arXiv:2006.11477v3 [cs.CL] 22 Oct 2020

wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations

Alexei Baevski **Henry Zhou** **Abdelrahman Mohamed** **Michael Auli**

{abaevski,henryzhou7,abdo,michaelauli}@fb.com

Facebook AI

Abstract

We show for the first time that learning powerful representations from speech audio alone followed by fine-tuning on transcribed speech can outperform the best semi-supervised methods while being conceptually simpler. wav2vec 2.0 masks the speech input in the latent space and solves a contrastive task defined over a quantization of the latent representations which are jointly learned. Experiments using all labeled data of Librispeech achieve 1.8/3.3 WER on the clean/other test sets. When lowering the amount of labeled data to one hour, wav2vec 2.0 outperforms the previous state of the art on the 100 hour subset while using 100 times less labeled data. Using just ten minutes of labeled data and pre-training on 53k hours of unlabeled data still achieves 4.8/8.2 WER. This demonstrates the feasibility of speech recognition with limited amounts of labeled data.[1]

1 Introduction

Neural networks benefit from large quantities of labeled training data. However, in many settings labeled data is much harder to come by than unlabeled data: current speech recognition systems require thousands of hours of transcribed speech to reach acceptable performance which is not available for the vast majority of the nearly 7,000 languages spoken worldwide [31]. Learning purely from labeled examples does not resemble language acquisition in humans: infants learn language by listening to adults around them - a process that requires learning good representations of speech.

In machine learning, self-supervised learning has emerged as a paradigm to learn general data representations from unlabeled examples and to fine-tune the model on labeled data. This has been particularly successful for natural language processing [43, 45, 9] and is an active research area for computer vision [20, 2, 36, 19, 6].

In this paper, we present a framework for self-supervised learning of representations from raw audio data. Our approach encodes speech audio via a multi-layer convolutional neural network and then masks spans of the resulting latent speech representations [26, 56], similar to masked language modeling [9]. The latent representations are fed to a Transformer network to build contextualized representations and the model is trained via a contrastive task where the true latent is to be distinguished from distractors [54, 49, 48, 28] (§ 2).

As part of training, we learn discrete speech units [53, 32, 7, 18] via a gumbel softmax [24, 5] to represent the latent representations in the contrastive task (Figure 1) which we find to be more effective than non-quantized targets. After pre-training on unlabeled speech, the model is fine-tuned

[1] Code and models are available at `https://github.com/pytorch/fairseq`

Preprint. Under review.

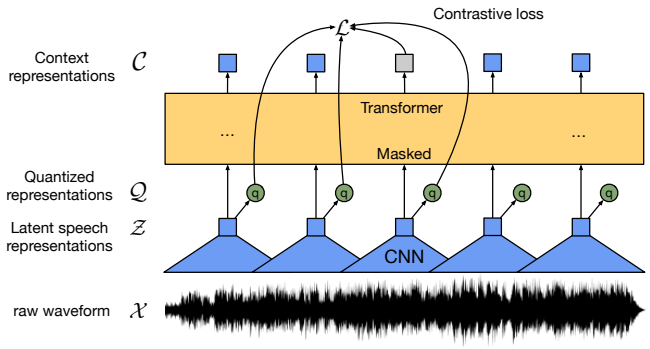

Figure 1: Illustration of our framework which jointly learns contextualized speech representations and an inventory of discretized speech units.

on labeled data with a Connectionist Temporal Classification (CTC) loss [14, 4] to be used for downstream speech recognition tasks (§ 3)

Previous work learned a quantization of the data followed by a contextualized representations with a self-attention model [5, 4], whereas our approach solves both problems end-to-end. Masking parts of the input with Transformer networks for speech has been explored [4, 26], but prior work relies either on a two-step pipeline or their model is trained by reconstructing the filter bank input features. Other related work includes learning representations from auto-encoding the input data [52, 11] or directly predicting future timesteps [8].

Our results show that jointly learning discrete speech units with contextualized representations achieves substantially better results than fixed units learned in a prior step [4]. We also demonstrate the feasibility of ultra-low resource speech recognition: when using only 10 minutes of labeled data, our approach achieves word error rate (WER) 4.8/8.2 on the clean/other test sets of Librispeech. We set a new state of the art on TIMIT phoneme recognition as well as the 100 hour clean subset of Librispeech. Moreover, when we lower the amount of labeled data to just one hour, we still outperform the previous state of the art self-training method of [42] while using 100 times less labeled data and the same amount of unlabeled data. When we use all 960 hours of labeled data from Librispeech, then our model achieves 1.8/3.3 WER (§ 4, § 5).

2 Model

Our model is composed of a multi-layer convolutional feature encoder $f : \mathcal{X} \mapsto \mathcal{Z}$ which takes as input raw audio $\mathcal{X}$ and outputs latent speech representations $\mathbf{z}_1, \ldots, \mathbf{z}_T$ for T time-steps. They are then fed to a Transformer $g : \mathcal{Z} \mapsto \mathcal{C}$ to build representations $\mathbf{c}_1, \ldots, \mathbf{c}_T$ capturing information from the entire sequence [9, 5, 4]. The output of the feature encoder is discretized to $\mathbf{q}_t$ with a quantization module $\mathcal{Z} \mapsto \mathcal{Q}$ to represent the targets (Figure 1) in the self-supervised objective (§ 3.2). Compared to vq-wav2vec [5], our model builds context representations over continuous speech representations and self-attention captures dependencies over the entire sequence of latent representations end-to-end.

Feature encoder. The encoder consists of several blocks containing a temporal convolution followed by layer normalization [1] and a GELU activation function [21]. The raw waveform input to the encoder is normalized to zero mean and unit variance. The total stride of the encoder determines the number of time-steps T which are input to the Transformer (§ 4.2).

Contextualized representations with Transformers. The output of the feature encoder is fed to a context network which follows the Transformer architecture [55, 9, 33]. Instead of fixed positional embeddings which encode absolute positional information, we use a convolutional layer similar to [37, 4, 57] which acts as relative positional embedding. We add the output of the convolution followed by a GELU to the inputs and then apply layer normalization.

Quantization module. For self-supervised training we discretize the output of the feature encoder $\mathbf{z}$ to a finite set of speech representations via product quantization [25]. This choice led to good

results in prior work which learned discrete units in a first step followed by learning contextualized representations [5]. Product quantization amounts to choosing quantized representations from multiple codebooks and concatenating them. Given G codebooks, or groups, with V entries $e \in \mathbb{R}^{V \times d/G}$, we choose one entry from each codebook and concatenate the resulting vectors $e_1, \ldots, e_G$ and apply a linear transformation $\mathbb{R}^d \mapsto \mathbb{R}^f$ to obtain $\mathbf{q} \in \mathbb{R}^f$.

