and "Deepest Column" refers to the number of layers along the deepest column in the architecture. Backpropagation times are reported using batch size = 16 on a Titan-X GPU

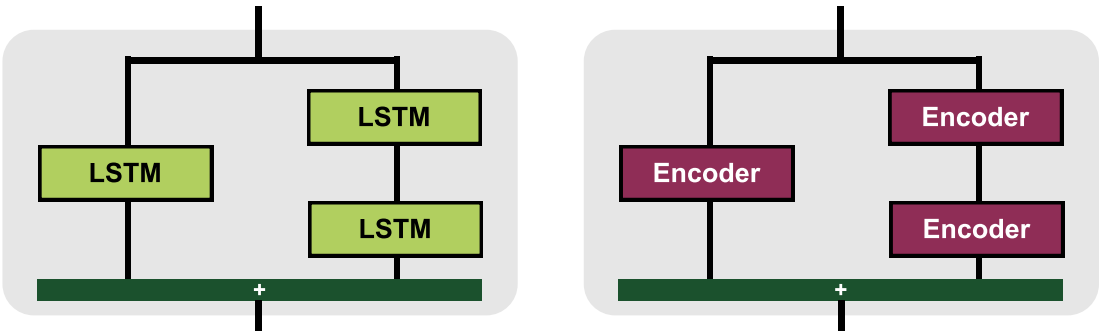
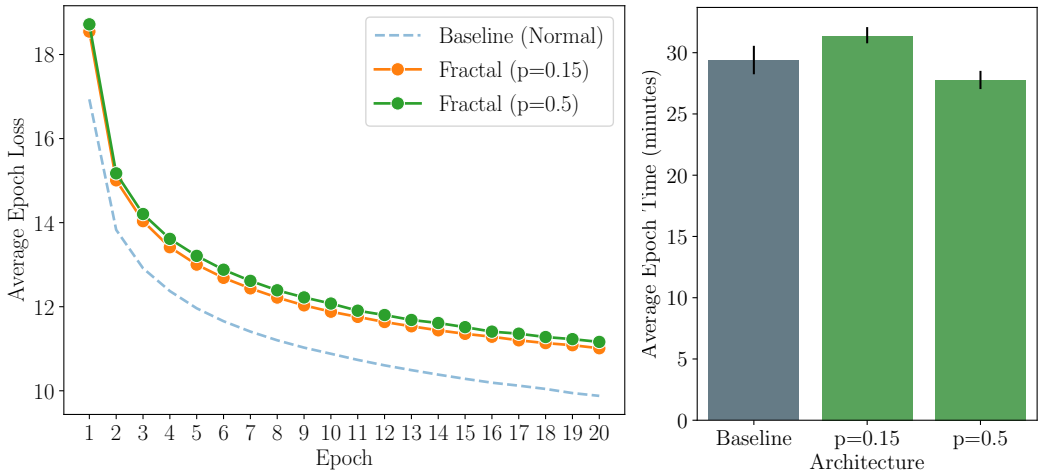


Figure 6.1: Two-column Fractal Architecture used for fractal models in this chapter. Left shows Fractal LSTM and right shows Fractal Transformer where each "Encoder" corresponds to a full Transformer Encoder. The dark green band labelled "+" is the join layer which I choose to be element-wise averaging.

As before, I train these three models using the APC objective with an offset of 5 frames and a batch size of 16 for 20 epochs on a single Nvidia GTX 1060 GPU in each case. As with stochastic LSTM experiments, I use an Adam optimizer with a 1e-3 learning rate. Figure 6.2a shows the training loss curves for each model along with a barplot comparing their average epoch times in minutes.



(a) Fractal LSTM APC Training Loss

(b) Avg. Epoch Times

Figure 6.2: (Left) Fractal LSTM APC (k=5) training loss over 20 epochs. "Baseline (Normal)" refers to a regular LSTM with no residual connections and "p" refers to the local drop-path threshold being used. Similar trends to those observed in Stochastic Depth networks are also present here. (Right) Fractal LSTM Average Epoch Time with error bars showing range. Localdrop 15% assignment is insufficient to improve pre-training times making the larger 50% assignment more suitable.

Although we once again see that the proposed architecture fails to perfectly match the training dynamics of the baseline network, the compromise here is notably smaller than with Stochastic Depth networks, even for the more aggressive 0.5 local drop-path

rule. This time, both architectures see around an 11.5% increase in final training loss compared to the almost double 22% seen in Section 5.1. In fact, the local drop-path threshold appears to make very minimal difference in the overall shape of the training curve this time around which could be due to the relatively "narrow" blocks being used in this network. In other words, training in this fashion seems to produce independently performant branches very early on which result in the final training loss being relatively unaffected by the choice of dropout strategy.

Naturally, this initially optimistic analysis is counteracted by the much less impressive time savings achieved with this method. In fact, we observe that the 0.15 local drop-path rule fails to improve training time over the baseline 6-layer LSTM, despite also using global drop path 50% of the time. Local drop-path of 0.5, however, does manage to moderately reduce training times, achieving a modest 5.3% reduction per epoch on average which totals to about 0.5 hours of time saved in this particular context. While this time saving is significantly inferior to Stochastic Depth networks, the much improved loss curve and generalization capacity (see Table 6.2) could justify the practical use of this approach over its simpler alternative. Moreover, these results once again support my initial hypothesis about the relative performance of FractalNets compared to Stochastic Depth networks at least in the pre-training domain (H2).

Model	Best Validation Loss	Epoch	+/-
Baseline	12.691	9	0.0
Fractal (p=0.15)	12.826	19	**+1.06**%
Fractal (p=0.5)	12.847	19	+1.23%

Table 6.2: Best validation losses on the APC objective after 20 epochs of pre-training on Fractal LSTM networks. Validation results are obtained by using the deepest column in the architecture. Results show that either local probability rule manages to get within almost a single percent of the generalization capacity of the baseline network. This is a considerable improvement over both Stochastic Depth networks discussed in the previous section

6.2 Transformer Encoder Experiments

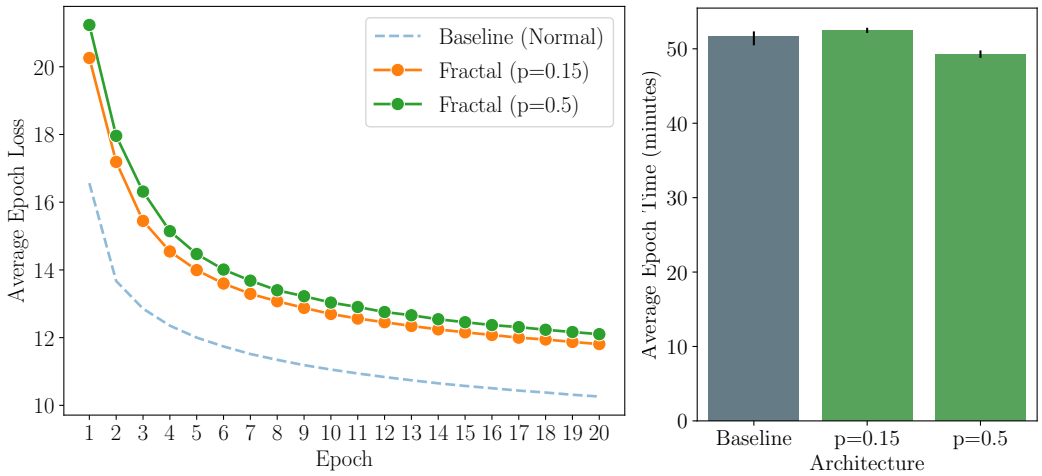
This section considers fractal architectures with Transformer Encoder blocks as the most basic neural unit. Such an encoder block consists of an embedding linear layer, positional encoding, multihead attention with 2 attention heads and a final feedforward linear layer with layer normalizations as described in Section 2.2.2. Moreover, I use a stack of 5 Fractal Blocks in this case so as to achieve networks with a deepest column consisting of 10 stacked encoder blocks. This then enables comparison against the same baselines used in the Stochastic Depth Transformer experiments. As before, a more complete description of the 3 models considered here are provided in Table 6.3.

