PERFORMANCE-EFFICIENCY TRADE-OFFS IN UNSUPERVISED PRE-TRAINING FOR SPEECH RECOGNITION

Felix Wu[†] *Kwangyoun Kim*[†] *Jing Pan*[†] *Kyu J. Han*[†] *Kilian Q. Weinberger*[†‡] *Yoav Artzi*[†‡]

[†]ASAPP Inc. [‡]Cornell University

ABSTRACT

This paper is a study of performance-efficiency trade-offs in pre-trained models for automatic speech recognition (ASR). We focus on wav2vec 2.0, and formalize several architecture designs that influence both the model performance and its efficiency. Putting together all our observations, we introduce *SEW-D (Squeezed and Efficient Wav2vec with Disentangled Attention)*, a pre-trained model architecture with significant improvements along both performance and efficiency dimensions across a variety of training setups. For example, under the 100h-960h semi-supervised setup on LibriSpeech, SEW-D achieves a 1.9x inference speedup compared to wav2vec 2.0, with a 13.5% relative reduction in word error rate. With a similar inference time, SEW reduces word error rate by 25–50% across different model sizes.

Index Terms— Speech Recognition, Unsupervised, Efficiency

1. INTRODUCTION

Recently, there is significant interest in self-supervised pre-training using unlabeled audio data to learn versatile feature representations, that are subsequently fine-tuned on task-specific annotated audio [1, 2, 3, 4]. This follows similar trends in natural language processing [NLP; 5, 6, 7] and computer vision [CV; 8, 9, 10]. The most prominent example of this class of models is wav2vec 2.0 [W2V2; 11], which achieves competitive word error rate (WER) following fine-tuning on only ten minutes of transcribed (labeled) data, when prior supervised approaches often require nearly a thousand hours. If recent developments in NLP and CV are any indication, the importance of such pre-trained audio models will only increase. Indeed, W2V2 has already been studied with focus on the impact of pre-training data [12, 13], pre-training task [14], or combination with pseudo labelling [3, 1].

In this paper, we study trade-offs in W2V2's model design, and propose SEW-D (Squeezed and Efficient Wav2vec with Disentangled Attention). Our focus is on efficiency for practical applications. As W2V2-type models become increasingly common, understanding their efficiency trade-offs is critical for their real-world deployment, where any increase in efficiency can substantially reduce the inference costs and energy footprints across a plethora of applications. SEW-D achieves superior performance-efficiency trade-off compared to the original W2V2 by using (a) a squeezed context network, (b) a compact wave feature extractor, (c) MLP projection heads [9], and (d) disentangled attention modules [7].

Compared to the official W2V2-large release, our best SEW-D-base+ achieves 2.7× and 3.2× speed-ups for inference and pre-training with comparable WER using half the number of parameters. Compared to W2V2-base, our SEW-D-mid achieves 1.9× inference speed-up with a 13.5% relative reduction in WER. Fig. 1 (left) shows the performance-efficiency trade-offs with various model sizes. SEW-D outperforms W2V2 in most pre-training settings, when experimenting with LibriSpeech [15], Ted-lium 3 [16], VoxPopuli [17], and Switchboard [18] datasets. Pre-trained models and code are available at `https://github.com/asappresearch/sew`.

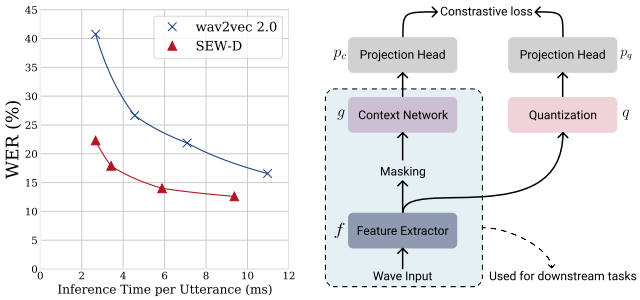

Fig. 1: **Left**: Word error rate (WER) and average utterance inference time on LibriSpeech (dev-other) of wav2vec 2.0 and our SEW-D models fine-tuned with 100h labeled data for 100K updates. **Right**: Wav2vec 2.0 framework.