The Gumbel softmax enables choosing discrete codebook entries in a fully differentiable way [16, 24, 35]. We use the straight-through estimator [26] and setup G hard Gumbel softmax operations [24]. The feature encoder output $\mathbf{z}$ is mapped to $\mathbf{l} \in \mathbb{R}^{G \times V}$ logits and the probabilities for choosing the v-th codebook entry for group g are

$$p_{g,v} = \frac{\exp(l_{g,v} + n_v)/\tau}{\sum_{k=1}^{V} \exp(l_{g,k} + n_k)/\tau}, \tag{1}$$

where τ is a non-negative temperature, $n = -\log(-\log(u))$ and u are uniform samples from $\mathcal{U}(0, 1)$. During the forward pass, codeword i is chosen by $i = \text{argmax}_j p_{g,j}$ and in the backward pass, the true gradient of the Gumbel softmax outputs is used.

3 Training

To pre-train the model we mask a certain proportion of time steps in the latent feature encoder space (§ 3.1), similar to masked language modeling in BERT [9]. The training objective requires identifying the correct quantized latent audio representation in a set of distractors for each masked time step (§ 3.2) and the final model is fine-tuned on the labeled data (§ 3.3).

3.1 Masking

We mask a proportion of the feature encoder outputs, or time steps before feeding them to the context network and replace them with a trained feature vector shared between all masked time steps; we do not mask inputs to the quantization module. To mask the latent speech representations output by the encoder, we randomly sample without replacement a certain proportion p of all time steps to be starting indices and then mask the subsequent M consecutive time steps from every sampled index; spans may overlap.

3.2 Objective

During pre-training, we learn representations of speech audio by solving a contrastive task $\mathcal{L}_m$ which requires to identify the true quantized latent speech representation for a masked time step within a set of distractors. This is augmented by a codebook diversity loss $\mathcal{L}_d$ to encourage the model to use the codebook entries equally often.

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d \tag{2}$$

where α is a tuned hyperparameter.

Contrastive Loss. Given context network output $\mathbf{c}_t$ centered over masked time step t, the model needs to identify the true quantized latent speech representation $\mathbf{q}_t$ in a set of $K + 1$ quantized candidate representations $\tilde{\mathbf{q}} \in \mathbf{Q}_t$ which includes $\mathbf{q}_t$ and K distractors [23, 54]. Distractors are uniformly sampled from other masked time steps of the same utterance. The loss is defined as

$$\mathcal{L}_m = -\log \frac{\exp(sim(\mathbf{c}_t, \mathbf{q}_t)/\kappa)}{\sum_{\tilde{\mathbf{q}} \sim \mathbf{Q}_t} \exp(sim(\mathbf{c}_t, \tilde{\mathbf{q}})/\kappa)} \tag{3}$$

where we compute the cosine similarity $sim(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{b} / \|\mathbf{a}\| \|\mathbf{b}\|$ between context representations and quantized latent speech representations [19, 6].

Diversity Loss. The contrastive task depends on the codebook to represent both positive and negative examples and the diversity loss $\mathcal{L}_d$ is designed to increase the use of the quantized codebook representations [10]. We encourage the equal use of the V entries in each of the G codebooks by maximizing the entropy of the averaged softmax distribution $\mathbf{l}$ over the codebook entries for each

codebook $\bar{p}_g$ across a batch of utterances; the softmax disribution does not contain the gumbel noise nor a temperature:[2]

$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v} \tag{4}$$

3.3 Fine-tuning

Pre-trained models are fine-tuned for speech recognition by adding a randomly initialized linear projection on top of the context network into C classes representing the vocabulary of the task [4]. For Librispeech, we have 29 tokens for character targets plus a word boundary token. Models are optimized by minimizing a CTC loss [14] and we apply a modified version of SpecAugment [41] by masking to time-steps and channels during training which delays overfitting and significantly improves the final error rates, especially on the Libri-light subsets with few labeled examples.

4 Experimental Setup

4.1 Datasets

As unlabeled data we consider the Librispeech corpus [40] without transcriptions containing 960 hours of audio (LS-960) or the audio data from LibriVox (LV-60k). For the latter we follow the pre-processing of [27] resulting in 53.2k hours of audio. We fine-tune on five labeled data settings: 960 hours of transcribed Librispeech, the train-clean-100 subset comprising 100 hours (100 hours labeled), as well as the Libri-light limited resource training subsets originally extracted from Librispeech, these are train-10h (10 hours labeled), train-1h (1 hour labeled), train-10min (10 min labeled). We follow the evaluation protocol of Libri-light for these splits and evaluate on the standard Librispech dev-other/clean and test-clean/other sets.

We fine-tune the pre-trained models for phoneme recognition on the TIMIT dataset [13]. It contains five hours of audio recordings with detailed phoneme labels. We use the standard train, dev and test split and follow the standard protocol of collapsing phone labels to 39 classes.

4.2 Pre-training

Models are implemented in fairseq [39]. For masking, we sample $p = 0.065$ of all time-steps to be starting indices and mask the subsequent $M = 10$ time-steps. This results in approximately 49% of all time steps to be masked with a mean span length of 14.7, or 299ms (see Appendix A for more details on masking).

The feature encoder contains seven blocks and the temporal convolutions in each block have 512 channels with strides (5,2,2,2,2,2,2) and kernel widths (10,3,3,3,3,2,2). This results in an encoder output frequency of 49 hz with a stride of about 20ms between each sample, and a receptive field of 400 input samples or 25ms of audio. The convolutional layer modeling relative positional embeddings has kernel size 128 and 16 groups.

We experiment with two model configurations which use the same encoder architecture but differ in the Transformer setup: BASE contains 12 transformer blocks, model dimension 768, inner dimension (FFN) 3,072 and 8 attention heads. Batches are built by cropping 250k audio samples, or 15.6sec, from each example. Crops are batched together to not exceed 1.4m samples per GPU and we train on a total of 64 V100 GPUs for 1.6 days [38]; the total batch size is 1.6h.

The LARGE model contains 24 transformer blocks with model dimension 1,024, inner dimension 4,096 and 16 attention heads. We crop 320k audio samples, or 20sec, with a limit of 1.2m samples per GPU and train on 128 V100 GPUs over 2.3 days for Librispeech and 5.2 days for LibriVox; the total batch size is 2.7h. We use dropout 0.1 in the Transformer, at the output of the feature encoder and the input to the quantization module. Layers are dropped at a rate of 0.05 for BASE and 0.2 for LARGE [22, 12]; there is no layer drop for LV-60k.

[2] Our implementation maximizes perplexity $\frac{GV - \sum_{g=1}^{G} \exp(-\sum_{v=1}^{V} p_{gv} \log p_{gv})}{GV}$ which is equivalent.