Training also proceeds in a similar fashion to Stochastic Depth transformers. The same APC objective with offset 5 is used and networks are trained for 20 epochs on a single Nvidia Titan-X GPU using a batch size of 8. The same Adam optimizer and 5e-4

Model	Total Layers	Deepest Column	E(Layers)	Backprop (ms)
Baseline	10	10	10	21.80
Fractal (p=0.15)	15	10	≈10.7	17.33
Fractal (p=0.5)	15	10	≈11.3	13.15

Table 6.3: Summary Statistics for Fractal Transformer Models. The probability p refers to the localdrop probability used for the Fractal model and "Deepest Column" refers to the number of layers along the deepest column in the architecture. Backpropagation times are reported using batch size = 8 on a Titan-X GPU

learning rate used for the stochastic transformers is also used here. Figure 6.3a shows the training loss curves along with the average epoch time comparison.



(a) Fractal Transformer APC Training Loss

(b) Avg. Epoch Times

Figure 6.3: (Left) Fractal Transformer APC (k=5) training loss over 20 epochs. "Baseline (Normal)" refers to a regular 10-layer Transformer with no residual connections and "p" refers to the local drop-path threshold being used. As before, the local drop-path assignment seems to have a near negligible effect on the final training loss. (Right) Fractal Transformer Average Epoch Time with error bars showing range. Localdrop 15% assignment is again insufficient to improve pre-training times making the larger 50% assignment more suitable.

Contrary to Stochastic Depth networks, results on these deeper fractal networks remain surprisingly unchanged from their shallower counterparts'. Once again, both probability rules show negligible differences in training after the first few epochs and both terminate at a near-equivalent training loss. This final training loss sees a slightly higher 15% increase over its respective baseline, but remains comparable to the 14% increase attained by the Stochastic Depth Transformer using the linear-decay rule. Similarly, the same pattern in training times is observed here, with the 15% local drop-path rule failing to improve on the baseline 10 layer transformer and the 50% rule achieving a small 4.57% reduction in training time per epoch, amounting to a 45-minute saving.

Generalization performance (Table 6.4) for the two networks is once again comparable, albeit marginally worse than the best performing Stochastic Depth transformer. The difference in probability rules here could be due to many factors, most prominently random initialization, but it could also be indicative of the fact that deeper fractal architectures are more sensitive to the dropout strategy being used. As explained by Larsson et al., the more aggressive rule promotes the independent learning of each column in the block which seems to allow the deeper column to attain better generalization in this case. While this local trend is expected, the decrease in generalization performance over the Stochastic Depth model was not. Beyond random initialization, a potential reason behind this is that deep stacks of shallow fractal blocks do not combine well together, especially when isolating a single branch as we are doing here. The original paper was working with much wider fractal blocks which could potentially help to eliminate, or at least mitigate, this issue. I will explore this hypothesis in the next and final section of this chapter.

Model	Best Validation Loss	Epoch	+/-
Baseline	12.4213	14	0.0
Fractal (p=0.15)	13.0744	19	+5.26%
Fractal (p=0.5)	12.9887	20	**+4.56**%

Table 6.4: Best validation losses on the APC objective after 20 epochs of pre-training on Fractal Transformer networks. Validation results are obtained by using the deepest column in the architecture as before. The larger local drop-path seems to improve generalization significantly but is unable to match the 3% increase in validation loss attained by the linear decay Stochastic Depth transformer.

6.3 Exploring the Effect of Fractal Block Width

In order to assess the impact of fractal block width on the final generalization performance of the deepest column in the architecture, we consider a new set of Transformer Encoder FractalNets along with a new 8-layer transformer baseline. Table 6.5 summarizes the specific network architectures studied in this section. Validation loss curves are depicted in the line plot in Figure 6.4. Table 6.6 shows the final (not best) validation losses attained by each model type after 20 epochs along with the average epoch times for each model.

These results clearly illustrate how wider fractal blocks are typically correlated with a moderate increase in model validation loss. This may appear to be initially contradictory but can be understood by realising that the drop-path rule is not changed despite gradually reducing the depth of the network in terms of number of fractal blocks. As such, we would expect that in the 4-column network, the deepest 8-layer column would be dropped a lot more frequently than the deepest 4-layer columns in the two fractal blocks of the 3-column network. This in turn explains the loss in performance observed and perhaps motivates an alternative drop-path assignment as wider fractal blocks are used.

Model	Fractal Blocks	Total Layers	Deepest Col.	Backprop (ms)
Baseline (8-layer)	0	8	8	10.96
Fractal 2-col (p=0.5)	4	12	8	11.42
Fractal 3-col (p=0.5)	2	14	8	9.93
Fractal 4-col (p=0.5)	1	15	8	10.08

Table 6.5: Summary Statistics for Fractal Transformer Models explored in this final section. In order to facilitate comparison, we consider a new 8-layer baseline along with different fractal transformers each with a deepest column consisting of 8 transformer encoders. The same drop-path rule as before is used for all fractal models. Backpropagation times are reported using batch size = 8 on a Titan-X GPU

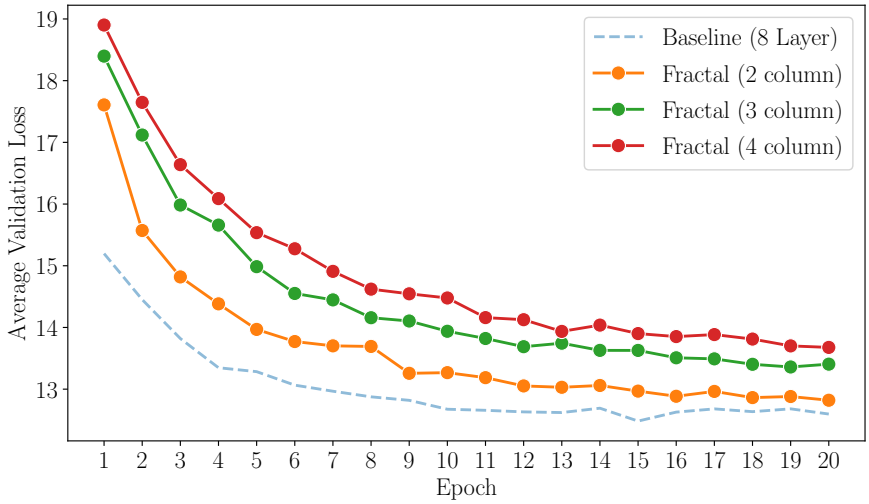


Figure 6.4: Transformer APC Validation Loss for different Fractal Transformers with varying block widths. Validation loss is computed using the deepest column of each architecture. As the number of columns in a fractal block increases, the validation loss curves flattens and terminates in a marginally higher final validation loss.

