

# AN ANALYSIS OF LINEAR COMPLEXITY ATTENTION SUBSTITUTES WITH BEST-RQ

Ryan Whetten<sup>1</sup>, Titouan Parcollet<sup>2</sup>, Adel Moumen<sup>1</sup>, Marco Dinarelli<sup>3</sup>, Yannick Estève<sup>1</sup>

<sup>1</sup> Laboratoire Informatique d’Avignon, Avignon Université, France

<sup>2</sup> Samsung AI Center Cambridge, United Kingdom

<sup>3</sup> Univervisté Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000, Grenoble, France

## ABSTRACT

Self-Supervised Learning (SSL) has proven to be effective in various domains, including speech processing. However, SSL is computationally and memory expensive. This is in part due the quadratic complexity of multi-head self-attention (MHSA). Alternatives for MHSA have been proposed and used in the speech domain, but have yet to be investigated properly in an SSL setting. In this work, we study the effects of replacing MHSA with recent state-of-the-art alternatives that have linear complexity, namely, HyperMixing, Fastformer, SummaryMixing, and Mamba. We evaluate these methods by looking at the speed, the amount of VRAM consumed, and the performance on the SSL MP3S benchmark. Results show that these linear alternatives maintain competitive performance compared to MHSA while, on average, decreasing VRAM consumption by around 20% to 60% and increasing speed from 7% to 65% for input sequences ranging from 20 to 80 seconds.

**Index Terms**— self-supervised learning, speech, efficiency, linear complexity

## 1. INTRODUCTION

Self-supervised learning (SSL) is an approach to training machine learning models where pseudo-targets are extracted from the data itself. Since SSL is unsupervised, these models can be pre-trained on immense amounts of unlabeled data, and then obtain good results on downstream tasks using minimal amounts of labeled data. SSL methods have proven to be useful in a variety of domains including in speech processing [1], where SSL has reached state-of-the-art performance in tasks like Automatic Speech Recognition (ASR) [2], Emotion Recognition (ER) [2], Automatic Speaker Verification (ASV) [2], Spoken Language Understanding (SLU) [2], and Automatic Speech Translation (AST) [3, 4].

Despite their performance, training SSL models is still very costly in terms of the amount of data, GPUs, and time

needed. For example, Google USM was trained on 12 million hours (or over 1,369 years) of audio [5], and the base and large XLS-R models were trained with 128 and 200 GPUs, respectively [4]. Even in efforts to make state-of-the-art models like HuBERT and data2vec more efficient, these SSL models still require around 1,000 A100 GPU hours of training [6, 7].

In a study of the architectures of SSL models for speech [8], the authors identified three main culprits: (i) the Acoustic Feature Extractor (AFE), which transforms the raw waveform into a latent representation; (ii) the context encoder, which is often a large Transformer [9] or Conformer [10]; and (iii) the SSL training objective.

When looking at these three culprits, BEST-RQ [11] seems to be theoretically one of the most efficient SSL models for speech that has been proposed. BEST-RQ starts with Mel Filterbanks, addressing culprit (i). Then, it creates pseudo labels using a frozen, randomly initialized linear projection and codebook coupled with cross-entropy training, addressing culprit (iii). In contrast, other models, such as wav2vec 2.0 [12], use a learnable codebook and typically use a combination of objectives, slowing down training. For instance, previous studies showed that BEST-RQ obtain comparable downstream performance to wav2vec 2.0 while being 2.4 times faster to train [13]. However, for culprit (ii), the context encoder, BEST-RQ uses Conformer layers, which are computationally expensive.

Conformers, like transformers, are expensive partly due to multi-head self-attention (MHSA) time complexity being quadratic with respect to the input sequence length. Therefore, to address culprit (ii), one needs to search for an efficient alternative to MHSA. Many studies have been conducted to reduce the complexity of MHSA, but only few have been applied to speech tasks and none have been applied to SSL for speech. Examples of such methods include HyperMixing [14], Fastformer [15], SummaryMixing [16], and Mamba [17].

The main contribution of this work is to address culprit (ii) by, for the first time, evaluating the most promising linear time complexity alternatives to MHSA in an SSL for speech setting. Downstream experiments conducted following the MP3S benchmark [18] show that our linear-time complexity BEST-RQ maintains performance with an equiva-

This work received funding from the French ANR E-SSL project (N°ANR-22-CE23-0013) and used HPC resources from GENCI-IDRIS (AD011014732 and A0131013821)

lent MHSA BEST-RQ while decreasing VRAM consumption by around 20% to 60% and increasing inference speed from 7% to 65% for input sequences from 20 to 80 seconds. As a second contribution, we open-source the code in the widely used SpeechBrain toolkit [19], enabling the community to experiment with efficient SSL models for speech<sup>1</sup>.

## 2. BACKGROUND

Multi-head self-attention (MHSA) has quadratic time complexity with respect to the input length due attention weights being calculated by a dot product between every query-key token pair. Despite its complexity, this dot product operation enables each token to access the global context which is important for reaching high performance [9].