2. TECHNICAL BACKGROUND: WAV2VEC 2.0 (W2V2)

W2V2 is made of a waveform feature extractor that generates a sequence of continuous feature vectors, each encoding a small segment of audio, and a context network that maps these vectors to context-dependent representations. Fig. 1 (right) illustrates the W2V2 framework, including (a) a feature extractor, (b) a context network, (c) an optional quantization module, and (d) two projection heads.

During pre-training, some of the features are masked out, and are not seen by the context network. In parallel, the pre-masking features are discretized as prediction targets. The context network is optimized to discriminate the discretized version of the original features at the masked positions from a pool of negative samples using an InfoNCE loss [19].

2.1. Wave Feature Extractor (WFE)

The wave feature extractor $f(\cdot)$ encodes and downsamples the raw waveform audio inputs $\mathbf{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_{T_{\text{input}}}) \in \mathbb{R}^{T_{\text{input}} \times d_{\text{input}}}$ ($d_{\text{input}} = 1$ for single-channel audio) into an array of feature vectors $\mathbf{Z} = f(\mathbf{X}) = (\boldsymbol{z}_1, ..., \boldsymbol{z}_T) \in \mathbb{R}^{T \times \mathbb{R}^{d_{\text{feat}}}}$. For example, W2V2 maps 16KHz audio sequences to 50Hz frames using a convolutional WFE with receptive field size of 400 and stride size of 320. Each feature vector encodes the raw signals within a 25ms ($= 1000/16000 \times 400$) window with a stride size 20ms ($= 1000/16000 \times 320$). The reduced sequence length is $T = \frac{T_{\text{input}} - 400}{320} + 1 = \frac{T_{\text{input}} - 80}{320}$.

2.2. Context Network

The context network $g(\cdot)$ follows a similar principle as masked language models in NLP (e.g., BERT [5] or RoBERTa [6]). During pre-training, each z_t is masked and replaced with a trainable mask vector $\boldsymbol{m}$ with a predefined probability p. To illustrate, $\mathbf{Z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3, \boldsymbol{z}_4, \boldsymbol{z}_5, \boldsymbol{z}_6, ..., \boldsymbol{z}_T)$ can become $\mathbf{Z}' = (\boldsymbol{z}_1, \boldsymbol{m}, \boldsymbol{m}, \boldsymbol{z}_4, \boldsymbol{m}, \boldsymbol{z}_6, ..., \boldsymbol{z}_T)$. The context network maps this masked sequence to a sequence of contextual representations $\mathbf{C} = g(\mathbf{Z}') = (\boldsymbol{c}_1, ..., \boldsymbol{c}_T) \in \mathbb{R}^{T \times d_{\text{feat}}}$ to incorporate context information. Even if $\boldsymbol{z}_t$ is masked and replaced with $\boldsymbol{m}$, we anticipate that $\boldsymbol{c}_t$ can recover the information in $\boldsymbol{z}_t$ because it contains information from surrounding, unmasked input vectors. The context network is usually implemented with a Transformer architecture [20, 21].

2.3. Quantization Module

The quatization module $q(\cdot)$ maps each unmasked vector $\boldsymbol{z}_t$ into a quantized form $\boldsymbol{q}_t = q(\boldsymbol{z}_t) \in \mathbb{R}^{d_{\text{feat}}}$ for each masked position t. Quantized $\boldsymbol{q}_t$'s are the prediction targets. The quantization module is based on Gumbel softmax with straight-through estimator [22, 23, 24]. There are G codebooks and each codebook has V entries, giving $G \times V$ vectors $\boldsymbol{e}_{g,v} \in \mathbb{R}^{\frac{d_{\text{feat}}}{G}}$. For each group g, the probability of assigning $\boldsymbol{z}_t$ to the v-th entry is $p_{g,v} = \frac{\exp(\boldsymbol{W}_v^g \cdot \boldsymbol{z}_t / \tau_Q)}{\sum_{v'=1}^{V} \exp(\boldsymbol{W}_{v'}^g \cdot \boldsymbol{z}_t / \tau_Q)}$, where $\boldsymbol{W}^g \in \mathbb{R}^{V \times d_{\text{feat}}}$ is a trainable matrix and τ_Q is a quantization temperature. For each group g, $\boldsymbol{z}_t$ is assigned to the v_g^*-th entry where $v_g^* = \arg\max_v p_{g,v}$. The corresponding embedding vectors $(\boldsymbol{e}_{1,v_1^*}, ..., \boldsymbol{e}_{G,v_G^*})$ are concatenated to a single vector $\boldsymbol{q}_t \in \mathbb{R}^{d_{\text{feat}}}$, to create a quantized feature sequence $\mathbf{Q} = (\boldsymbol{q}_1, ..., \boldsymbol{q}_T) \in \mathbb{R}^{T \times d_{\text{feat}}}$.