Model	Val. Loss	Avg. Time (min.)	+/- Loss (%)	+/- Epoch Time (%)
Baseline	12.60	47.26	0.0	0.0
Fractal 2-col	12.82	45.46	**+1.75**	-3.81
Fractal 3-col	13.41	43.10	+6.43	-8.80
Fractal 4-col	13.68	42.40	+8.57	**-10.28**

Table 6.6: Transformer APC Validation Loss results after 20 epochs of training on the deepest column of different fractal architectures with varying block width. We can see that increasing the block width moderately hurts final validation loss but also manages to greatly improve average epoch time. All figures are given to 2 decimal places.

On the other hand, these same findings also illustrate a clear benefit of using wider fractal blocks in the time savings reported. In fact, we see a monotonic decrease in proportional average epoch times across these networks, with the 4-column transformer

achieving a significant 10% decrease in epoch time over the baseline. This observation could make the time-saving potential of these networks more appealing and possibly give them the edge over Stochastic Depth approaches in certain scenarios. Once again, this analysis is limited by the time and resources I had available to me but exploring how these time savings scale with deeper FractalNets could reveal less compromised results that could make a strong case for these networks to be used over other alternatives.

6.4 Summary

In this chapter I provided and interpreted results from FractalNet architectures on the APC pre-training task for both LSTM and Transformer-based models. In particular, I demonstrated the success of the technique in providing a less compromised approximation of baseline pre-training while still managing to save a substantial amount of training time provided a suitable form of drop-path is used. Moreover, my results on shallow LSTM networks once again supported my initial hypothesis that Fractal architectures would outperform their Stochastic Depth counterparts in terms of pre-training performance. However, deeper fractal networks showed inferior generalization capabilities which I suspect is likely due to differences in random initialization and using stacks of very shallow fractal blocks. This observation also prompted further investigation into the effect of Fractal block width. These experiments concluded that wider fractal blocks result in a more compromised approximation using the same drop-path strategy but could also provide an avenue to increase the time saving potential of FractalNets applied to self-supervised pre-training, especially in deeper FractalNet architectures.

My results now cement both approaches as viable techniques to achieve approximate pre-training on the APC objective. More interestingly, they also present a useful trade-off between approximation accuracy and potential time savings, with Fractal Networks excelling at the former. Moreover, I showed that fractal networks are significantly more robust to the choice of hyperparameters, consistently providing good training results without requiring careful tuning or parameter searches as in Stochastic Depth networks. This ease of use could also become a compelling reason for practitioners to favour Fractal architectures over the simpler stochastic depth alternative.

In the following chapter, I will be completing my analysis of these techniques by examining their impact on the speech representations learned. By assessing the performance of these networks on the previously discussed downstream tasks, I will be able to determine how accessible and useful information remains in the new representations learned and, by extension, whether these approaches could ultimately be practical in the self-supervised speech domain.

Chapter 7

Probing Learned Representations on Speech Downstream Tasks

The preceding two chapters have discussed the success of the proposed techniques in providing good approximations of baseline APC pre-training while also managing to significantly reduce training times. However, the discussion so far has been limited to performance on the APC objective. Despite this, pre-training only constitutes a form of initialization in the self-supervised framework which cannot be considered in isolation. Instead, the practicality of these networks is subject to their capacity to generalize and adapt to various common speech downstream tasks through fine-tuning or probing.

In this final chapter, I will therefore be exploring the speech representations learned by the APC objective on the Stochastic Depth and FractalNet architectures in order to determine the quality and availability of the information retained relative to their respective baselines. In order to achieve this I will be probing each of the pre-trained networks on two different tasks, namely Phone Classification and Speaker Verification. For either subset of experiments, I will consider a layer-wise analysis so as to compare how each layer in the proposed networks performs relative to their counterpart in the baseline models. It is worth noting that ”layer-wise” here means I will be considering models composed of all layers up until the one being considered rather than the layer in isolation. Another point to note is that the ensuing analysis does not concern itself with demonstrating the competitive potential of the networks studied on the Speaker Verification objective. Instead, I consider the same set of simple baseline networks discussed in the previous two sections. Identifying and analysing any potential loss in performance in these baselines can still illustrate how we would expect these dynamic architectures to scale with larger competitive networks, which motivates this discussion.

The chapter looks at each downstream task in turn starting with Phone Classification in Section 7.1. In each case, I start by motivating the task of choice before going over my methodology and layer-wise analysis.

7.1 Phone Classification (PC)

Phone Classification aims to map the hidden representations learned for each frame by the APC networks onto a set of 39 phones without stress markers. As in Yang et al. [2022], the phone set also includes special tokens for silence and spoken noise. Success in this objective would suggest the networks learn durations and timings on top of the necessary phonetic information. Good performance on the WSJ datasets would also demonstrate that this phonetic understanding is speaker-independent as the test sets provided do not share any speakers with the training subset (see Section 7.1.1 below). Finally, proof that this sequential information is still readily available despite the approximations used would also entail that similar sequential data required for more traditional Automatic Speech Recognition (ASR) tasks should also remain present here.

7.1.1 Methodology

Phone Classification experiments are carried out on the Wall Street Journal (WSJ) dataset as previously mentioned [Paul and Baker, 1992]. For each network type, I train a single linear layer on top of the frozen pre-trained models to map the hidden representations onto the set of 39 phones included in WSJ for each frame. Training proceeds on the `si284` subset, similarly to Yang et al. [2022] where I use the phone alignments produced by a speaker-adaptive GMM-HMM based on Kaldi's `tri3b` recipe as the ground truth labels for each frame. This foregoes the need to use any form of alignment between the model's output and the phonetic transcription while still measuring how much of the timing and duration information is preserved. I use a simple cross entropy loss along with an Adam optimizer with learning rate 1e-3 to learn the parameters of the probing layer. After 10 epochs of training, I test the performance of the new models with their additional probing layers on the `dev93` and `eval92` subset and report Phone Error Rate (PER) for each as in the original paper. PER is calculated using the counts of insertion (I), substitution (S) and deletion (D) errors as specified in Equation 7.1 below (although there will only be substitution errors in this setting). Other summary statistics on the datasets used are outlined in Appendix A (Table A.1).

$$PER = \frac{I + S + D}{N} \times 100 \tag{7.1}$$

7.1.2 Layer-Wise Analysis

Analogous to Yang et al. [2022], I consider the performance of each frozen APC layer on the downstream task. This layer-wise analysis not only verifies that each part of the new networks behaves similarly to their baseline counterparts but also enables us to identify what the best performing layers are in the proposed architectures which could serve as a guide for other practitioners. After determining the best performing layer using the `eval92` set, I then report performance on both test sets for the best performing layers at the end of the section. As usual, I look at LSTM results first followed by Transformer Encoders.