In research on reducing the complexity of MHSA, there are two overarching methods: (i) those aiming to approximate this pair-wise token computation with a lower cost, and (ii) those that do not seek to mimic this pair-wise token computation, but instead introduce global context in another fashion.

Some methods in the first family include sparse attention, such as BigBird and Longformer [20, 21], which use a combination of sliding window, global, and random attention to achieve linear complexity. The main issue with these sparse attention methods is that they cannot fully model global context as they operate on windows. Other methods in this family, such as Linformer and Linear Transformer [22, 23] approximate attention by computing low-rank approximates of the key and value matrices or use the dot-product of kernel feature maps and make use of the associative property to achieve linear complexity. Yet, in practice, the aforementioned methods are still computationally expensive [15].

In contrast, Fastformer [15] has proven to perform well on text data, outperforming previously mentioned sparse attention methods and low-rank approximates. This is done by making use of additive attention and element-wise multiplication to summarize the query and key matrices as vectors resulting in linear complexity while fully modeling global context. Fastformer has also been applied to speech and proved to work well on ASR tasks with the branchformer architecture [24]. Due to these performances and the availability of its implementation, Fastformer is considered as a good representative candidate of the first family of methods.

The other family of solutions, which do not aim to approximate the self-attention mechanism, include, HyperMixing [25], SummaryMixing [16], and Mamba [17]. HyperMixing was first introduced with the HyperMixer [25]. The HyperMixer is an extension of the MLPMixer [26]. The disadvantage of MLPMixer is that it can not handle inputs of varying length, making it not suitable for NLP or speech tasks. HyperMixer extends MLPMixer by using hypernetworks [27] to capture global context for inputs that vary in length. Hy-

perMixer proved to have good performance on text tasks and fully-supervised ASR with the HyperConformer [25, 14].

Alternatively, SummaryMixing does not use hypernetworks, but instead the input is passed through a parametrized function, such as a multi layer perceptron, which is averaged across all time steps, resulting in a summary vector. To introduce global context to the input sequence, this summary vector is concatenated to the input at each time step and then fed through another parametrized function, which becomes the final output. SummaryMixing proved to perform just as well as MHSA on ASR, keyword spotting, and SLU, while decreasing the amount of required memory and training time.

Lastly, Mamba is a method for sequence modeling based on state space models [17]. State space models can be thought of as a combination of recurrent neural networks and convolutional neural networks, which scale linearly with respect to the sequence length. Mamba has shown good performance on text, audio (which were in an auto regressive SSL setting), and a two supervised speech tasks [28]: ASR and speech enhancement. HyperMixing, SummaryMixing, and Mamba all have available implementations, and are therefore selected as representatives of the second family of methods.

Despite the good performance of these substitutes for MHSA on supervised tasks, their performance in an SSL setting for speech tasks has not yet been studied, and the focus of these studies has been mostly on ASR. Furthermore, these methods have not been scaled to large models, e.g. from only about 5 to 100M parameters for speech tasks [14, 16, 24, 28], which contrasts with the current scale of SSL models. Thus, in this work we explore these unexamined aspects of linear complexity alternatives by (i) experimenting in an SSL setting, (ii) looking at performance on a variety of speech tasks using a benchmark from the community, and (iii) scaling up model size to above 300M parameters.

## 3. ATTENTION ALTERNATIVES IN BEST-RQ

In this section, we give an overview of BEST-RQ and describe how Fastformer, SummaryMixing, HyperMixing, and Mamba achieve linear complexity. Let us define the input to the self-attention module as  $\mathbf{X} \in \mathbb{R}^{T \times d} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T]$  or a sequence of vectors of dimension  $d$  with  $T$  time steps. The difference between methods lies in how this input is transformed to contain information about the global context.

**BEST-RQ** is, to this day, the most efficient SSL paradigm. It is notably used by large scale models such as Google USM [5] and Universal-1 from AssemblyAI. BEST-RQ begins with Mel Filterbanks which are passed through two paths, (i) the model, and (ii) the random-projection quantizer. For path (i), a random portion of the Mel Filterbanks are masked and then passed through two convolutional layers, a series of conformer layers, and then a linear layer. For path (ii) the unmasked Mel Filterbanks are passed through a ran-

<sup>1</sup><https://github.com/whettenr/brq-att-alt-exp>

domly initialized frozen linear layer. Then a codebook look up is performed following:

$$y = \arg \min_i \| \text{norm}_{l_2}(c_i) - \text{norm}_{l_2}(Am) \|, \quad (1)$$

where  $m$  is a stacked portion of four Mel Filterbank frames,  $A$  is the linear projection, and  $c$  is the codebook. The index,  $y$ , from this codebook look up is used as the pseudo-target for pre-training for that portion of the input. Then, the cross-entropy loss is calculated between the masked sections of the output of the linear layer from the path (i) and the corresponding pseudo-targets from path (ii). BEST-RQ originally uses self-attention and, therefore, exhibits quadratic time complexity.