2.4. Projection Heads

Two linear projection heads $p_c(\cdot)$ and $p_q(\cdot)$ reduce the dimensionality of $\mathbf{C}$ and $\mathbf{Q}$. For a $\boldsymbol{z}_t$ that is masked and replaced with $\boldsymbol{m}$, we want $p_c(\boldsymbol{c}_t) \in \mathbb{R}^{d_{\text{proj}}}$ to be similar to $p_q(\boldsymbol{q}_t) \in \mathbb{R}^{d_{\text{proj}}}$. Baevski et al. [11] do not separate between p_c and g or p_q and q in their original notations. However, we keep the distinctions, as they serve different roles and are discarded before downstream fine-tuning.

2.5. Pre-training Objective

W2V2 combines contrastive and diversity losses for pre-training: $\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d$. The goal of the contrastive loss $\mathcal{L}_m$ is to make the projected outputs $p_c(\boldsymbol{c}_t)$ close to $p_q(\boldsymbol{q}_t)$ and far away from any other $p_q(\boldsymbol{q}_{t'})$, where the feature vector $\boldsymbol{z}_t$ is masked and t' is any other position in the same sequence. W2V2 uses an InfoNCE loss [19]:

$$\mathcal{L}_m = \mathbb{E}_{\boldsymbol{z}_t = \boldsymbol{m}} \left[-\log \frac{\exp(\text{sim}(p_c(\boldsymbol{c}_t), p_q(\boldsymbol{q}_t))/\kappa)}{\sum_{\boldsymbol{q}_{t'} \in \mathbb{Q}} \exp(\text{sim}(p_c(\boldsymbol{c}_t), p_q(\boldsymbol{q}_{t'}))/\kappa)} \right],$$

with $\text{sim}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a}^\top \boldsymbol{b}}{\|\boldsymbol{a}\| \|\boldsymbol{b}\|}$, $\mathbb{Q}$ is a set containing the positive sample $\boldsymbol{q}_t$ and K negative samples, and κ is the temperature. The expectation is computed over masked positions only. The diversity loss $\mathcal{L}_d$ prevents the quantization module from collapsing to a trivial mapping (e.g., by collapsing all inputs to a single discrete code). It encourages the quantization probability $p_{g,v}$ to be evenly distributed:

$$\mathcal{L}_d = \mathbb{E}_t \left[1 - \frac{1}{GV} \sum_{g=1}^{G} \exp\left(-\sum_{v=1}^{V} p_{g,v} \log p_{g,v} \right) \right].$$

3. SEW-D

We propose *SEW-D (Squeezed and Efficient Wav2vec with Disentangled Attention)*, an efficient pre-trained model architecture. SEW-D differs from W2V2 in: (a) using a squeezed context network, (b) replacing WFE-O with WFE-C, (c) using MLP predictor heads with BatchNorm, and (d) replacing the normal multi-head attention with disentangled attention [7].

3.1. Squeezed Context Networks

Transformers require $O(n^2)$ time complexity where n is the sequence length. To reduce the sequence length, we propose to encode the features at a low resolution (e.g., 25Hz) while keeping contrastive learning at a high resolution (e.g., 50Hz). We add a down-sampling layer and an up-sampling layer around the original context network. Because there is already a convolution layer at the bottom of the W2V2 context network, we simply change its stride size from 1 to s to avoid additional computation, where s is the squeezing factor.[1] The up-sample layer is a transposed $1d$ convolution with kernel size s and stride size s ($s = 2$ in our experiments). Fig. 2 illustrates the context network squeezing.