We optimize with Adam [29], warming up the learning rate for the first 8% of updates to a peak of 5×10^{-4} for BASE and 3×10^{-4} for LARGE, and then linearly decay it. LARGE trains for 250k updates, BASE for 400k updates, and LARGE on LV-60k for 600k updates. We use weight $\alpha = 0.1$ for the diversity loss Equation 2. For the quantization module we use $G = 2$ and $V = 320$ for both models, resulting in a theoretical maximum of 102.4k codewords. Entries are of size $d/G = 128$ for BASE amd $d/G = 384$ for LARGE. The Gumbel softmax temperature τ is annealed from 2 to a minimum of 0.5 for BASE and 0.1 for LARGE by a factor of 0.999995 at every update. The temperature in the contrastive loss (Equation 3) is set to $\kappa = 0.1$. For the smaller Librispeech dataset, we regularize the model by applying an L2 penalty to the activations of the final layer of the feature encoder and scale down the gradients for the encoder by a factor of 10. We also use a slightly different encoder architecture where we do not use layer normalization, and instead of normalizing the raw waveform, the output of the first encoder layer is normalized. In the contrastive loss we use $K = 100$ distractors. We choose the training checkpoint with the lowest $\mathcal{L}_m$ on the validation set.

4.3 Fine-tuning

After pre-training we fine-tune the learned representations on labeled data and add a randomly initialized output layer on top of the Transformer to predict characters (Librispeech/Libri-light) or phonemes (TIMIT). For Libri-light, we train three seeds with two different learning rates (2e-5 and 3e-5) for all subsets and choose the configuration with lowest WER on dev-other subset decoded with the official 4-gram language model (LM) with beam 50 and fixed model weights (LM weight 2, word insertion penalty -1). For BASE on the labeled 960h subset we use a learning rate of 1e-4.

We optimize with Adam and a tri-state rate schedule where the learning rate is warmed up for the first 10% of updates, held constant for the next 40% and then linearly decayed for the remainder. BASE uses a batch size of 3.2m samples per GPU and we fine-tune on 8 GPUs, giving a total batch size of 1,600sec. LARGE batches 1.28m samples on each GPU and we fine-tune on 24 GPUs, resulting in an effective batch size of 1,920sec. For the first 10k updates only the output classifier is trained, after which the Transformer is also updated. The feature encoder is not trained during fine-tuning. We mask the feature encoder representations with a strategy similar to SpecAugment [41] detailed in Appendix B.

4.4 Language Models and Decoding

We consider two types of language models (LM): a 4-gram model and a Transformer [3] trained on the Librispeech LM corpus. The Transformer LM is identical to [51] and contains 20 blocks, model dimension 1,280, inner dimension 6,144 and 16 attention heads. We tune the weights of the language model (interval $[0, 5]$) and a word insertion penalty ($[-5, 5]$) via Bayesian optimization[3]: we run 128 trials with beam 500 for the 4-gram LM and beam 50 for the Transformer LM and choose the best set of weights according to performance on dev-other. Test performance is measured with beam 1,500 for the n-gram LM and beam 500 for the Transformer LM. We use the beam search decoder of [44].

5 Results

5.1 Low-Resource Labeled Data Evaluation

We first evaluate our pre-trained models in settings where the amount of labeled data is limited to get a sense of how the representations learned on unlabeled data can improve low resource settings. If a pre-trained model captures the structure of speech, then it should require few labeled examples to fine-tune it for speech recognition. The models are pre-trained on the audio data of either Librispeech (LS-960) or LibriVox (LV-60k) and most results are obtained by decoding with a Transformer language model (Transf.); Appendix C shows results with no language model at all as well as with an n-gram language model.

The LARGE model pre-trained on LV-60k and fine-tuned on only 10 minutes of labeled data achieves a word error rate of 5.2/8.6 on the Librispeech clean/other test sets. Ten minutes of labeled data corresponds to just 48 recordings with an average length of 12.5 seconds. This demonstrates that ultra-low resource speech recognition is possible with self-supervised learning on unlabeled data.

[3] `https://github.com/facebook/Ax`

Table 1: WER on the Librispeech dev/test sets when training on the Libri-light low-resource labeled data setups of 10 min, 1 hour, 10 hours and the clean 100h subset of Librispeech. Models use either the audio of Librispeech (LS-960) or the larger LibriVox (LV-60k) as unlabeled data. We consider two model sizes: BASE (95m parameters) and LARGE (317m parameters). Prior work used 860 unlabeled hours (LS-860) but the total with labeled data is 960 hours and comparable to our setup.

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
10 min labeled						
Discrete BERT [4]	LS-960	4-gram	15.7	24.1	16.3	25.2
BASE	LS-960	4-gram	8.9	15.7	9.1	15.6
		Transf.	6.6	13.2	6.9	12.9
LARGE	LS-960	Transf.	6.6	10.6	6.8	10.8
	LV-60k	Transf.	4.6	7.9	4.8	8.2
1h labeled						
Discrete BERT [4]	LS-960	4-gram	8.5	16.4	9.0	17.6
BASE	LS-960	4-gram	5.0	10.8	5.5	11.3
		Transf.	3.8	9.0	4.0	9.3
LARGE	LS-960	Transf.	3.8	7.1	3.9	7.6
	LV-60k	Transf.	2.9	5.4	2.9	5.8
10h labeled						
Discrete BERT [4]	LS-960	4-gram	5.3	13.2	5.9	14.1
Iter. pseudo-labeling [58]	LS-960	4-gram+Transf.	23.51	25.48	24.37	26.02
	LV-60k	4-gram+Transf.	17.00	19.34	18.03	19.92
BASE	LS-960	4-gram	3.8	9.1	4.3	9.5
		Transf.	2.9	7.4	3.2	7.8
LARGE	LS-960	Transf.	2.9	5.7	3.2	6.1
	LV-60k	Transf.	2.4	4.8	2.6	4.9
100h labeled						
Hybrid DNN/HMM [34]	-	4-gram	5.0	19.5	5.8	18.6
TTS data augm. [30]	-	LSTM			4.3	13.5
Discrete BERT [4]	LS-960	4-gram	4.0	10.9	4.5	12.1
Iter. pseudo-labeling [58]	LS-860	4-gram+Transf.	4.98	7.97	5.59	8.95
	LV-60k	4-gram+Transf.	3.19	6.14	3.72	7.11
Noisy student [42]	LS-860	LSTM	3.9	8.8	4.2	8.6
BASE	LS-960	4-gram	2.7	7.9	3.4	8.0
		Transf.	2.2	6.3	2.6	6.3
LARGE	LS-960	Transf.	2.1	4.8	2.3	5.0
	LV-60k	Transf.	1.9	4.0	2.0	4.0

Our approach of jointly learning discrete units and contextualized representations clearly improves over previous work which learned quantized audio units in a separate step [4], reducing WER by a about a third.

A recent iterative self-training approach [42] represents the state of the art on the clean 100 hour subset of Librispeech but it requires multiple iterations of labeling, filtering, and re-training. Our approach is simpler: we pre-train on the unlabeled data and fine-tune on the labeled data. On the 100 hour subset of Librispeech, their method achieves WER 4.2/8.6 on test-clean/other which compares to WER 2.3/5.0 with the LARGE model in a like for like setup, a relative WER reduction of 45%/42%.

When the LARGE model uses an order of magnitude less labeled data (10h labeled), then it still achieves WER 3.2/6.1, an error reduction of 24%/29% relative to iterative self-training. Using only a single hour of labeled data, the same model achieves WER 3.9/7.6 which improves on both test-clean and test-other by 7%/12% - with two orders of magnitude less labeled data. We note that the Libri-

Table 2: WER on Librispeech when using all 960 hours of labeled data (cf. Table 1).