**Fastformer.** The complexity is reduced to linear by using additive attention to summarize the attention matrices as a single vector. To show how this is done, let  $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{T \times d}$ , be the standard linear transformations from the transformer composed of  $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_T]$  and  $\mathbf{K} = [\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_T]$ .  $\mathbf{Q}$  is summarized as a query vector  $\mathbf{q}$  by the following:

$$\alpha_t = \text{softmax}\left(\frac{\mathbf{w}_q^T \mathbf{q}_t}{\sqrt{d}}\right); \quad \mathbf{q} = \sum_{t=1}^T \alpha_t \mathbf{q}_t. \quad (2)$$

where  $\mathbf{w}_q \in \mathbb{R}^d$  is a learnable vector used with each vector of  $\mathbf{Q}$  to generate the attention score  $\alpha_t$ . The summary query vector  $\mathbf{q}$  is calculated as a weighted sum of the attention scores with their respective  $\mathbf{q}_t$ . Then,  $\mathbf{q}$  is multiplied by each vector in  $\mathbf{K}$ , resulting in a global context-aware key matrix that relies on the more efficient element-wise multiplication instead of dot products.

**HyperMixing** reduces the complexity by using a token mixing multi-layered perceptron (TM-MLP) from the MLP-Mixer [16], which can be described as:

$$\text{TM-MLP}(\mathbf{X}) = \text{LayerNorm}(\mathbf{W}_1(\sigma(\mathbf{W}_2^T \mathbf{X}^T))), \quad (3)$$

where  $W_1, W_2$  are weight matrices and  $\sigma$  represents some non-linear activation function. This is a standard MLP except the linear operations are performed on the transposed  $\mathbf{X}$ . This can be thought of as operating on the tokens instead of the channels which introduces global context as it allows information to pass between tokens. The key difference between MLP-Mixer and HyperMixing is that  $\mathbf{W}_1$  and  $\mathbf{W}_2$  are generated by another MLP and thus can vary in length.

**SummaryMixing.** The input,  $\mathbf{X}$ , is passed into two functions  $f$  and  $s$  taking the form, in practice, of two MLPs. The output of  $s$  is averaged across all time steps, resulting in a summary vector  $\bar{s}$ . To introduce global context to the input sequence,  $\bar{s}$  is concatenated to the output of  $f$  at each time step,  $t$ , and

then fed through another function, or MLP in this case, called  $c$ . The output of  $c$  becomes the final output  $\mathbf{h}$ . The overall SummaryMixing mechanism can be expressed as:

$$\bar{s} = \frac{1}{T} \sum_{t=1}^T s(\mathbf{x}_t); \quad \mathbf{h} = c(f(\mathbf{X}), \bar{s}). \quad (4)$$

Calculating the summary vector  $\bar{s}$  is an average and thus is linear with respect to the input length. As a result, SummaryMixing is able to introduce global context with linear complexity.

**Mamba.** The memory complexity is reduced to linear by processing the input sequence in an unidirectional manner similar to a recurrent network. Being a selective state space model, Mamba represents the past information of  $\mathbf{X}$  as a hidden state  $\mathbf{h}_t$  with constant memory consumption, hence solving the problematic quadratic time complexity from MHSA. To do so, the model employs a discretized state transition matrix  $\bar{A} \in \mathbb{R}^{N \times N}$ , an input discretized projection matrix  $\bar{B} \in \mathbb{R}^{H \times 1}$ , and an output projection matrix  $C \in \mathbb{R}^{1 \times H}$  to compute the hidden state  $\mathbf{h}_t$  as follow:

$$\mathbf{h}_t = \bar{A}\mathbf{h}_{t-1} + \bar{B}\mathbf{X}_t, \quad y_t = C\mathbf{h}_t. \quad (5)$$

To reduce the computation overhead brought by the recurrence, an hardware-aware parallel scan algorithm introduced in [17] allows to unroll equation 5 as the input sequence  $\mathbf{X}$  convolved with a structured kernel composed of  $\bar{A}, \bar{B}$ , and  $C$  fixed. With this improvement, the throughput of Mamba is five times higher than a Transformer. Finally, to introduce global context to the model, similar to MHSA, we use a bidirectional Mamba as motivated by [28].