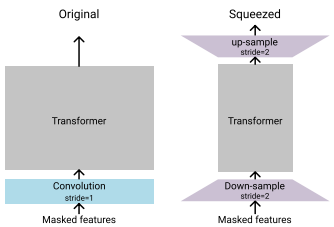

Fig. 2: Original vs. squeezed context network. The sequence length is halved by the down-sampling.

3.2. Compact Wave Feature Extractors (WFE-C)

W2V2 has the same number of channels in all layers of its convolutional wave feature extractor (WFE-O; "O" stands for original). The first few layers consume much of the computation time, while the last three consume less than 10% of the total computation. We hypothesize that the first few layers are unnecessarily large, and that the computation can be more evenly distributed across layers.

We introduce a compact wave feature extractor (WFE-C) which doubles the number of channel when the sequence length is down-sampled by 4 times. The progression of channel dimensionality is (c, $2c$, $2c$, $4c$, $4c$, $8c$, $8c$) across its 7 conv layers where c is a hyper-parameter. We keep the kernel sizes (7, 3, 3, 3, 3, 2, 2) and strides (5, 2, 2, 2, 2, 2, 2) of WFE-O. We further study scaling up WFE-C by adding a point-wise (kernel size 1) convolutional layer after each original convolutional layer except for the first layer, which creates a 13-layers convolutional network with kernel sizes (7, 3, 1, 3, 1, 3, 1, 3, 1, 2, 1, 2, 1) and strides (5, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1). We refer this model as WFE-C-c128-l1, where "l1" denotes one additional intermediate layer between every two original layers.

3.3. MLP Predictor Heads with BatchNorm

Chen et al. [9] use MLP predictor heads instead of linear ones for unsupervised image representation learning, leading to better pre-trained features with little overhead during pre-training. We replace the linear projection of W2V2 with a two-layer MLP with hidden size 4096, a ReLu activation in between, and BatchNorm [25] after

[1] There is a shortcut connection in W2V2 that adds the convolution inputs to its outputs and passes it to the Transformer. We do average pooling with kernel and stride sizes s in this shortcut path that averages every two steps into one so that it can be added to the outputs of the strided convolution.

Model	# P aram. (M)	Time PT (h)	Time Infer. (s)	WER (No LM / + LM) test-clean	WER (No LM / + LM) test-other
W2V2-tiny	11.1	24.7	7.5±0.04	22.8 / 8.3	42.1 / 25.6
SEW-D-tiny	24.1	23.8	7.5±0.02	**10.4 / 4.9**	**22.8 / 13.9**
W2V2-small	24.8	34.6	12.8±0.05	12.8 / 5.7	27.2 / 16.0
SEW-D-small	41.0	38.1	**9.6**±0.04	**7.8 / 4.2**	18.2 / **11.4**
W2V2-mid	44.1	**40.3**	19.9±0.04	9.6 / 4.8	22.2 / 13.5
SEW-D-mid	78.8	51.7	**16.5**±0.03	**6.4 / 3.8**	**14.2 / 9.5**
W2V2-base	94.4	**55.2**	30.8±0.05	7.1 / 4.0	16.4 / 10.4
SEW-D-base	175.1	59.1	**26.3**±0.06	5.8 / 3.6	13.2 / 9.3
SEW-D-base+	177.0	68.4	27.8±0.05	**5.3 / 3.5**	**12.6 / 9.0**

Table 1: Libri-Speech 100h-960h semi-supervised setup pretrained for 100K updates.

Model	WFE Type	WFE c	WFE l	Context Network Conv-k	Context Network Sq.	Context Network E	Context Network L	Context Network D-Attn	Pred. Head Layer	Pred. Head BN	Infer. Time
W2V2-tiny	O	256		128		256	12		1		7.5±0.04
W2V2-small	O	384		128		384	12		1		12.8±0.05
W2V2-mid	O	512		128		512	12		1		19.9±0.04
W2V2-base	O	512		128		768	12		1		30.8±0.05
W2V2-large	O	512		128		1024	24		1		74.4±0.05
SEW-D-tiny	C	64	1	31	2	384	12	✓	2	✓	7.5±0.02
SEW-D-small	C	64	1	31	2	512	12	✓	2	✓	9.6±0.04
SEW-D-mid	C	64	1	31	2	512	24	✓	2	✓	16.5±0.03
SEW-D-base	C	64	1	31	2	768	24	✓	2	✓	26.3±0.06
SEW-D-base+	C	96	1	31	2	768	24	✓	2	✓	27.8±0.05

Table 2: Model hyper-parameters, categorized by inference time. We focus on performance-efficiency trade-off, and therefore we do not control for model size within each time category.

each linear layer. Because the predictor heads are discarded following pre-training, there is no inference overhead.