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
Supervised						
CTC Transf [51]	-	CLM+Transf.	2.20	4.94	2.47	5.45
S2S Transf. [51]	-	CLM+Transf.	2.10	4.79	2.33	5.17
Transf. Transducer [60]	-	Transf.	-	-	2.0	4.6
ContextNet [17]	-	LSTM	1.9	3.9	1.9	4.1
Conformer [15]	-	LSTM	2.1	4.3	1.9	3.9
Semi-supervised						
CTC Transf. + PL [51]	LV-60k	CLM+Transf.	2.10	4.79	2.33	4.54
S2S Transf. + PL [51]	LV-60k	CLM+Transf.	2.00	3.65	2.09	4.11
Iter. pseudo-labeling [58]	LV-60k	4-gram+Transf.	1.85	3.26	2.10	4.01
Noisy student [42]	LV-60k	LSTM	1.6	3.4	1.7	3.4
This work						
LARGE - from scratch	-	Transf.	1.7	4.3	2.1	4.6
BASE	LS-960	Transf.	1.8	4.7	2.1	4.8
LARGE	LS-960	Transf.	1.7	3.9	2.0	4.1
	LV-60k	Transf.	1.6	3.0	1.8	3.3

light data splits contain both clean and noisy data leading to better accuracy on test-other compared to test-clean. Increasing model size reduces WER on all setups with the largest improvements on test-other (BASE vs. LARGE both on LS-960) and increasing the amount of unlabeled training data also leads to large improvements (LARGE LS-960 vs. LV-60k).

5.2 High-Resource Labeled Data Evaluation on Librispeech

In this section we evaluate the performance when large quantities of labeled speech are available to assess the effectiveness of our approach in a high resource setup. Specifically, we fine-tune the same models as before on the full 960 hours of labeled Librispeech: BASE and LARGE pre-trained on LS-960 as well as LARGE pre-trained on LV-60k.

Table 2 shows that our approach achieves WER 1.8/3.3 on test-clean/other on the full Librispeech benchmark. This is despite a weaker baseline architecture: supervised training of our architecture achieves WER 2.1/4.6 (LARGE - from scratch) compared to WER 1.9/4.1 for ContextNet [17], the baseline architecture of the state of the art [42]. We use a simple Transformer with CTC which does not perform as well as seq2seq models [51].

Note that the vocabulary of our acoustic model (characters) does not match the vocabulary of the LM (words) which delays feedback from the LM and is likely to be detrimental. Most recent work [51, 58, 17, 42] uses the better performing word pieces [50] for both models. Moreover, our result is achieved without any data balancing such as [42]. Finally, self-training is likely complimentary to pre-training and their combination may yield even better results. Appendix E presents a detailed error analysis of our pre-trained models in various labeled data setups.

5.3 Phoneme Recognition on TIMIT

Next, we evaluate accuracy on TIMIT phoneme recognition by fine-tuning the pre-trained models on the labeled TIMIT training data. We fine-tune as for the 10 hour subset of Libri-light but do not use a language model. Table 3 shows that our approach can achieve a new state of the art on this dataset, reducing PER by a relative 23%/29% over the next best result on the dev/test sets. Appendix D shows an analysis of how the discrete latent speech representations related to phonemes. Other recent work on pre-training which evaluates on TIMIT includes [47] who solve multiple tasks to learn good representations of speech.

Table 3: TIMIT phoneme recognition accuracy in terms of phoneme error rate (PER).

	dev PER	test PER
CNN + TD-filterbanks [59]	15.6	18.0
PASE+ [47]	-	17.2
Li-GRU + fMLLR [46]	–	14.9
wav2vec [49]	12.9	14.7
vq-wav2vec [5]	9.6	11.6
This work (no LM)		
LARGE (LS-960)	7.4	8.3

Table 4: Average WER and standard deviation on combined dev-clean/other of Librispeech for three training seeds. We ablate quantizing the context network input and the targets in the contrastive loss.

	avg. WER	std.
Continuous inputs, quantized targets (Baseline)	7.97	0.02
Quantized inputs, quantized targets	12.18	0.41
Quantized inputs, continuous targets	11.18	0.16
Continuous inputs, continuous targets	8.58	0.08

5.4 Ablations

A difference to previous work [5, 4] is that we quantize the latent audio representations only for the contrastive loss, i.e., when latents are used as *targets*, but not when the latents are *input* to the Transformer network. We motivate this choice by an ablating for which we adopt a reduced training setup to increase experimental turn around: we pre-train BASE on LS-960 for 250k updates with masking probability $p = 0.075$, fine-tune on train-10h for 60k updates on a single GPU with 640k samples per batch, or 40 sec of speech audio. We report the average WER and standard deviation on the concatenation of dev-clean and dev-other (dev PER) for three seeds of fine-tuning.

Table 4 shows that our strategy of continuous inputs with quantized targets (Baseline) performs best. Continuous latent speech representations retain more information to enable better context representations and quantizing the target representations leads to more robust training. Quantizing the latents both in the input and the targets performs least well, and explains the lower performance of prior work [5, 4]. Continuous targets reduce the effectiveness of self-supervised training since targets can capture detailed artifacts of the current sequence, e.g. speaker and background information, which make the task easier and prevent the model from learning general representations beneficial to speech recognition. The training accuracy of identifying the correct latent audio representation increases from 62% to 78.0% when switching from quantized to continuous targets. Continuous inputs and continuous targets perform second best but various attempts to improve it did not lead to better results (see Appendix F for this experiment and other ablations on various hyperparameters).

6 Conclusion

We presented wav2vec 2.0, a framework for self-supervised learning of speech representations which masks latent representations of the raw waveform and solves a contrastive task over quantized speech representations. Our experiments show the large potential of pre-training on unlabeled data for speech processing: when using only 10 minutes of labeled training data, or 48 recordings of 12.5 seconds on average, we achieve a WER of 4.8/8.2 on test-clean/other of Librispeech.

Our model achieves results which achieve a new state of the art on the full Librispeech benchmark for noisy speech. On the clean 100 hour Librispeech setup, wav2vec 2.0 outperforms the previous best result while using 100 times less labeled data. The approach is also effective when large amounts of labeled data are available. We expect performance gains by switching to a seq2seq architecture and a word piece vocabulary.

Broader Impact

There are around 7,000 languages in the world and many more dialects. However, for most of them no speech recognition technology exists since current systems require hundreds or thousands of hours of labeled data which is hard to collect for most languages. We have shown that speech recognition models can be built with very small amounts of annotated data at very good accuracy. We hope our work will make speech recognition technology more broadly available to many more languages and dialects.

Acknowledgments

We thank Tatiana Likhomanenko and Qiantong Xu for helpful discussion and their help with wav2letter integration.

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv*, 2016.

[2] P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning representations by maximizing mutual information across views. In *Proc. of NeurIPS*, 2019.

[3] A. Baevski and M. Auli. Adaptive input representations for neural language modeling. In *Proc. of ICLR*, 2018.