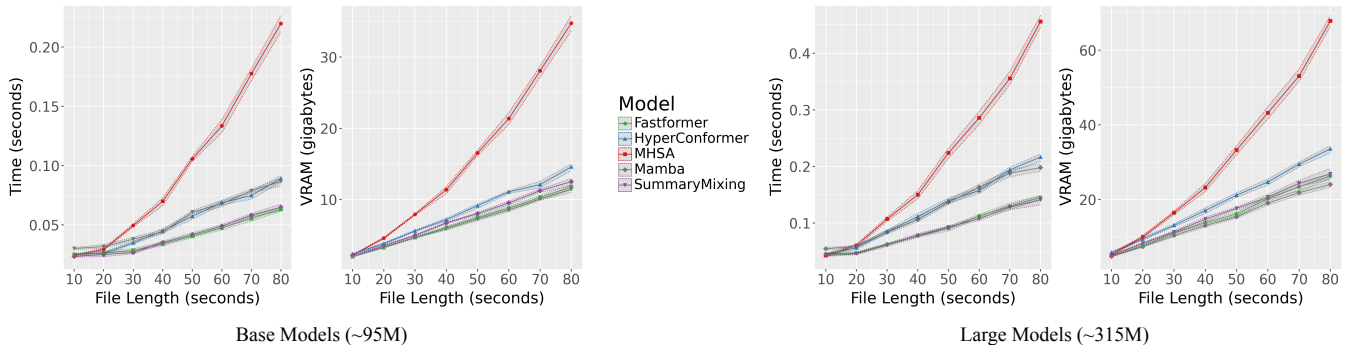
## 4. EXPERIMENTS

In this section, we describe our SSL pre-training protocol and summarize the downstream tasks, giving a brief overview of the datasets and evaluation metrics for each task.

### 4.1. Self-supervised Pre-training

All models are open-sourced in the SpeechBrain [19] library along with the hyperparameters for each model.

**Architecture details.** We use the implementation of BEST-RQ from [13] which uses Conformer layers [10] with relative sinusoidal positional embedding and the PyTorch multi-head self-attention. For all models the pre-training framework stays the same. We only replace the MHSA cell with either Hypermixing, Fastformer, SummaryMixing, or Mamba. The small models have 12 conformer layers and the large models have 24 layers. We adjust the hidden dimensions to make all the models have around the same number of parameters, that is, 95M and 315M for the small and large ones respectively. The detailed list of hyperparameters can be found in



**Fig. 1:** Inference speed and peak memory of BEST-RQ base and large models with various types of attention. Input length is increased from 10 to 80 seconds. Multi-head self-attention (MHSA) requires significantly more time and VRAM as input size increases where as the alternatives do not.

the open-sourced SpeechBrain recipes.

**Pre-training details.** We pre-train all our models for the same amount of steps, set to 200k using the 960 hours of training data from the LibriSpeech [29] dataset. Although 200k steps is a lower number than other state-of-the-art models trained for 400k or 800k steps [12, 6], previous research has empirically shown that, 200k steps is sufficient to compare SSL model performance [8, 13]. Dynamic batching is used, meaning audio files are grouped or bucketed together with those of a similar length, and the number of samples per batch varies based on the bucket size in order to keep the number of seconds of input per batch similar. In our experiments, and following the literature [12], we set the batch size to 1.7 hours for all models.

## 4.2. Downstream Tasks and Metrics

For the downstream evaluation we use a portion of the tasks from the *Multi-Probe Speech Self-Supervision* benchmark or MP3S [18]. In MP3S, pre-trained models are frozen and a learned weighted sum of the outputs of the hidden layers of the pre-trained models are fed into a downstream model called a *probe*. Because results were shown to vary depending on the probe’s architecture, for each task two probes are provided. Furthermore, and as SSL models are commonly fine-tuned with the downstream use case, we added an evaluation with a full fine-tuning for ASR.

**Automatic Speaker Verification (ASV)** is a binary classification task where given 2 utterances, the goal is to determine whether the speakers are the same. The evaluation metric for ASV is the Equal Error Rate (EER). For the dataset, we use VoxCeleb1 [30], which is made of utterances from celebrities sourced from YouTube. The dataset is divided into train and test splits, which we used accordingly. The probes for this task are the X-Vectors [31] and ECAPA-TDNN [32].

**Intent Classification (IC)** is a classification task of predicting the main purpose or objective of an utterance. We use accuracy as the evaluation metric. For this task, we use the SLURP dataset [33] containing 18 intents coming from single-turn user interaction with a voice assistant. Some examples of intents are *email*, *calendar*, or *play* (as in *play a song*). The probes for this task are a linear probe, in which the output of the SSL model are average-pooled along the time dimension and then passed into a linear classifier, and an LSTM probe, which consists of a two-layered BiLSTM followed by a linear classification layer.

**Emotion Recognition (ER)** is the task of predicting the emotion of a speaker. Similar to IC, we use accuracy to measure performance. We use the IEMOCAP dataset, which consists of 10 actors performing scripts with four different emotions (neutral, happy, sad and angry). The probes for this task are a linear probe, like IC, and an ECAPA-DNN probe, like ASV.