LSTM Results. I limit LSTM results on Phone Classification to the constant 50%

probability rule for Stochastic Depth networks and the 50% local drop-path two-column FractalNet. These networks are selected as examples of successful pre-trained networks due to their superior generalization performance on the APC objective. Both 6 layer LSTM baselines are considered in order to provide a point of comparison between both methods. The line plot in Figure 7.1 shows PER across each of the 6 layers in these models.

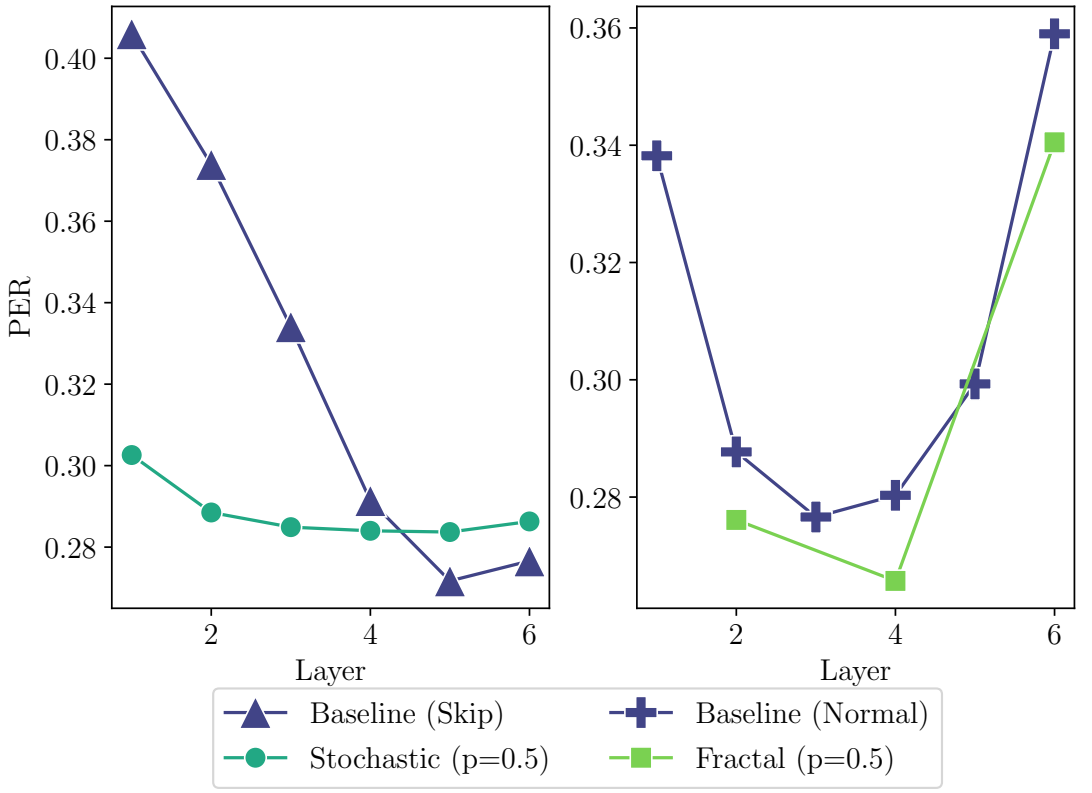


Figure 7.1: LSTM Phone Error Rate across different layers of original APC models. Fractal LSTM results are reported every two layers as each Fractal Block consists of 2 layers in its deepest column. Both techniques seem to approach baseline performance in this task with Fractal LSTMs managing to outperform both the normal and residual baselines. Stochastic Depth model features a flatter trend, probably as a result of more strongly independent sub-networks. Further discussion below.

The left plot shows a clear global trend throughout with an almost monotonic decrease in PER as the number of layers under consideration increases. We also see that the best performing layer/block is generally the penultimate one, consistent with the understanding that the final layer in LSTM networks is task-specific while earlier layers preserve more general information that could be useful to the PC objective [Hermans and Schrauwen, 2013].

Despite the global trend being conserved, we can also see some impact on the exact PER attained due to the proposed dynamic architectures. Most interestingly, the stochastic network seems to have a much 'flatter' curve suggesting that the architecture enabled all layers to be trained to a roughly equivalent extent. This supports the original

author's claims about Stochastic Depth promoting the independent learning of an ensemble of networks with varying depths. It is also worth noting that earlier layers greatly benefit from this approach, managing to beat their skip baseline PER by as much as 10%. This is likely because earlier layers capture more abstract and less readily useful information in regular LSTMs [Hermans and Schrauwen, 2013]. There is, however, one perceivable drawback of using the stochastic network as we are still losing a small amount of information in the network due to the approximation, resulting in a 1.2% increase in PER at the best performing layer. Proportionally, however, this represents a very small 4.42% difference which might not be of great concern depending on the application and could also be corrected with further epochs or even a more rigorous hyperparameter setting informed by a grid search. Altogether, the loss in performance here is comparable to that attained in regular APC pre-training and could therefore motivate Stochastic Depth approaches in SOTA LSTM-based self-supervised models.

Finally, fractal results are also consistent with expectations. Not only do we see the same macro-level trend as in the other two networks but we also get a significant reduction in PER for earlier layers in the deepest column. In fact, the best results on the Fractal LSTM outperform even the LSTM with residuals by a modest half a percentage (or a 2.21% reduction). The results are even more impressive when considering the regular baseline, where the Fractal LSTM outperforms by almost 4%. This observation is likely a result of the aforementioned implicit deep supervision speeding up the training dynamics of both columns in the Fractal architecture. Altogether, Fractal macro-architectures once again demonstrate their real potential to reduce pre-training times while not compromising much, or in this case even improving, accuracy over the baseline.

Transformer Results. Transformer results on PC consider the Stochastic Depth linear decay probabilty rule and the 50% local drop-path two-column FractalNet. These are once again selected as a result of their good generalization performance on APC. Due to the larger depth of these networks, layer-wise analysis here proceeds by considering every second layer instead of every first. This enables discussion about training performance throughout the model's depth while abstracting any potential (random or otherwise) variations across adjacent layers away. Figure 7.2 shows these results.

The trend here is less pronounced than in LSTM networks but nevertheless consistent throughout each model type. This time around, we see that the optimal layer is somewhere between the 6th and 8th layer instead of always being the penultimate one. This lines up with Pasad et al. [2021] who empirically demonstrate that the middle layers in the wav2vec 2.0 transformer contain the most phonetic information. Beyond this, these networks also perform generally worse when evaluated at their earlier layers.