3.4. SEW-D (SEW with Disentangled Attention)

Disetangled attention [7] is a variant of self-attention with relative position representations [26], which outperforms Transformer's multi-head attention [20] on various NLP tasks. Unlike Transformer which adds absolute positional embeddings to the content embeddings at the beginning, disentangled attention keeps the positional embeddings and content embeddings separate and has three components in its attention weight computation: (a) content-to-content, (b) content-to-position, and (c) position-to-content attentions.

4. EXPERIMENTS

4.1. Experimental Setup

We use the official W2V2 implementation in fairseq [27], with the hyper-parameters of W2V2-base [11]. Table 2 shows the hyper-parameters of W2V2 and SEW-D with different inference budgets. We describe key hyper-parameters; the linked configuration files provide the full details.

Pre-training: We use the LibriSpeech [15] 960h training data for unsupervised pre-training, leaving 1% as validation set for pre-training. We use the same hyperparameters as W2V2-base.[2] To speed-up and reduce the cost of our experiments, we pre-train all models for 100K updates similar to Hsu et al. [14]. All experiments use an AWS p3.16xlarge instance with 8 NVIDIA V100 GPUs and 64 Intel Xeon 2.30GHz CPU cores. Because Baevski et al. [11] use 64 GPUs, we set the gradient accumulation steps to 8 to simulate their 64-GPU pre-training with 8 GPUs.

Model	Inference Time (s)	WER (%) on test-other 10m	1h	10h
W2V2-base	30.8±0.05	54.0±1.18	40.0±2.34	22.6±0.15
+ LM		**28.2**±0.23	18.4±0.65	13.8±0.13
SEW-D-mid	**16.5**±0.03	59.5±1.09	36.8±0.18	20.7±0.07
+ LM		33.4±1.20	17.5±0.11	13.0±0.07
SEW-D-base+	27.8±0.05	**49.3**±0.42	**28.9**±0.11	**18.1**±0.06
+ LM		30.6±0.25	**17.4**±0.38	**11.7**±0.24

Table 3: LibriSpeech results with 10m, 1h, or 10h supervised data. All the models are pre-trained for 100K updates on LibriSpeech 960h. We report inference times on dev-clean and WERs without LM and with 4-gram LM (beam size 50). The mean and standard deviations are calculated from three random runs. Similar to BERT-large [31], we observe that our SEW-D-base+ is unstable during fine-tuning. We run five random runs and stop two degenerate runs after 3K updates.

Fine-tuning: We add a linear classifier to the top of the context network and fine-tune the model using a CTC objective on LibriSpeech train-clean 100h set for 80K updates using the same set of hyper-parameters as W2V2-base.[3]

Evaluation: We use CTC greedy decoding [28] for all experiments because it is faster than Viterbi decoding [29] and we do not find any WER differences between the two using baseline W2V2 models. We use LibriSpeech dev-other for validation, and hold out test-clean and test-other as test sets. We consider three metrics to evaluate model efficiency and performance: pre-training time, inference time, and WER. All evaluation is done on an NVIDIA V100 GPU with FP32 operations, unless specified otherwise. When decoding with a language model (LM), we use the official 4-gram LM[4] and the wav2letter [30] decoder[5] with the default LM weight 2, word score -1, and beam size 50. Reducing the inference time with LM is an important direction for future work, as the wav2letter decoder is the bottleneck and is at least 3× slower than W2V2-base.

4.2. Initial Results: Pre-training with 100K Updates

We pre-train W2V2 and SEW-D on 960h LibriSpeech audio for 100K updates and fine-tune them on 100h labelled data. Table 1 shows pre-training times, inference times, and WER with and without an LM. Without an LM, compared with W2V2-tiny, SEW-D-tiny reduces WER by 54.4% (22.8% to 10.4%) and 45.8% (42.1% to 22.8%) on test-clean and test-other with the same inference speed. With an LM, WER improves by 41.0% and 45.7% on test-clean and test-other. Compared with the W2V2-mid, SEW-D-mid reduces WER by 33.0% (9.6% to 6.4%) and 36.0% (22.2% to 14.2%) with 17.1% reduction in inference time. SEW does incur slight increase in training time compared to W2V2 with similar inference times. However, SEW has lower WER even compared to a slower W2V2 which has more parameters and takes longer to train (e.g., SEW-D-small vs. W2V2-mid or SEW-D-mid vs. W2V2-base).