**Automatic Speech Recognition (ASR)** is the task of transcribing what was said in an utterance. For ASR, we measure performance by the word error rate (WER). We use the LibriSpeech *train-clean-100* for training, *dev-clean* for validation, and *test-other* for final testing. We report on the final test WER without a language model and with the official 4-gram language model<sup>2</sup> using beam search and shallow fusion. Also, and to test performance in low-resource and out-of-domain settings, which is one use-case of SSL models, we use the two low-resource language ASR datasets proposed in MP3S, Welsh and Basque from the CommonVoice 11.0 dataset [34]. For LibriSpeech, we use the 2-layered BiLSTM (LSTM) and ContextNet [35] probes, and for CommonVoice we use the linear and LSTM probes offered by MP3S. As mentioned, it is common for pre-trained models to be fine-tuned on a given task instead of being frozen. As a representative of this, we fine tune the models using the 100 hours of labeled data in

<sup>2</sup>[openslr.org/11/](https://openslr.org/11/)

**Table 1:** Results of various alternatives to MHSA for speech SSL models on the MP3S benchmark. Base models are set to have around 94M parameters and large models, denoted -LG, around 315M. Linear complexity alternatives perform competitively with multi-head self-attention (MHSA) even surpassing depending on the task and probe.

Model/Task	Specs	LibriSpeech train-100 ASR				CommonVoice		VoxCeleb	SLURP	IEMOCAP
Metric		WER ↓				WER ↓	WER ↓	EER ↓	Acc. ↑	Acc. ↑
1st Probe		LSTM				Lin.	Lin.	Xvectors	Lin.	Lin.
	# Params.	Clean	Clean LM	Other	Other LM	Welsh	Bask	ASV	IC	ER
MHSA	94.0M	<b>10.31</b>	<b>6.88</b>	<b>24.95</b>	<b>18.54</b>	80.04	82.56	9.36	50.3	60.1
HyperConf	94.3M	10.96	7.24	25.82	19.05	<b>79.65</b>	<b>81.41</b>	<b>8.54</b>	53.2	60.1
Fastformer	93.9M	13.21	8.59	33.11	24.24	82.03	84.62	12.27	36.2	56.4
SummaryMix	94.1M	11.92	7.76	28.25	20.80	79.97	81.99	11.13	52.1	63.0
Mamba	94.2M	10.46	6.90	26.00	18.88	80.71	83.66	11.91	<b>53.5</b>	<b>64.10</b>
MHSA-LG	313.5M	<b>7.10</b>	<b>4.98</b>	<b>17.63</b>	<b>12.92</b>	84.46	88.88	8.60	54.8	62.1
HC-LG	315.4M	7.73	5.47	18.39	13.83	77.66	79.94	8.64	62.3	62.8
Fastformer-LG	315.9M	18.74	11.83	41.07	30.55	82.85	84.59	10.13	34.5	56.9
SummaryMix-LG	313.7M	7.22	5.04	17.67	13.14	<b>76.94</b>	<b>78.88</b>	<b>8.30</b>	<b>62.8</b>	65.3
Mamba-LG	314.9M	7.84	5.23	20.57	14.66	80.86	84.66	10.54	57.7	<b>66.15</b>
2nd Probe		Contextnet				LSTM	LSTM	ECAPA	LSTM+Lin.	ECAPA
	Time(h) Mem(GB)	Clean	Clean LM	Other	Other LM	Welsh	Bask	ASV	IC	ER
MHSA	453 8.86	<b>10.10</b>	6.30	<b>22.84</b>	16.83	57.78	50.24	4.37	74.9	60.83
HyperConf	438 8.60	10.66	6.78	23.46	17.54	56.74	48.39	<b>3.65</b>	<b>76.9</b>	60.10
Fastformer	357 7.25	13.10	8.10	29.98	22.75	59.14	52.59	4.84	68.3	54.30
SummaryMix	368 7.68	11.55	7.21	25.13	18.72	58.24	50.05	4.09	76.0	<b>64.34</b>
Mamba	427 6.62	10.27	<b>5.96</b>	23.93	<b>16.37</b>	<b>55.22</b>	<b>47.51</b>	4.51	76.2	63.05
MHSA-LG	973 20.70	7.02	4.40	15.78	<b>11.82</b>	58.39	50.35	3.96	74.3	63.74
HyperConf-LG	1033 22.28	7.62	4.86	16.79	12.48	55.75	47.99	<b>3.45</b>	78.0	64.62
Fastformer-LG	780 18.76	18.41	10.97	38.64	28.92	61.18	54.49	4.26	68.6	55.44
SummaryMix-LG	805 19.49	<b>6.79</b>	<b>4.38</b>	<b>15.69</b>	11.83	58.24	48.59	4.05	<b>78.6</b>	<b>65.21</b>
Mamba-LG	1010 16.99	7.54	4.51	18.49	12.63	<b>55.42</b>	<b>47.59</b>	3.76	78.1	64.21

the *train-clean-100* split of LibriSpeech and evaluate on the *test-clean* and *test-other* splits. The downstream architecture for this task is a feed-forward neural network with CTC loss.

## 5. RESULTS

We first evaluate the speed and memory gains on a controlled toy task (Figure 1). The findings are then extended to real SSL pre-training and the MP3S benchmark.