[4] A. Baevski, M. Auli, and A. Mohamed. Effectiveness of self-supervised pre-training for speech recognition. *arXiv*, abs/1911.03912, 2019.

[5] A. Baevski, S. Schneider, and M. Auli. vq-wav2vec: Self-supervised learning of discrete speech representations. In *Proc. of ICLR*, 2020.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv*, abs/2002.05709, 2020.

[7] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord. Unsupervised speech representation learning using wavenet autoencoders. *arXiv*, abs/1901.08810, 2019.

[8] Y. Chung, W. Hsu, H. Tang, and J. R. Glass. An unsupervised autoregressive model for speech representation learning. *arXiv*, abs/1904.03240, 2019.

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

[10] S. Dieleman, A. van den Oord, and K. Simonyan. The challenge of realistic music generation: modelling raw audio at scale. *arXiv*, 2018.

[11] R. Eloff, A. Nortje, B. van Niekerk, A. Govender, L. Nortje, A. Pretorius, E. Van Biljon, E. van der Westhuizen, L. van Staden, and H. Kamper. Unsupervised acoustic unit discovery for speech synthesis using discrete latent-variable neural networks. *arXiv*, abs/1904.07556, 2019.

[12] A. Fan, E. Grave, and A. Joulin. Reducing transformer depth on demand with structured dropout. In *Proc. of ICLR*, 2020.

[13] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CDROM. *Linguistic Data Consortium*, 1993.

[14] A. Graves, S. Fernández, and F. Gomez. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proc. of ICML*, 2006.

[15] A. Gulati, J. Qin, C.-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*, 2020.

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

[17] W. Han, Z. Zhang, Y. Zhang, J. Yu, C.-C. Chiu, J. Qin, A. Gulati, R. Pang, and Y. Wu. Contextnet: Improving convolutional neural networks for automatic speech recognition with global context. *arXiv*, 2020.

[18] D. Harwath, W.-N. Hsu, and J. Glass. Learning hierarchical discrete linguistic units from visually-grounded speech. In *Proc. of ICLR*, 2020.

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

[20] O. J. Hénaff, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv*, abs/1905.09272, 2019.

[21] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv*, 2016.

[22] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. Deep networks with stochastic depth. *arXiv*, 2016.

[23] M. G. A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. of AISTATS*, 2010.

[24] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv*, abs/1611.01144, 2016.

[25] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, Jan. 2011.

[26] D. Jiang, X. Lei, W. Li, N. Luo, Y. Hu, W. Zou, and X. Li. Improving transformer-based speech recognition using unsupervised pre-training. *arXiv*, abs/1910.09932, 2019.

[27] J. Kahn et al. Libri-light: A benchmark for asr with limited or no supervision. In *Proc. of ICASSP*, 2020.

[28] K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. van den Oord. Learning robust and multilingual speech representations. *arXiv*, 2020.

[29] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proc. of ICLR*, 2015.

[30] A. Laptev, R. Korostik, A. Svischev, A. Andrusenko, I. Medennikov, and S. Rybin. You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation. *arXiv*, abs/2005.07157, 2020.

[31] M. P. Lewis, G. F. Simon, and C. D. Fennig. Ethnologue: Languages of the world, nineteenth edition. Online version: `http://www.ethnologue.com`, 2016.

[32] A. H. Liu, T. Tu, H. yi Lee, and L. shan Lee. Towards unsupervised speech recognition and synthesis with quantized speech representation learning. *arXiv*, 2019.

[33] 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 preprint arXiv:1907.11692*, 2019.

[34] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney. Rwth asr systems for librispeech: Hybrid vs attention. In *Interspeech 2019*, 2019.

[35] C. J. Maddison, D. Tarlow, and T. Minka. A* sampling. In *Advances in Neural Information Processing Systems*, pages 3086–3094, 2014.

[36] I. Misra and L. van der Maaten. Self-supervised learning of pretext-invariant representations. *arXiv*, 2019.

[37] A. Mohamed, D. Okhonko, and L. Zettlemoyer. Transformers with convolutional context for ASR. *arXiv*, abs/1904.11660, 2019.

[38] M. Ott, S. Edunov, D. Grangier, and M. Auli. Scaling neural machine translation. In *Proc. of WMT*, 2018.

[39] 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 *Proc. of NAACL System Demonstrations*, 2019.

[40] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *Proc. of ICASSP*, pages 5206–5210. IEEE, 2015.

[41] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. In *Proc. of Interspeech*, 2019.

[42] D. S. Park, Y. Zhang, Y. Jia, W. Han, C.-C. Chiu, B. Li, Y. Wu, and Q. V. Le. Improved noisy student training for automatic speech recognition. *arXiv*, abs/2005.09629, 2020.

[43] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proc. of ACL*, 2018.

[44] V. Pratap, A. Hannun, Q. Xu, J. Cai, J. Kahn, G. Synnaeve, V. Liptchinsky, and R. Collobert. Wav2letter++: A fast open-source speech recognition system. In *Proc. of ICASSP*, 2019.

[45] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. `https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf`, 2018.

[46] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio. Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):92–102, 2018.

[47] M. Ravanelli, J. Zhong, S. Pascual, P. Swietojanski, J. Monteiro, J. Trmal, and Y. Bengio. Multi-task self-supervised learning for robust speech recognition. *arXiv*, 2020.

[48] M. Rivière, A. Joulin, P.-E. Mazaré, and E. Dupoux. Unsupervised pretraining transfers well across languages. *arXiv*, abs/2002.02848, 2020.

[49] S. Schneider, A. Baevski, R. Collobert, and M. Auli. wav2vec: Unsupervised pre-training for speech recognition. In *Proc. of Interspeech*, 2019.

[50] M. Schuster and K. Nakajima. Japanese and korean voice search. In *Proc. of ICASSP*, 2012.

[51] G. Synnaeve, Q. Xu, J. Kahn, T. Likhomanenko, E. Grave, V. Pratap, A. Sriram, V. Liptchinsky, and R. Collobert. End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures. *arXiv*, abs/1911.08460, 2020.

[52] A. Tjandra, B. Sisman, M. Zhang, S. Sakti, H. Li, and S. Nakamura. Vqvae unsupervised unit discovery and multi-scale code2spec inverter for zerospeech challenge 2019. *arXiv*, 1905.11449, 2019.

[53] A. van den Oord, O. Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

[54] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv*, abs/1807.03748, 2018.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. of NIPS*, 2017.

[56] W. Wang, Q. Tang, and K. Livescu. Unsupervised pre-training of bidirectional speech encoders via masked reconstruction. *arXiv*, 2020.

[57] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli. Pay less attention with lightweight and dynamic convolutions. In *Proc. of ICLR*, 2019.

[58] Q. Xu, T. Likhomanenko, J. Kahn, A. Hannun, G. Synnaeve, and R. Collobert. Iterative pseudo-labeling for speech recognition. *arXiv*, 2020.