Stochastic Transformers once again exhibit a considerably flatter trend than the baseline. Stochasticity also greatly helps to improve the performance of earlier layers in an analogous way to what we observed in LSTM models (up to 10% decrease in PER). Perhaps most notably, however, the performance loss in this instance appears to be more pronounced than in LSTM networks, with the best performing stochastic network exhibiting a 2.8% increase in PER over the best performing baseline network. While

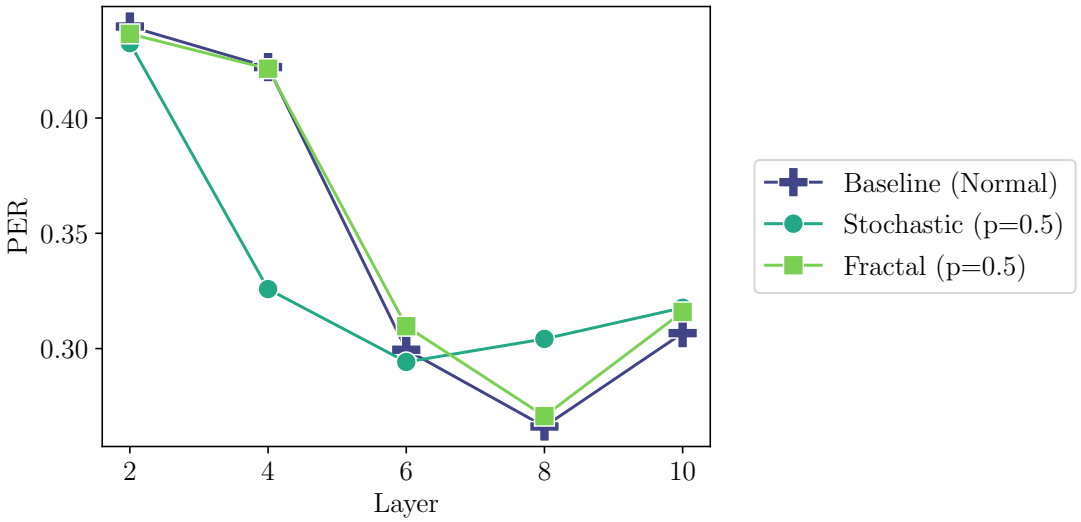


Figure 7.2: Transformer Phone Error Rate across different layers of original APC models. All results are reported every two layers. Fractal Transformer performance is basically identical to baseline 10 layer transformer while stochastic depth model loses some accuracy but features the same flatter trend.

other adjacent layers in the stochastic architecture (perhaps the 5 or 7 layer networks) might have slightly better performance, this could still be a sign that the linear decay rule being used is not optimal for transformer architectures as it is more likely to drop middle-late layers that are more feature-rich. An alternative assignment, such as a logarithmic rule might prove more appropriate and could help to minimise this performance loss. Additionally, hyperparameter tuning could again help given the method's sensitivity to this assignment.

Finally, Fractal Transformers also appear to perform similarly to Fractal LSTMs on this task, despite their previous sub-par generalization performance on the APC objective. This makes a strong case for the networks still learning most of the useful acoustic information necessary for downstream tasks. In fact, fractal models enable a slight improvement over the baseline in earlier layers and less compromised optimal PER relative to stochastic networks. The same global trend is also observed with the 8 layer transformer being the best performing model.

7.1.3 Conclusions

In this first section, I have empirically demonstrated that the training dynamics observed in the APC objective carry over to the Phone Classification downstream task. Namely, we observed that both LSTMs and Transformers preserve many of the general trends in downstream performance relative to their baselines and offer comparable best-case results. Furthermore, my results also illustrated other less obvious patterns, such as the improvements attained in earlier layers of Stochastic Depth networks and the higher accuracy that Fractal networks are consistently capable of achieving.

Model	Best Layer	eval92	dev93	Avg +/-
Baseline LSTM	3	27.66	28.97	0.00
Skip LSTM	5	27.17	28.37	0.00
Stochastic LSTM (p=0.5)	5	28.37	29.60	+1.22
Fractal LSTM (p=0.5)	4	26.57	27.62	**-1.22**
Baseline Transformer	8	26.62	28.42	0.00
Stochastic Transformer (p=DECAY)	6	29.42	31.04	+2.71
Fractal Transformer (p=0.5)	8	27.07	28.60	**+0.32**

Table 7.1: Best performing layer Phone Error Rates (%) on both Wall Street Journal test sets (eval92 and dev93). "Avg +/-" gives the mean absolute difference in PER attained by the model relative to its baseline across the two test sets. We can see that the differences in performance are generally quite low despite the significant pre-training time improvements. FractalNets also consistently provide less compromised results and even manage to outperform the LSTM baseline, while Stochastic Networks remain a valid alternative for both model types. Further discussion of the results can be found in the text above. All error rates are given to 2 decimal places.

Withal, results so far are generally optimistic and could further motivate the application of these techniques in larger SOTA models, especially in less performance-critical applications such as in consumer devices where the performance loss attained could have a negligible impact on the end user. In one final attempt to solidify the validity of these dynamic architectures in self-supervised speech domain contexts, I will now be investigating whether these results are mirrored in the speaker verification task.

7.2 Speaker Verification (SV)

In this section, I aim to measure how well speaker characteristics in the recordings are preserved by the hidden representations learned by the APC networks. This is typically done by attempting to map each utterance in the dataset to its source speaker and verifying the system's accuracy on this task. Unlike with the phonetic information being tested in the previous task, however, it is harder to define what these speaker characteristics are exactly. Many aspects of a recording could be considered speaker-dependent such as specific formant frequencies, spoken dialects and even prosodic features like the rhythm and intonation of the production. As such, I once again follow the same approach as Yang et al. in which I avoid trying to predict these less well-defined, overlapping individual features in favour of a more holistic verification objective. Success in this task would suggest that the representations learned contain sufficient information to distinguish the speakers in a dataset which could be useful in other common classification problems such as speaker diarization [Fujita et al., 2019].

7.2.1 Methodology

Speaker Verification experiments use the VoxCeleb1 dataset (Vox1) which contains utterances from 1,251 different speakers [Nagrani et al., 2017]. Summary statistics on the specific verification split chosen are given in Appendix A (Table A.2). Since this objective requires some form of per-utterance representation, I choose to follow the common approach proposed in Shon et al. [2018] of averaging all the frames in the hidden representations obtained from the APC network. As such, it is worth noting that my procedure for speaker verification requires no additional training as it tests the APC representations directly.

The verification procedure then consists of computing embeddings for pairs of utterances and determining whether the recordings belong to the same speaker or not. For this comparison, I compute the cosine similarity between the two aggregated utterance vectors. Furthermore, instead of computing a simpler accuracy metric, I instead opt to calculate the Equal Error Rate (EER) obtained, which is more commonly seen in the literature. This computation is done by sweeping different classification thresholds and computing the True Positive Rate (TPR) and the False Positive Rate (FPR) obtained at different points. TPR represents the rate at which the system correctly matches recordings from the same speaker, whereas FPR determines the rate at which it rejects recordings from the same speaker. The threshold that maximizes the difference $TPR - FPR$ is then reported as the EER of the system. For interpretation purposes, a lower EER indicates better performance.

7.2.2 Layer-Wise Analysis

As before, I consider each layer's performance on the verification task in order to determine and interpret any potential information loss throughout the architecture. Moreover, I again relate the results observed back to the current theoretical understanding for each model. For each network type, I consider the best performing hyperparameter settings found on the APC objectives in Chapters 5 and 6. Finally, I summarise the best performing layer for all networks considered in Table 7.2.

LSTM Results. The LSTM networks selected once again consist of the Stochastic Depth model with the 50% constant survival probability rule, the two-column FractalNet with 50% local drop-path and the regular and skip baselines. Figure 7.3 shows the trend in EER across each of the layers in these models. Note that FractalNet architectures are once again evaluated per-block (i.e. every 2 layers).