**Speed and Memory Evaluation.** We give the models randomly generated data of lengths that range from 10 to 80 seconds with 10 seconds intervals. We set the batch size to 6, which was chosen to prevent running out of memory too quickly with MHSA. We perform 10 runs at each input length and time the forward pass as well as measure the VRAM consumption. Only the forward pass is evaluated to concur with a deployment scenario. Averages of computation time and peak VRAM over the 10 runs alongside the 95% bootstrapped confidence interval are plotted in Figure 1. Measurements were taken on an isolated node with a Nvidia A100 80GB GPU.

For input sequences of 10 seconds, the amount of time and VRAM needed is relatively similar for all models. However, a difference starts to appear with 20 seconds, where the

alternatives, on average, use 24% less memory, and run 7% faster relative to MHSA. On the other extreme, at an input of 80 seconds, the alternatives use on average 64% less memory and run 65% faster. SummaryMixing and Fastformer were the fastest compared to MHSA with 15% and 11% increase in speed with an input of 20 seconds and 70% and 71% increase in speed at 80 seconds, respectively. In terms of memory, Mamba and Fastformer proved to be the most memory efficient with 28% and 29% decrease in peak memory with an input of 20 seconds, and 67% and 66% decrease in memory at 80 seconds, respectively. These findings, however, need to be validated with real-scale experiments to make sure that these alternatives can reach decent downstream performance.

For pre-training time estimates, we run 5 epochs and measure the time and max VRAM on Nvidia V100 32GB GPUs keeping the batch size at 1.7 hours as in the full pre-training. We scale these numbers up to 200k steps to get an estimate of the full number GPU hours. All of the alternatives prove to be faster and use less memory except the HyperMixing-LG and Mamba-LG models. We report these in Table 1.

**Speech Recognition on LibriSpeech.** With the LSTM probe (Table 1), MHSA performed better than Mamba, HyperMixing, and SummaryMixing. For the base models with the

Model	Fine-Tune LibriSpeech train-100			
	Clean	Clean LM (C.I.)	Other	Other LM
BRQ	<b>7.01</b>	<b>5.35</b> ( $\pm 0.26$ )	<b>16.98</b>	<b>13.55</b>
HyperConf	8.22	5.77 ( $\pm 0.28$ )	19.29	15.03
Fastformer	9.32	6.82 ( $\pm 0.31$ )	22.75	17.95
SummaryMix	8.72	6.30 ( $\pm 0.28$ )	21.78	16.97
Mamba	7.61	5.50 ( $\pm 0.28$ )	19.97	15.37
BRQ-LG	<b>5.03</b>	<b>3.98</b> ( $\pm 0.21$ )	<b>11.52</b>	<b>9.42</b>
HyperConf-LG	5.87	4.54 ( $\pm 0.32$ )	13.13	10.78
Fastformer-LG	13.16	9.89 ( $\pm 0.34$ )	31.91	26.75
SummaryMix-LG	5.28	4.20 ( $\pm 0.25$ )	12.80	10.50
Mamba-LG	5.59	4.48 ( $\pm 0.25$ )	15.47	12.66

**Table 2:** Results with fine-tuning on LibriSpeech *train-100*. For the *test-clean* set, we report the confidence interval (C.I.).

Contextnet probe, the base version of Mamba outperformed MHSA. For larger models with the Contextnet probe, SummaryMixing performed better than MHSA except with the LM on the *test-other* split, in which SummaryMixing was 0.01 behind MHSA. Despite a dedicated parameter tuning, the large Fastformer falls significantly behind the others.

We then fine-tuned the SSL models on the *train-100* subset of LibriSpeech (Table 2). Confidence intervals come from 1000 bootstraps on the *test-clean* set with a LM. Interestingly, the confidence intervals of HyperMixing and Mamba base models and the SummaryMixing Large model overlap with MHSA best performance. It is therefore unclear if MHSA would always outperform these alternatives.

**Speech Recognition on CommonVoice.** This dataset tells a different story and we hypothesize that this is due to a major issue with most speech SSL model benchmarks. Indeed, LibriSpeech is always used both during the pre-training and downstream evaluation, introducing a clear bias. With CommonVoice (Table 1), and with both probes, MHSA is not the best model anymore. With the linear probe, HyperConformer and SummaryMixing both performed better than MHSA on Welsh and Bask. With the LSTM probe, Mamba performed the best followed by HyperMixing, then SummaryMixing or MHSA depending on the language. This demonstrates the ability of these alternative methods to generalize well to out-of-domain and low-resource datasets compared to MHSA.

**Speaker Verification.** As for CommonVoice, results varied depending on the probe used, however, MHSA clearly is not the best performing solution (Table 1). For the ECAPA probe HyperMixing proved to be the best with an EER of 3.65 and 3.54 for the base and large models respectively. With the Xvectors probe, HyperMixing performed best out of the small models with an EER of 8.54 and for the large model SummaryMixing performed best with an EER of 8.30.

**Intent Classification.** The accuracies observed on this task validate our previous findings (Table 1). All alternatives, except Fastformer, and with both probes performed significantly better than MHSA.