[59] N. Zeghidour, N. Usunier, I. Kokkinos, T. Schaiz, G. Synnaeve, and E. Dupoux. Learning filterbanks from raw speech for phone recognition. In *Proc. of ICASSP*, 2018.

[60] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. *arXiv*, 2020.

Appendices

A Masking distribution

When choosing which time-steps to mask, each latent speech representation in an utterance is considered a candidate starting time-step with probability p where M is the length of each masked span starting from the respective time step; both are hyper-parameters. Sampled starting time steps are expanded to length M and spans can overlap.

For a 15 sec long audio sample, the average mask length is 14.7 time-steps, corresponding to 299ms of audio, with a median of 10 time-steps, and a maximum of about 100 time steps; about 49% of all time-steps in the sample will be masked. A plot of the corresponding mask length distribution is shown in Figure 2 and an ablation of M and p as well as the effect of other masking strategies is shown in Table 5. Reducing M results in increased prediction accuracy for the self-supervised but the task becomes trivial when spans with length one are masked, leading to poor performance on downstream speech recognition tasks. We also consider other masking strategies: w/o overlap uniform(a,b) samples for each starting index a span length M^s from interval a to b and masks the subsequent M^s time-steps taking care not to overlap with existing spans; poisson(λ) and normal(μ, σ) sample M^s from Poisson and normal distributions.

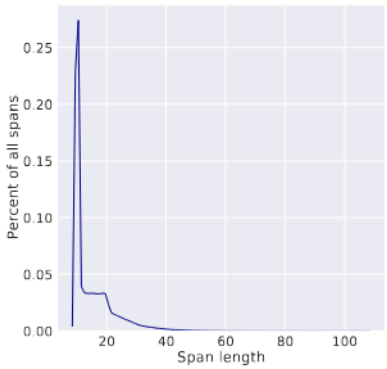

Figure 2: Mask length distribution for a 15 second sample with $p = 0.065$ and $M = 10$.

Table 5: Ablations on settings for the masking strategy during pre-training. When masking without overlap, we choose starting time steps with $p = 0.037$ which results in the total number of masked tokens to match the baseline.

	avg WER	std
Baseline ($p = 0.075$)	7.97	0.02
Mask length $M = 8$	8.33	0.05
Mask length $M = 12$	8.19	0.08
Mask length $M = 15$	8.43	0.19
Mask probability $p = 0.065$	7.95	0.08
Mask probability $p = 0.06$	8.14	0.22
Mask w/o overlap, uniform(1,31)	8.39	0.02
Mask w/o overlap, uniform(10,30)	9.17	0.05
Mask w/o overlap, poisson(15)	8.13	0.04
Mask w/o overlap, normal(15, 10)	8.37	0.03
Mask w/o overlap, length 10	9.15	0.02
Mask w/o overlap, length 15	9.43	0.26

B Fine-tuning Setup

During fine-tuning we apply a masking strategy to the feature encoder outputs similar to SpecAugment [41]: we randomly choose a number of starting time steps for which a span of ten subsequent time-steps is replaced with a mask embedding; spans may overlap and we use the same masked time step embedding as during pre-training. We also mask channels by choosing a number of channels as starting indices and then expand each one to cover the subsequent 64 channels. Spans may overlap and the selected channel spans are set to zero value. We use LayerDrop [22, 12] at a rate of 0.05 for BASE and 0.1 for LARGE during fine-tuning.

Table 6 summarizes the fine-tuning hyper-parameter settings used for the different labeled data setup. Table 7 shows the decoding parameters used for final evaluations of the various labeled data setups for Librispeech pre-trained models and Table 8 shows decoding parameters for LibriVox.

Table 6: Fine-tuning hyperparameters

	timestep mask prob.	channel mask prob.	updates
10 min	0.075	0.008	12k
1 hour	0.075	0.004	13k
10 hours	0.065	0.004	20k
100 hours	0.05	0.008	50k
960 hours	0.05	0.0016	320k
TIMIT	0.065	0.012	40k

Table 7: Decoding parameters for Librispeech subsets for models pre-trained on Librispeech

	4gram LM weight	4gram word insert.	TransLM weight	TransLM word insert.
10 min	3.23	-0.26	1.20	-1.39
1 hour	2.90	-1.62	1.15	-2.08
10 hours	2.46	-0.59	1.06	-2.32
100 hours	2.15	-0.52	0.87	-1.00
960 hours	1.74	0.52	0.92	-0.86

Table 8: Decoding parameters for Librispeech subsets for models pre-trained on Librivox.

	4gram LM weight	4gram word insert.	TransLM weight	TransLM word insert.
10 min	3.86	-1.18	1.47	-2.82
1 hour	3.09	-2.33	1.33	-0.69
10 hours	2.12	-0.90	0.94	-1.05
100 hours	2.15	-0.52	0.87	-1.00
960 hours	1.57	-0.64	0.90	-0.31

C Full results for Libri-light and Librispeech

Table 9: WER on the Librispeech dev/test sets when training on the Libri-light low-resource labeled data setups (cf. Table 1).

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
10 min labeled						
BASE	LS-960	None	46.1	51.5	46.9	50.9
		4-gram	8.9	15.7	9.1	15.6
		Transf.	6.6	13.2	6.9	12.9
LARGE	LS-960	None	43.0	46.3	43.5	45.3
		4-gram	8.6	12.9	8.9	13.1
		Transf.	6.6	10.6	6.8	10.8
LARGE	LV-60k	None	38.3	41.0	40.2	38.7
		4-gram	6.3	9.8	6.6	10.3
		Transf.	4.6	7.9	4.8	8.2
1h labeled						
BASE	LS-960	None	24.1	29.6	24.5	29.7
		4-gram	5.0	10.8	5.5	11.3
		Transf.	3.8	9.0	4.0	9.3
LARGE	LS-960	None	21.6	25.3	22.1	25.3
		4-gram	4.8	8.5	5.1	9.4
		Transf.	3.8	7.1	3.9	7.6
LARGE	LV-60k	None	17.3	20.6	17.2	20.3
		4-gram	3.6	6.5	3.8	7.1
		Transf.	2.9	5.4	2.9	5.8
10h labeled						
BASE	LS-960	None	10.9	17.4	11.1	17.6
		4-gram	3.8	9.1	4.3	9.5
		Transf.	2.9	7.4	3.2	7.8
LARGE	LS-960	None	8.1	12.0	8.0	12.1
		4-gram	3.4	6.9	3.8	7.3
		Transf.	2.9	5.7	3.2	6.1
LARGE	LV-60k	None	6.3	9.8	6.3	10.0
		4-gram	2.6	5.5	3.0	5.8
		Transf.	2.4	4.8	2.6	4.9
100h labeled						
BASE	LS-960	None	6.1	13.5	6.1	13.3
		4-gram	2.7	7.9	3.4	8.0
		Transf.	2.2	6.3	2.6	6.3
LARGE	LS-960	None	4.6	9.3	4.7	9.0
		4-gram	2.3	5.7	2.8	6.0
		Transf.	2.1	4.8	2.3	5.0
LARGE	LV-60k	None	3.3	6.5	3.1	6.3
		4-gram	1.8	4.5	2.3	4.6
		Transf.	1.9	4.0	2.0	4.0

Table 10: WER on Librispeech when using all 960 hours of Librispeech as labeled data (cf. Table 2).

Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
LARGE - from scratch	-	None	2.8	7.6	3.0	7.7
	-	4-gram	1.8	5.4	2.6	5.8
	-	Transf.	1.7	4.3	2.1	4.6
BASE	LS-960	None	3.2	8.9	3.4	8.5
		4-gram	2.0	5.9	2.6	6.1
		Transf.	1.8	4.7	2.1	4.8
LARGE	LS-960	None	2.6	6.5	2.8	6.3
		4-gram	1.7	4.6	2.3	5.0
		Transf.	1.7	3.9	2.0	4.1
LARGE	LV-60k	None	2.1	4.5	2.2	4.5
		4-gram	1.4	3.5	2.0	3.6
		Transf.	1.6	3.0	1.8	3.3

D Analysis of Discrete Latent Speech Representations

Next, we investigate whether the discrete latent speech representations $\mathbf{q}_t$ learned by the quantizer relate to phonetic information: Using LARGE pre-trained on LV-60k and without any fine-tuning, we compute the discrete latents for the training data of TIMIT and compute the co-occurrence between human annotated phonemes and the latents. Ties are broken by choosing the phoneme which is most represented in the receptive field of $\mathbf{q}_t$. The training data contains 3696 utterances of average length 13.6 sec, or 563k discrete latents.

Figure 3 plots $P(phoneme|\mathbf{q}_t)$ and shows that many discrete latents appear to specialize in specific phonetic sounds. The silence phoneme (bcl) represents 22% of all human annotated speech data and is therefore also modeled by many different latents.

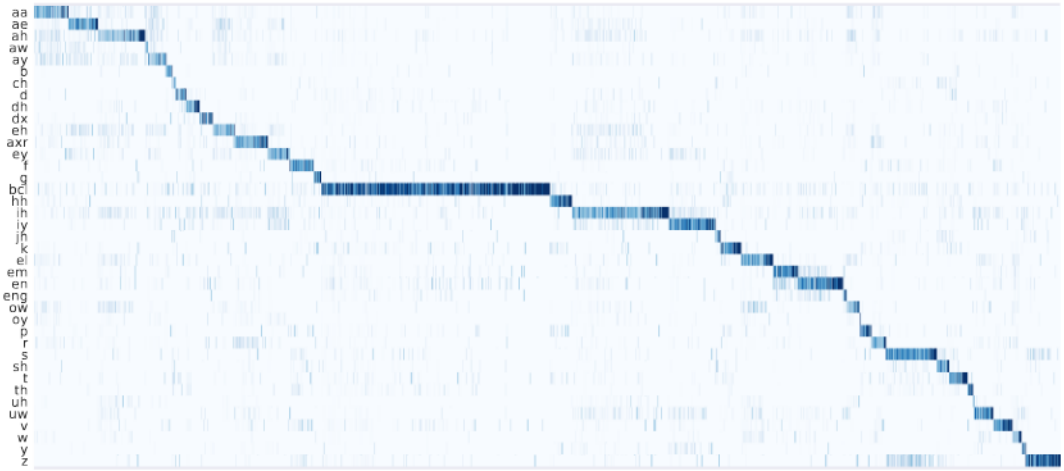

Figure 3: Visualization of the co-occurrence between discrete latent speech representations and phonemes. We plot the conditional probability $P(phoneme|\mathbf{q}_t)$ on TIMIT train data. The y-axis shows the collapsed 39 classes of phonemes and the x-axis is over the different discrete latents.

E Speech Recognition Error Analysis

In this section we study the most common errors our models make when fine-tuned on different amounts of labeled data (Table 11). We also show transcriptions of a few relatively challenging utterances from the dev-clean subset of Librispeech (Table 12).

We consider models with no lexicon or no language model decoding, marked None in Table 9: Larger capacity decreases error rates: LARGE on LS-960 improves the word error rate on dev-clean from 46.1 to 43 compared to BASE. Increasing the amount of unlabeled training data further decreases the error rate to 33.8 for LARGE on LS-960.

In the ten minute labeled data setup, the model is still able to recognize basic units of speech: Table 11 shows that most errors are around spelling of words, e.g., omitting silent characters such as *could* $\rightarrow$ *coud*, *know* $\rightarrow$ *now*, or ignoring repeated letters such as *still* $\rightarrow$ *stil*, *little* $\rightarrow$ *litle*. The LARGE LV-60k model achieves WER 38.3 on dev-clean and adding a Transformer language model enables to choose more likely pronunciations during the search and gives a large WER improvement to 5.0.

The ten minute models without lexicon and language model tend to spell words phonetically and omit repeated letters, e.g., *will* $\rightarrow$ *wil* (Table 11). Spelling errors decrease with more labeled data: with one hour of labeled data, slightly less common words move into the list of the most frequent errors, e.g., *heaven* and *food* are spelled phonetically. At ten hours, top errors include articles, e.g., *a*, *the* which are a common source of errors in speech recognition in general. There are also alternative spellings, *color* vs. *colour* as well as relatively rare words including person names, still spelled phonetically, e.g., *phoebe* $\rightarrow$ *feeby*.

At 100 hours, person names dominate the most frequent errors: *phoebe* $\rightarrow$ *phebe*, along with incorrect spacing *anyone* $\rightarrow$ *any one*, *awhile* $\rightarrow$ *a while*. Finally at 960 hours the word error rate falls to 2% and top errors are mostly articles, incorrect splits, and some very rare words or names such as *deucalion* or *gryce*.

The "from scratch" 960 hour model has a similar word error rate as the 100 hour pre-trained model and displays a similar pattern of errors.

The pre-trained speech representations can be easily adapted to recognize specific sounds while fine-tuning grounds these representations to the actual spelling.

Table 11: Top word errors for models trained on 10m, 1h and 10h, 100h, 960h of labeled data and decoded on the Librispeech dev-clean subset without a language model or lexicon (see Table 9 and Table 10 - None). In brackets is the total number of occurrences of each error.