Unlike with phone classification, the global trend in this case is considerably less pronounced, but instead shows a decrease in EER as the number of layers in consideration increase. The decrease in many of these models however is not monotonic and often varies with the architecture. Generally, it would seem that models with residual connections benefit from extracting speaker representations from later layers while earlier layers appear to perform better otherwise.

Stochastic LSTM networks once again show a flatter trend, suggesting that each of these sub-networks is independently strong on the objective. The same benefit seen in

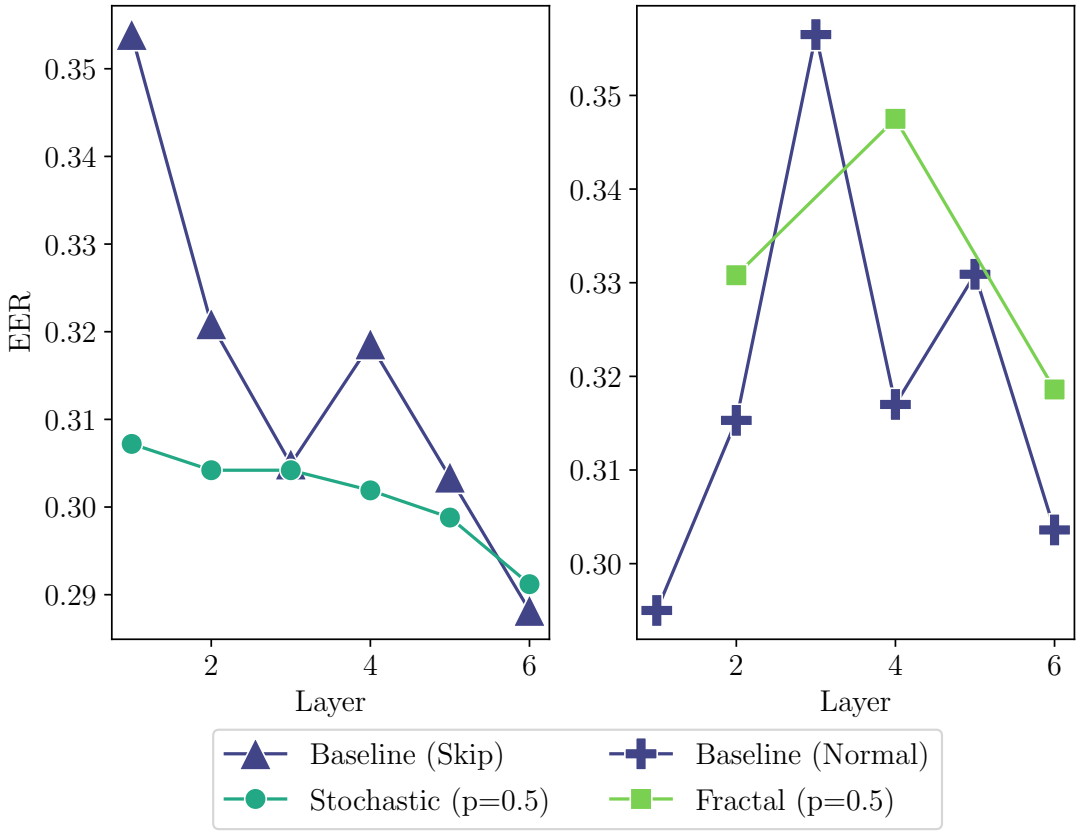


Figure 7.3: LSTM Equal Error Rate across different layers of original APC models. Fractal LSTM results are reported every two layers as each Fractal Block consists of 2 layers in its deepest column. Stochastic Depth networks exhibit the same flatter trend as before and Fractal macro-architectures appear to hurt EER considerably more than in the previous task. This is likely due to residual connections being very beneficial to this task as can be seen by the irregular performance of the normal LSTM baseline. Further discussion below.

PC where earlier layer performance is greatly improved by this approach is also seen here, with a decrease of up to 5% in the first layer. A loss in performance between the best performing models is also once again visible but it is worth noting how this is almost negligible compared to the difference seen in phone classification, constituting a very small 0.3% increase. The reason for this disparity could be due to the original hypothesis (H3) about speaker-specific information being much more abundant a priori due to how varied it can be. Overall, stochastic depth approaches seem to provide almost identical results (if not much better for smaller models) to their baseline, suggesting once again that the necessary speaker-dependent information are replicated by this alternative training procedure.

Fractal LSTMs on the other hand provide a more surprising result. Not only do these models arguably underperform on this objective relative to their baseline but the global trend is also less clear. In fact, the best performing network in this case is the full 6-layer column in the fractal architecture, which doesn't manage to achieve better

results than the stochastic counterparts. I suspect that the reason we are not observing the same trend here as we saw for PC is that the residual connections are significantly helping to lower the EERs obtained by directly injecting unperturbed signal features into later layers. These identity signals are likely to have less noise which makes their aggregate embeddings more useful to the verification task. However, this observation only represents an advantage of residual connections in pre-trained networks being used for this task and does not invalidate the use of Fractal macro-architectures altogether. Indeed, we can observe that the Fractal LSTM provides a better approximation to regular baseline LSTM across all layers being considered suggesting that the effect of the approach is still comparable to what was discussed in PC. Altogether, Fractal LSTMs still appear to learn much of the same information as regular LSTM of the same depth with a moderate decrease in performance.

Transformer Results. Once again, the Transformer networks analysed in this section are the Stochastic Depth 10-layer model with the 50% linear decay probability rule, the five-block, two-column FractalNet architecture with 50% local drop-path and the baseline 10-layer transformer. EER results across these models are presented in Figure 7.4.

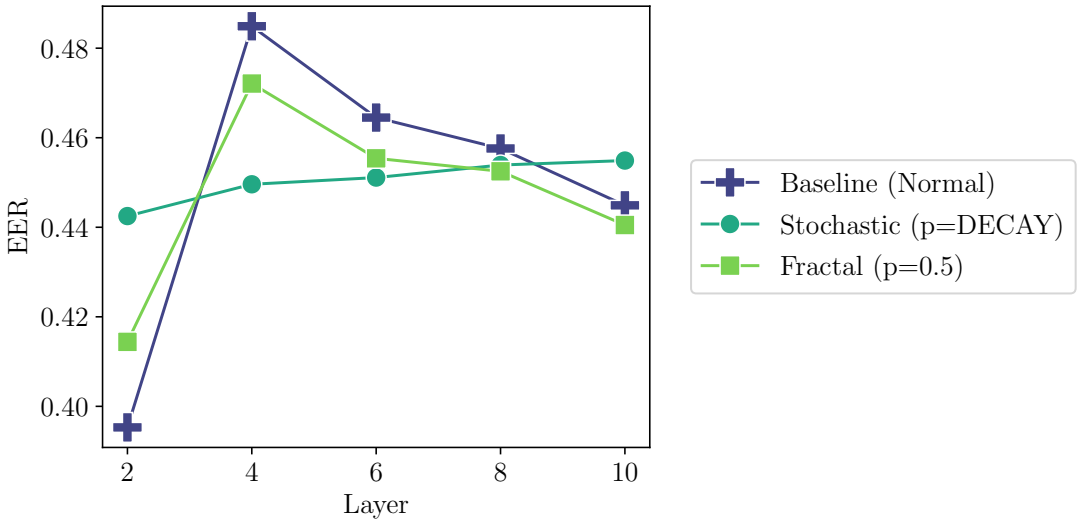


Figure 7.4: Transformer Equal Error Rate across different layers of original APC models. All models are considered every two layers. Stochastic Depth networks have a flatter trend as before. Fractal Transformers are also not negatively affecting performance, supporting my previous hypothesis about residual connections.