We also experiment with only 10min, 1h, and 10h of supervised

[2]https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/config/pretraining/wav2vec2_base_librispeech.yaml

[3]https://github.com/pytorch/fairseq/blob/master/examples/wav2vec/config/finetuning/base_100h.yaml

[4]https://www.openslr.org/resources/11/4-gram.arpa.gz

[5]https://github.com/flashlight/wav2letter/tree/v0.2/bindings/python

Model	# Param. (M)	Time PT (GPU-day)	Time Infer. (s)	WER (No LM / + LM) test-clean	WER (No LM / + LM) test-other
W2V2-base	94.4	102.4(73.5)	30.8±0.05	6.1 / 3.5	13.3 / 8.7
W2V2-large	315.5	294.4	74.4±0.40	4.7 / **2.9**	**9.1** / **6.4**
SEW-D-mid	78.8	**68.9**	**16.5**±0.03	4.9 / 3.3	11.5 / 8.2
SEW-D-base+	177.0	91.2	27.8±0.05	**4.4** / 3.1	9.2 / 7.0

Table 4: LibriSpeech 100h-960h semi-supervised setup pre-trained for 400K updates. We use the public W2V2s [11] as the baselines. Notably, W2V2-base is reported as taking 102.4 GPU-days to pre-train on 64 GPUs, but from our estimation it only takes 73.5 GPU-days on 8 GPUs. Unlike Baevski et al. [11], we neither tune the decoding hyper-parameters nor use a huge beam size 1500.

data. Table 3 shows WER for W2V2 and SEW-D. SEW-D-mid outperforms W2V2-base in the 1h and 10h scenarios while being more efficient. SEW-D-mid is worse than W2V2-base in the extreme 10m scenario; however, we did not tune the fine-tuning hyper-parameters and use the ones tuned for W2V2-base. SEW-D-base+ achieves significantly better performance than W2V2-base in most of the setups except for using 10 minutes supervision and decoded with LM. Potentially due to a large model size, we observe that SEW-D-base+ is unstable to fine-tune; therefore, instead of using W2V2's tuned hyperparameters, we reduce the learning rate by 5 times to 10^{-5}, set the dropout rates as the pre-training (W2V2 uses different sets of dropouts during pre-training and fine-tuning), and do not freeze the context network at the beginning of the fine-tuning. These adjustments stabilizes the fine-tuning of the model.

4.3. Comparison to Published Results

We continue training our best SEW-D-mid model to 400K updates to compare it with the official W2V2-base[6] and W2V2-large[7] checkpoints [11]. Table 4 shows inference times and WERs with or without an LM. Compared to W2V2-base, SEW-D-mid reduces inference time by 46.4% (a 1.9× speed-up) and WER by 19.7% and 13.5% on both test sets without an LM; SEW-D-base+ reduces inference time by 9.7% and WER by 27.9% and 30.8%. Compared to W2V2-large, SEW-D-base+ achieves 2.7× and 3.2× speed-ups for inference and pre-training with comparable WER and half of the model size.

We evaluate W2V2 and SEW-D pre-trained models on three additional ASR datasets: TED-LIUM 3 (TL3) [16], VoxPopuli (VP) [17], and Fisher+Switchboard (F+SB) [18, 32, 33] with a similar setup to Hsu et al. [13]. We use 10h of supervised audio to stress-test low resource domain transfer. Table 5 shows inference times and WERs. SEW-D-mid consistently reduces inference times by about 30% while providing lower WERs on TL3, similar WERs on VP, and slightly higher WERs on F+SB. SEW-D-base+ consistently outperforms W2V2-base by a large margin while being only 10% slower.