**Emotion Recognition.** This task tells a similar story to intent classification, but with a slightly different leader board (Table 1). Again, MHSA is not the best performing solution. Indeed, Mamba performed the best with the linear probe reaching an accuracy of 64.10% and 66.15% for the base and large model respectively, compared to 60.1% and 62.1% for MHSA. For the ECAPA-TDNN probe, SummaryMixing performed the best with an accuracy of 64.34% and 65.21% for the base and large model respectively.

## 6. DISCUSSION AND CONCLUSION

One of the goals of SSL pre-training is to develop a model that can represent data without the need of labeled data in a way that is useful for a variety of downstream tasks. As this process is expensive, making SSL speech models less resource intensive is an active area of research. Part of this expense is due to multi-head self-attention. While alternatives exist, when it comes to speech tasks, they have only been applied in fully supervised tasks. In this work, we explore replacing multi-head self-attention with four state-of-the-art alternatives: HyperMixing, Fastformer, SummaryMixing and Mamba in a SSL setting for speech tasks. We show that for sequences 20 seconds or more these alternatives are substantially faster and consume less memory. Based on the toy test, and considering that about 75% of LibriSpeech is below 15 seconds, we believe that would see a greater difference in the pre-training time and memory when pre-training with a set such as [11] where all pre-training audio was cropped to be between 32 and 64 seconds.

However, for small input under approximately 20 seconds the amount of time and memory a model consumes are all relatively similar. This threshold, however, could reduce drastically if the model does not combine Mel Filterbanks with a two-dimensional CNN, as the acoustic feature extractor plays a critical role in the length of the sequence arriving to the self-attention module. For instance, one may expect that replacing this CNN with a one-dimensional one would bring this threshold to 10 seconds. Nevertheless, many files in common speech datasets are shorter than 20 seconds and many speech tasks do not require global context from anything over 20 seconds. Splitting audio files based on silence for long audio file processing or streaming is also possible which brings to question the necessity of these alternatives to MHSA.

We believe that future work could involve processing large audio files and developing Large Audio Foundation Models, similar to research trends to include more context in Large Language Models (LLM). With this trend in mind, one

could imagine doing common LLM tasks, such as summarization, that require long context directly from audio without the need of a transcription.

Nevertheless, as a result of our findings, we believe that unless specifically working with long audio files or from the raw waveform, further speed and memory gains will not be obtained by replacing MHSA with alternatives with linear complexity. We believe that more efficient SSL for speech models might be reached by other architectural changes, pruning/quantization methods, and data selection.

## 7. REFERENCES

- [1] Abdelrahman Mohamed, Hung-yi Lee, Lasse Borgholt, Jakob D Havtorn, Joakim Edin, Christian Igel, Katrin Kirchhoff, Shang-Wen Li, Karen Livescu, Lars Maaløe, et al., “Self-supervised speech representation learning: A review,” *IEEE Journal of Selected Topics in Signal Processing*, 2022.
- [2] Shu Wen Yang, Po Han Chi, Yung Sung Chuang, Cheng I Jeff Lai, Kushal Lakhota, Yist Y Lin, Andy T Liu, Jiatong Shi, Xuankai Chang, Guan Ting Lin, et al., “Superb: Speech processing universal performance benchmark,” in *22nd Annual Conference of the International Speech Communication Association, INTERSPEECH 2021*. International Speech Communication Association, 2021, pp. 3161–3165.
- [3] Ha Nguyen, Fethi Bougares, Natalia Tomashenko, Yannick Esteve, and Laurent Besacier, “Investigating self-supervised pre-training for end-to-end speech translation,” in *Interspeech 2020*, 2020.
- [4] Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhota, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, Alexei Baevski, Alexis Conneau, and Michael Auli, “XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale,” in *Proc. Interspeech 2022*, 2022, pp. 2278–2282.
- [5] Yu Zhang, Wei Han, James Qin, Yongqiang Wang, Ankur Bapna, Zhehuai Chen, Nanxin Chen, Bo Li, Vera Axelrod, Gary Wang, et al., “Google usm: Scaling automatic speech recognition beyond 100 languages,” *arXiv preprint arXiv:2303.01037*, 2023.
- [6] William Chen, Xuankai Chang, Yifan Peng, Zhaoheng Ni, Soumi Maiti, and Shinji Watanabe, “Reducing barriers to self-supervised learning: Hubert pre-training with academic compute,” *arXiv preprint arXiv:2306.06672*, 2023.
- [7] Alexei Baevski, Arun Babu, Wei-Ning Hsu, and Michael Auli, “Efficient self-supervised learning with contextualized target representations for vision, speech and language,” in *International Conference on Machine Learning*. PMLR, 2023, pp. 1416–1429.
- [8] Titouan Parcollet, Shucong Zhang, Rogier van Dalen, Alberto Gil CP Ramos, and Sourav Bhattacharya, “On the (in) efficiency of acoustic feature extractors for self-supervised speech representation learning,” in *Interspeech 2023*, 2023.
- [9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser,