The global trend in transformer networks is clearer here than in the LSTM experiments. The earliest layers seem to perform the best in these models. The baseline and fractal trends both see a large spike in EER in middle layers followed by a monotonic improvement after this point. The reason why earlier layers seem to perform better in this case could once again be due to ”close-to-surface” form representations having the most accessible speaker-specific information. This would again support the LSTM observations and the benefit observed with residual connections as discussed above.

In terms of specific networks, Stochastic Depth Transformers exhibit the familiar flat trend again, but the former improvement to earlier layers is not as clear in this case. In fact, the best performing Stochastic Depth network in this objective causes a significant 4.72% increase in EER, which is the largest performance loss observed across both downstream objectives. Therefore, the narrative here is almost the opposite of what we observed with LSTMs, as Stochastic Depth networks seem to moderately hurt the accessibility to speaker-specific information in the networks. This difference could be due to random chance if the earlier layers in the model were dropped more frequently during training as they seem to be consistently the best performing. This is supported by the fact that the second layer model in the baseline transformer presents itself as an outlier almost, with every other EER across layers being comparable to its stochastic counterpart.

Fractal Transformer results are also at odds with Fractal LSTM's and are instead more consistent with fractal architectures in PC. In other words, they again provide a significantly lower compromise in performance throughout (less than 2% increase for best performing model) which could be explained by the deep implicit supervision speeding up training relative to stochastic models. It is also worth noting that since transformer networks have residual connections by default, the same disparity between Fractal and Stochastic networks is not observed here, which supports my hypothesis about residual connections being the defining difference in LSTM networks.

Model	Best Layer	EER (%)	+/-
Baseline LSTM	1	29.50	0.00
Skip LSTM	6	28.82	0.00
Stochastic LSTM (p=0.5)	6	29.12	**+0.30**
Fractal LSTM (p=0.5)	6	31.86	+2.36
Baseline Transformer	2	39.53	0.00
Stochastic Transformer (p=DECAY)	2	44.25	+4.72
Fractal Transformer (p=0.5)	2	41.44	**+1.91**

Table 7.2: Best performing layer Equal Error Rates (%) on VoxCeleb1 Speaker Verification dataset. "+/-" indicates the absolute percentage difference in EER attained by the model relative to its baseline. Once again, differences in performance are relatively low although results are generally superior in models with residual connections. This is likely due to the direct averaging method being used to extract speaker embeddings in this task. Further discussion can be found in the text above. All error rates are give to 2 decimal places.

7.2.3 Conclusions

We once again see many of the same trends when probing APC networks on the speaker verification task as we have elsewhere. Namely, Stochastic approaches manage to train all layers in the network to a similar degree and FractalNet architectures present a much reduced compromise in performance over the baseline. The one notable difference in

this case is that networks that only include earlier layers consistently perform better than deeper alternatives, suggesting that the task benefits from as close to surface-form representations as possible, particularly in transformer networks. This is likely a by-product of the feature extraction being used whereby utterance embeddings are computed by directly averaging hidden representations across all frames without any further processing. This observation is also consistent with Pasad et al. [2021] who show that acoustic information is most similar to hidden representations in earlier transformer layers of wav2vec 2.0.

A potential improvement to this methodology would be to consider a less crude feature extraction approach. For example, Yang et al. proposed and demonstrated that learning a latent 'speaker' space by training an additional linear layer on top of the network greatly improves the EER results obtained. This could in turn change the trends observed and consequently the best performing layers.

Contrary to my initial hypothesis (H3), however, it is not very clear whether the proposed approach has a lesser impact on this speaker verification task than in PC, particularly due to the aforementioned limitations. Further investigations would therefore need to be carried out to ascertain whether this is indeed the case or not. Despite this, my results still demonstrate that the dynamic architectures generally have a very minor (sometimes negligible) impact on the downstream performance of the APC networks on this task. Although the training procedure is greatly changed, the information diversity, quantity and accessibility seems to remain largely the same across each model type, making a strong case for their use in the self-supervised speech domain.

7.3 Summary

In this chapter, I empirically showed that the proposed pre-training strategies indeed preserve much of the information learned by baseline networks, making them practically useful in the self-supervised paradigm. Although I only considered these two sample downstream tasks, I expect we would obtain similar results on derivative tasks (such as automatic speech recognition or speaker identification) and other more complex tasks where APC has been shown to succeed in, such as speech translation [Chung and Glass, 2020]. Similarly, my results also illustrate how the proportional performance loss we observe with these dynamic architectures is often minimal. I would expect this to continue being the case even in pre-training SOTA models, which again provides an optimistic outlook on their potential in practical applications.

These findings could hence motivate the use of these dynamic architectures in industrial settings, particularly those where a marginal decrease in performance is not critical to success, such as in commercial systems.

Chapter 8

Conclusions

In this project, I investigate how dynamic architectures like Stochastic Depth and Fractal networks can be used in order to reduce the pre-training time requirements of foundation models in self-supervised speech representation learning. My experiments demonstrate, at depth, the impact these techniques have both in the pre-training and downstream objectives where APC is used as a representative pre-training loss. In particular, I have made the following contributions.

1. I have demonstrated that both Stochastic Depth and FractalNet architectures can be adapted successfully to the speech domain in order to significantly reduce pre-training times over their respective baselines while only incurring a moderate compromise in accuracy. This has been demonstrated on both LSTMs and Transformer networks, which represent two of the most popular speech architectures and are featured to some degree in many SOTA models like wav2vec 2.0.

2. I have demonstrated that fractal macro-architectures manage to consistently provide a lesser compromise in performance over their Stochastic Depth counterparts in both criteria (pre-training and downstream task performance). This is consistent with findings from the original authors in deep residual convolutional neural networks.

3. I have empirically shown that Stochastic Depth performance remains sensitive to the choice of hyperparameters in this new domain, while fractal macro-architectures are more robust to the specific drop-path strategy being used. This informs the need for practitioners looking to use Stochastic Depth techniques to consider the layer dynamics of their model type and carry out careful hyperparameter tuning in order to attain the best results. I also show that increasing Fractal Block width could be a practical way of further reducing FractalNet pre-training time requirements without compromising much performance.

4. I have empirically demonstrated that the information quantity, diversity and availability in the pre-trained networks with these dynamic architectures remains largely unchanged, making these techniques practically useful in the self-supervised speech representation learning paradigm. Specifically, I have shown that sequential phonetic information as well as speaker-dependent features are

present in these architectures, which motivates their use for many common speech downstream tasks such as automatic speech recognition and speaker diarization. Finally, although my experiments focused on probing to illustrate this point, these results also suggest that fine-tuning performance is likely to also remain largely unaffected by these dynamic architectures.

8.1 Limitations

Due to limitations in time and resources, there have been some aspects of my methodology that could be revised in order to provide a more thorough analysis.