4.4. Ablation Study

We conduct an ablation study on tiny models with similar inference time.[8] Table 6 shows the model size inference time, and WER on LibriSpeech dev-other set. Rows 1 and 2 are W2V2 baseline. Row 3 improves on row 2 by using a squeeze context network. Row 4 replaces WFE-O with WFE-C, and row 5 reduces the size of WFE-C

[6]https://dl.fbaipublicfiles.com/fairseq/wav2vec/wav2vec_small_100h.pt

[7]https://dl.fbaipublicfiles.com/fairseq/wav2vec/wav2vec_big_100h.pt

[8]A longer version of this paper (https://arxiv.org/abs/2109.06870) provides a more detailed ablation study.

Model	TED-LIUM 3 (10h) Time (s)	TED-LIUM 3 (10h) test (%)	VoxPopuli (10h) Time (s) (%)	VoxPopuli (10h) test (%)	Fisher+SB (10h) Time (s)	Fisher+SB (10h) test (%)
W2V2-base	10.2±0.03	14.8±0.13	30.8±0.22	18.1±0.03	33.8±0.12	23.2±0.10
+ LM		9.3±0.09		11.6±0.01		16.7±0.00
SEW-D-mid	**6.9**±0.02	13.9±0.04	**20.3**±0.08	18.1±0.03	**21.6**±0.07	24.3±0.49
+ LM		9.2±0.11		11.2±0.26		18.3±0.53
SEW-D-base+	11.2±0.01	**12.2**±0.14	33.8±0.04	**15.5**±0.35	36.8±0.03	**20.5**±0.06
+ LM		**8.7**±0.22		**10.9**±0.39		**15.7**±0.06

Table 5: Inference time and WER of LibriSpeech pre-trained models (400K updates) transferred to TED-LIUM 3, VoxPopuli, and Fisher+Switchboard datasets with only 10h labels. SEW-D-mid outperforms W2V2 base on all settings while being at least 30% faster. We report the inference time on the dev sets. The mean and standard deviations are computed over three random runs.

#	Model	# Par. (M)	Infer. Time (s)	Dev-other WER	Dev-other (+LM)
1	W2V2-small (E384L12 + WFE-O-c384)	24.8	12.8±0.05	26.6	15.6
2	W2V2-tiny (E256L12 + WFE-O-c256)	11.1	7.5±0.04	40.7	23.4
3	Sq-W2V E384L12 + WFE-O-c256	23.9	7.0±0.01	29.9	17.5
4	Sq-W2V E384L12 + WFE-C-c96-l1	27.2	7.1±0.04	29.6	17.2
5	Sq-W2V E512L12 + WFE-C-c48-l1	41.7	7.1±0.01	24.4	14.7
6	Sq-W2V E512L12 (conv k31) + WFE-C-c64-l1	40.7	**6.7**±0.02	24.4	14.5
7	+ MLP predictor heads (SEW-tiny)	40.7	**6.7**±0.02	23.7	14.4
8	+ Disentangled attention (SEW-D-tiny)	24.1	7.5±0.02	**22.3**	**13.4**
9	#7 - WFE-C-c64-l1 + FBFE-c160	42.2	7.5±0.02	25.0	15.1

Table 6: Ablation study with a controlled inference time. Rows 2 → 3: using a squeezed context network. Rows 3 → 4: replace WFE-O with WFE-C. Rows 4 → 5: Smaller WFE and larger Transformer. Rows 5 → 6: Smaller conv kernel size in the context network and wider WFE. Rows 6 → 7: using MLP predictor heads. Rows 7 → 8: using 80D filter bank features (FBFE-c160, i.e. Filter Bank Feature Extractor with two 2D convolutional layers and 160 channels followed by a linear projection layer) instead of raw waveform (WFE-C). Rows 7 → 9: using disentangled attention and reduces the width from 512 to 384 to match the inference time.

which allows us to use a larger context network. Row 6 reduces the size of the convolution channels in the downsampling layer of the squeezed context network and increase the size WFE-C. Row 7 uses MLP predictor heads and finally, row 8 replaces the multi-head attention with disentangled attention. As we can see from rows 2–8, the WER keep decreasing as we add new techniques. We also replace the WFE-C with filter bank feature extractor (FBFE) to justify the use of raw waveform features instead of the popular log-mel filterbank features as the inputs (row 9).