- and Illia Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [10] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al., “Conformer: Convolution-augmented transformer for speech recognition,” *Interspeech 2020*, 2020.
- [11] Chung-Cheng Chiu, James Qin, Yu Zhang, Jiahui Yu, and Yonghui Wu, “Self-supervised learning with random-projection quantizer for speech recognition,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 3915–3924.
- [12] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli, “wav2vec 2.0: A framework for self-supervised learning of speech representations,” *Advances in neural information processing systems*, vol. 33, pp. 12449–12460, 2020.
- [13] Ryan Whetten, Titouan Parcollet, Marco Dinarelli, and Yannick Estève, “Open implementation and study of best-rq for speech processing,” *arXiv preprint arXiv:2405.04296*, 2024.
- [14] Florian Mai, Juan Zuluaga-Gomez, Titouan Parcollet, and Petr Motlicek, “HyperConformer: Multi-head HyperMixer for Efficient Speech Recognition,” in *Proc. INTERSPEECH 2023*, 2023, pp. 2213–2217.
- [15] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie, “Fastformer: Additive attention can be all you need,” *arXiv preprint arXiv:2108.09084*, 2021.
- [16] Titouan Parcollet, Rogier van Dalen, , Shucong Zhang, and Sourav Bhattacharya, “SummaryMixing: A linear-complexity alternative to self-attention for speech recognition and understanding,” 2023, *arXiv:2307.07421*.
- [17] Albert Gu and Tri Dao, “Mamba: Linear-time sequence modeling with selective state spaces,” *arXiv preprint arXiv:2312.00752*, 2023.
- [18] Salah Zaiem, Youcef Kemiche, Titouan Parcollet, Slim Essid, and Mirco Ravanelli, “Speech Self-Supervised Representation Benchmarking: Are We Doing it Right?,” in *INTERSPEECH 2023*, 2023, pp. 2873–2877.
- [19] Mirco Ravanelli, Titouan Parcollet, Peter Plantinga, Aku Rouhe, Samuele Cornell, Loren Lugosch, Cem Subakan, Nauman Dawalatabad, Abdelwahab Heba, Jianyuan Zhong, et al., “Speechbrain: A general-purpose speech toolkit,” *arXiv preprint arXiv:2106.04624*, 2021.
- [20] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al., “Big bird: Transformers for longer sequences,” *Advances in neural information processing systems*, vol. 33, pp. 17283–17297, 2020.
- [21] Iz Beltagy, Matthew E. Peters, and Arman Cohan, “Longformer: The long-document transformer,” *arXiv:2004.05150*, 2020.
- [22] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma, “Linformer: Self-attention with linear complexity,” *arXiv preprint arXiv:2006.04768*, 2020.
- [23] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret, “Transformers are RNNs: Fast autoregressive transformers with linear attention,” in *Proceedings of the 37th International Conference on Machine Learning*, Hal Daumé III and Aarti Singh, Eds. 13–18 Jul 2020, vol. 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165, PMLR.
- [24] Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe, “Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 17627–17643.
- [25] Florian Mai, Arnaud Pannatier, Fabio Fehr, Haolin Chen, Francois Marelli, François Fleuret, and James Henderson, “Hypermixer: An mlp-based low cost alternative to transformers,” in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 15632–15654.
- [26] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al., “Mlp-mixer: An all-mlp architecture for vision,” *Advances in neural information processing systems*, vol. 34, pp. 24261–24272, 2021.
- [27] David Ha, Andrew M Dai, and Quoc V Le, “Hypernetworks,” in *International Conference on Learning Representations*, 2016.
- [28] Xiangyu Zhang, Qiquan Zhang, Hexin Liu, Tianyi Xiao, Xinyuan Qian, Beena Ahmed, Eliathamby Ambikairajah, Haizhou Li, and Julien Epps, “Mamba in speech: Towards an alternative to self-attention,” *arXiv preprint arXiv:2405.12609*, 2024.
- [29] 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)*. IEEE, 2015, pp. 5206–5210.

- [30] Arsha Nagrani, Joon Son Chung, and Andrew Senior, “Voxceleb: a large-scale speaker identification dataset,” *arXiv preprint arXiv:1706.08612*, 2017.
- [31] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, “X-vectors: Robust dnn embeddings for speaker recognition,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5329–5333.
- [32] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck, “ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification,” in *Proc. Interspeech 2020*, 2020, pp. 3830–3834.
- [33] Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser, “SLURP: A spoken language understanding resource package,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, Eds., Online, Nov. 2020, pp. 7252–7262, Association for Computational Linguistics.
- [34] Rosana Ardila, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, and Gregor Weber, “Common voice: A massively-multilingual speech corpus,” in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, Eds., Marseille, France, May 2020, pp. 4218–4222, European Language Resources Association.
- [35] Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu, “ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context,” in *Proc. Interspeech 2020*, 2020, pp. 3610–3614.