1. Inference and backpropagation times reported are likely to be somewhat inconsistent due to the GPU cluster used having some unpredictable latency due to memory and network overheads. Although I tried to control for most of this variance by running all experiments using machines with the same single GPU, I expect that this puts into question the validity of some of the absolute timing values reported in this project. Nonetheless, the relative differences observed between models are likely to remain largely the same in more consistent and faster hardware, validating my analysis and results.
2. Speaker Verification results could likely benefit from a less crude feature extraction method such as training an additional linear layer to project the APC features onto a latent space. This eliminates any confounding factors and has the potential to both improve results across models and possibly make some of the trends discussed clearer.

8.2 Future Work

Finally, this work presents a few clear avenues of future research:

1. A simple extension to this work would be to explore how these methods translate to other architectures and self-supervised speech losses (e.g. Contrastive Predictive Coding, wav2vec 2.0 contrastive loss, etc.[Oord et al., 2018, Baevski et al., 2020]).
2. Fractal-specific extensions could consider a more thorough search of the best possible sub-network that we could extract from the Fractal architecture for different speech tasks. Maybe the deepest column is not the best candidate for certain downstream objectives and shallower columns, or even hybrid networks consisting of a combination of different columns might be better suited.
3. Finally, a more theoretical avenue would be to attempt to quantify the exact information that is being conserved/lost in the networks due to the use of these dynamic architectures. This could be done through dimensionality reduction methods such as t-SNE and PCA [Maaten and Hinton, 2008, Maćkiewicz and Ratajczak, 1993].

Bibliography

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. 2020. doi: 10.48550/ARXIV.2006.11477. URL https://arxiv.org/abs/2006.11477.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL https://aclanthology.org/2022.acl-short.1.

Yu-An Chung and James Glass. Generative pre-training for speech with autoregressive predictive coding. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3497–3501. IEEE, 2020.

Diego De Benito-Gorron, Alicia Lozano-Diez, Doroteo T. Toledano, and Joaquin Gonzalez-Rodriguez. Exploring convolutional, recurrent, and hybrid deep neural networks for speech and music detection in a large audio dataset. *EURASIP Journal on Audio, Speech, and Music Processing*, 2019(1):9, December 2019. ISSN 1687-4722. doi: 10.1186/s13636-019-0152-1.

R Fong and A Vedaldi. Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018)*. Institute of Electrical and Electronics Engineers, 2018.

Yusuke Fujita, Naoyuki Kanda, Shota Horiguchi, Yawen Xue, Kenji Nagamatsu, and Shinji Watanabe. End-to-End Neural Speaker Diarization with Self-Attention. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 296–303, 2019. doi: 10.1109/ASRU46091.2019.9003959.

Kunihiko Fukushima. Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4): 322–333, 1969. doi: 10.1109/TSSC.1969.300225.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Michiel Hermans and Benjamin Schrauwen. Training and Analysing Deep Recurrent

Neural Networks. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006. doi: 10.1126/science.1127647. URL https://www.science.org/doi/abs/10.1126/science.1127647.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. 2015. doi: 10.48550/ARXIV.1503.02531. URL https://arxiv.org/abs/1503.02531.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. 2012. doi: 10.48550/ARXIV.1207.0580. URL https://arxiv.org/abs/1207.0580.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

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 and Lang. Proc.*, 29:3451–3460, October 2021. ISSN 2329-9290. doi: 10.1109/TASLP.2021.3122291. URL https://doi.org/10.1109/TASLP.2021.3122291.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep Networks with Stochastic Depth. In *Computer Vision – ECCV 2016*, volume 9908, pages 646–661. Springer International Publishing, Cham, 2016. ISBN 978-3-319-46492-3 978-3-319-46493-0. doi: 10.1007/978-3-319-46493-$0_3$9.

F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532–556, 1976. doi: 10.1109/PROC.1976.10159.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. 2020. doi: 10.48550/ARXIV.2001.04451. URL https://arxiv.org/abs/2001.04451.

Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions. 2019. doi: 10.48550/ARXIV.1910.10261. URL https://arxiv.org/abs/1910.10261.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. FractalNet: Ultra-Deep Neural Networks without Residuals. 2016. doi: 10.48550/ARXIV.1605.07648. URL https://arxiv.org/abs/1605.07648.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 562–570, San Diego, California, USA, May 2015. PMLR. URL https://proceedings.mlr.press/v38/lee15a.html.

Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243.

Jianing Li and Vardan Papyan. Residual Alignment: Uncovering the Mechanisms of Residual Networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Luis Lugo and Valentin Vielzeuf. Efficiency-oriented approaches for self-supervised speech representation learning. 2023. doi: 10.48550/ARXIV.2312.11142. URL https://arxiv.org/abs/2312.11142.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (PCA). *Computers & Geosciences*, 19(3):303–342, 1993. ISSN 0098-3004. doi: https://doi.org/10.1016/0098-3004(93)90090-R. URL https://www.sciencedirect.com/science/article/pii/009830049390090R.

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. 2022. doi: 10.48550/ARXIV.2205.10643. URL https://arxiv.org/abs/2205.10643.

Michael C. Mozer. A Focused Backpropagation Algorithm for Temporal Pattern Recognition. *Complex Syst.*, 3, 1989. URL https://api.semanticscholar.org/CorpusID:18036435.

A. Nagrani, J. S. Chung, and A. Zisserman. VoxCeleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. 2018. doi: 10.48550/ARXIV.1807.03748. URL https://arxiv.org/abs/1807.03748.

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 *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 914–921. IEEE, 2021.

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.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, and others. The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

Mirco Ravanelli, Philemon Brakel, Maurizio Omologo, and Yoshua Bengio. Light Gated Recurrent Units for Speech Recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):92–102, 2018. doi: 10.1109/TETCI.2017.2762739.

Suwon Shon, Hao Tang, and James Glass. Frame-level speaker embeddings for text-independent speaker recognition and analysis of end-to-end model. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1007–1013. IEEE, 2018.

S. S. Stevens, J. Volkmann, and E. B. Newman. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, 8(3): 185–190, January 1937. ISSN 0001-4966, 1520-8524. doi: 10.1121/1.1915893.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. Lite Transformer with Long-Short Range Attention. 2020. doi: 10.48550/ARXIV.2004.11886. URL https://arxiv.org/abs/2004.11886.

Gene-Ping Yang, Sung-Lin Yeh, Yu-An Chung, James Glass, and Hao Tang. Autoregressive Predictive Coding: A Comprehensive Study. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1380–1390, October 2022. ISSN 1932-4553, 1941-0484. doi: 10.1109/JSTSP.2022.3203608. URL https://ieeexplore.ieee.org/document/9874771/.

Appendix A

Downstream Dataset Statistics

Dataset	Time	Speakers	Utterances
Training (`si284`)	80 h	283	37,416
`eval92`	42 min	8	333
`dev93`	1 h 5 min	10	503

Table A.1: Summary statistics on training and testing subsets from the Wall Street Journal speech recognition dataset.

Dataset	Speakers	Distinct Utterances	Utterance Pairs
VoxCeleb1 (cleaned)	40	4,715	37,720

Table A.2: VoxCeleb1 Verification Split Summary Statistics.