5. CONCLUSION

While model performance is commonly prioritized in research, the economics of inference times are often as critical for model deployment in the real world. We have demonstrated that our proposed SEW-D achieves significantly better performance-efficiency trade-off than the existing W2V2 on a variety of fine-tuning setups for ASR. We provide strong tiny models with reasonable WER trained using only eight GPUs within a day (compared to Baevski et al. [11] pre-training W2V2-base with 64 GPUs for 1.6 days), which significantly reduces the cost of each round of experiment and benefits the future self-supervised learning researches in the community. In future work, we plan to apply SEW-D to other downstream speech tasks as well as improve pre-training framework for SEW-D.

References

[1] Y. Zhang, J. Qin, D. Park, W. Han, C. Chiu, R. Pang, Q. V. Le, and Y. Wu, "Pushing the limits of semi-supervised learning for automatic speech recognition," *arXiv:2010.10504*, 2020.

[2] C. Wang, A. Wu, J. Pino, A. Baevski, M. Auli, and A. Conneau, "Large-scale self- and semi-supervised learning for speech translation," *arXiv:2104.06678*, 2021.

[3] Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, and M. Auli, "Self-training and pre-training are complementary for speech recognition," *arXiv:2010.11430*, 2020.

[4] L. Pepino, P. Riera, and L. Ferrer, "Emotion recognition from speech using wav2vec 2.0 embeddings," *arXiv:2104.03502*, 2021.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*, 2018.

[6] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv:1907.11692*, 2019.

[7] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *arXiv:2006.03654*, 2020.

[8] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," *arXiv:1911.05722*, 2019.

[9] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv:2002.05709*, 2020.

[10] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," *arXiv:2006.07733*, 2020.

[11] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv:2006.11477*, 2020.

[12] A. Conneau, A. Baevski, R. Collobert, A. rahman Mohamed, and M. Auli, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv:2006.13979*, 2020.

[13] W.-N. Hsu, A. Sriram, A. Baevski, T. Likhomanenko, Q. Xu, V. Pratap, J. Kahn, A. Lee, R. Collobert, G. Synnaeve, and M. Auli, "Robust wav2vec 2.0: Analyzing domain shift in self-supervised pre-training," *arXiv:2104.01027*, 2021.

[14] W.-N. Hsu, Y.-H. H. Tsai, B. Bolte, R. Salakhutdinov, and A. Mohamed, "Hubert: How much can a bad teacher benefit asr pre-training," in *Self-Supervised Learning for Speech and Audio Processing Workshop @ NeurIPS*, 2020.

[15] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," *ICASSP*, 2015.

[16] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Estève, "Ted-lium 3: twice as much data and corpus repartition for experiments on speaker adaptation," *arXiv:1805.04699*, 2018.

[17] C. Wang, M. Rivière, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," *arXiv:2101.00390*, 2021.

[18] J. Godfrey and E. Holliman, "Switchboard-1 release 2 ldc97s62," *Web Download. Linguistic Data Consortium, Philadelphia*, 1993.

[19] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv:1807.03748*, 2018.

[20] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv:1706.03762*, 2017.

[21] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv:2005.08100*, 2020.

[22] E. J. Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*. US Government Printing Office, 1954, vol. 33.

[23] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv:1611.01144*, 2016.

[24] C. J. Maddison, D. Tarlow, and T. Minka, "A* sampling," *arXiv:1411.0030*, 2014.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167*, 2015.

[26] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *NAACL-HLT*, 2018.

[27] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "fairseq: A fast, extensible toolkit for sequence modeling," in *NAACL-HLT*, 2019.

[28] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," *ICML*, 2006.

[29] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, 1967.

[30] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv:1609.03193*, 2016.

[31] J. Phang, T. Févry, and S. R. Bowman, "Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks," *arXiv:1811.01088*, 2018.

[32] C. Cieri, D. Graff, O. Kimball, D. Miller, and K. Walker, "Fisher english training speech parts 1 and 2 ldc200{4,5}s13," *Web Download. Linguistic Data Consortium, Philadelphia*, 2004,2005.

[33] ——, "Fisher english training speech parts 1 and 2 transcripts ldc200{4,5}t19," *Web Download. Linguistic Data Consortium, Philadelphia*, 2004,2005.