10m LARGE LV-60k	1h LARGE LV-60k	10h LARGE LV-60k
all → al (181)	too → to (26)	in → and (15)
are → ar (115)	until → untill (24)	a → the (11)
will → wil (100)	new → knew (22)	o → oh (10)
you → yo (90)	door → dor (18)	and → in (9)
one → on (89)	says → sais (18)	mode → mod (9)
two → to (81)	soul → sol (17)	ursus → ersus (9)
well → wel (80)	bread → bred (16)	tom → tome (8)
been → ben (73)	poor → pore (16)	randal → randol (7)
upon → apon (73)	a → the (13)	the → a (7)
good → god (67)	either → ither (13)	color → colour (6)
see → se (66)	food → fud (13)	flour → flower (6)
we → whe (60)	doubt → dout (12)	phoebe → feeby (6)
little → litle (54)	earth → erth (12)	an → and (5)
great → grate (53)	led → lead (12)	cucumbers → cucombers (5)
your → yor (53)	sea → see (12)	egg → eg (5)
could → coud (51)	thee → the (12)	macklewain → macklewaine (5)
here → hear (51)	tom → tome (12)	magpie → magpi (5)
know → now (45)	add → ad (11)	milner → millner (5)
there → ther (45)	good → god (11)	stacy → staci (5)
three → thre (45)	heaven → heven (11)	trevelyan → trevellion (5)
still → stil (42)	mary → marry (11)	verloc → verlock (5)
off → of (40)	randal → randel (11)	ann → an (4)
don't → dont (37)	answered → ansered (10)	anyone → one (4)
shall → shal (36)	blood → blod (10)	apartment → appartment (4)
little → litl (35)	bozzle → bosel (10)	basin → bason (4)

100h LARGE LV-60k	960h LARGE LV-60k	960h LARGE from scratch
a → the (13)	a → the (12)	and → in (20)
and → in (10)	and → in (9)	a → the (16)
in → and (10)	macklewain → mackelwaine (7)	in → and (13)
o → oh (8)	in → and (6)	the → a (10)
minnetaki → minnitaki (7)	o → oh (6)	in → an (8)
randal → randall (7)	bozzle → bosell (5)	and → an (5)
christie → cristy (6)	criss → chris (5)	clarke → clark (4)
macklewain → mackelwane (6)	bozzle → bosel (4)	grethel → gretel (4)
randal → randoll (6)	clarke → clark (4)	macklewain → mackelwaine (4)
bozzle → bosall (5)	colored → coloured (4)	this → the (4)
kaliko → calico (5)	grethel → gretel (4)	an → and (3)
trevelyan → trevelian (5)	lige → lyge (4)	anyone → one (3)
an → and (4)	the → a (4)	bozzle → basell (3)
and → an (4)	and → an (3)	buns → bunds (3)
anyone → one (4)	ann → marianne (3)	carrie → carry (3)
bozzle → bozall (4)	butte → bute (3)	criss → chris (3)
clarke → clark (4)	color → colour (3)	he's → is (3)
gryce → grice (4)	deucalion → ducalion (3)	his → is (3)
i'm → am (4)	forcemeat → meat (3)	honor → honour (3)
in → ind (4)	gryce → grice (3)	lattimer → latimer (3)
letty → lettie (4)	honor → honour (3)	millet → mellet (3)
phoebe → phebe (4)	kearny → kirney (3)	pyncheon → pension (3)
the → a (4)	nuova → noiva (3)	tad → ted (3)
ann → anne (3)	thing → anything (3)	thing → anything (3)
awhile → while (3)	this → the (3)	trevelyan → trevelian (3)

Table 12: Examples of transcription of selected utterances from the dev-clean subset by various models without a language model or lexicon. Capitalized words indicate errors.

Model	Transcription
Reference	i'm mister christopher from london
10m LV-60k	IM mister CRESTIFER FROME LUNDEN
1h LV-60k	IM mister CRISTIFFHER from LOUNDEN
10h LV-60k	i'm mister CHRYSTEPHER from london
100h LV-60k	i'm mister christopher from london
960h LV-60k	i'm mister christopher from london
960h scratch	I MISSTER christopher from london
Reference	il popolo e una bestia
10m LV-60k	ILPOPULAR ONABESTIA
1h LV-60k	O POPOLAONABASTIA
10h LV-60k	U POPULAONABASTIAR
100h LV-60k	O POPALOON A BASTYA
960h LV-60k	YOU'LL POP A LAWYE ON A BAISTYE
960h scratch	OL POPALOY ON ABESTIA
Reference	he smelt the nutty aroma of the spirit
10m LV-60k	he SMELTD the NUDY aroma of the spirit
1h LV-60k	he SMELTD the NUDDY ARROMA of the spirit
10h LV-60k	he smelt the NUDDY ERROMA of the spirit
100h LV-60k	he smelt the NUDDY aroma of the spirit
960h LV-60k	he smelt the NUTTIE aroma of the spirit
960h scratch	he smelt the nutty EROMA of the spirit
Reference	phoebe merely glanced at it and gave it back
10m LV-60k	FEABY MEARLY glanced at it and gave it BAK
1h LV-60k	FIEABY merely glanced at it and gave it back
10h LV-60k	FEEBY merely glanced at it and gave it back
100h LV-60k	BEBE merely glanced at it and gave it back
960h LV-60k	phoebe merely glanced at it and gave it back
960h scratch	phoebe merely glanced at it and gave it back
Reference	sauterne is a white bordeaux a strong luscious wine the best known varieties being
10m LV-60k	SULTERIN is a white BORDOE a strong LUCHOUS WIN the best NOWN VERIATYS being
1h LV-60k	CLTEREN is a white BORDO a strong LUCHIOUS wine the best known VERIETIES being
10h LV-60k	SOTERN is a white BOURDO a strong LUCIOUS wine the best known VORIETIES being
100h LV-60k	SOTERN is a white BORDAUX a strong LUCIOUS wine the best known varieties being
960h LV-60k	SOTERN is a white bordeaux a strong luscious wine the best known varieties being
960h scratch	SOTERAN is a white bordeaux a strong luscious wine the best known varieties being
Reference	i happen to have mac connell's box for tonight or there'd be no chance of our getting places
10m LV-60k	i HAPEND to have MECONALES BOXS for TONIT ORE THIRLD be no chance of OR GETING places
1h LV-60k	i happen to have MACCONNEL'S BOCXS for tonight or TE'ELD be no chance of our getting places
10h LV-60k	i HAPPENED to have MUKONNEL'S box for tonight or THERED be no chance of our getting places
100h LV-60k	i HAPPENED to have MC CONNEL'S box for TO NIGHT or there'd be no chance of our getting places
960h LV-60k	i happen to have MC CONALL'S box for TO NIGHT or there'd be no chance of our getting places
960h scratch	i HAPPENE to have MACONEL'S box for TO NIGHT or there'd be no chance of our getting places

F Ablations

Table 13 ablates various hyperparameter choices of our architecture. The setup for the baseline model is described in § 5.4. First, we tried to improve the continuous input and continuous target model (§ 5.4) by adding an MLP on top of the continuous target representation and we also tried to use a separate set of encoder parameters for the representations used as input and targets (Separate encoders). Both did not lead to meaningful improvements.

Increasing the receptive field size from 25ms to 30ms had little effect. Setting the diversity penalty weight (α) too low results in lower codebook usage and lower performance. Setting it too high leads to slight instability. Doubling the number of relative positional embeddings to 256 also did not help. Stopping gradients from the quantizer to the encoder shows that the encoder requires training signal from the quantizer as well.

Next, increasing the number of negatives did not result in better performance ($K = 200$) and sampling negatives from the entire batch of utterances hurt performance, likely because candidates from other utterances are easy to distinguish. Sampling negatives from any time step in the utterance, masked or unmasked, does not help and is more computationally expensive. Gumbel noise is important and increasing the number of codebooks did not result in better performance.

Table 13: Ablation of various hyper-parmeter choices. We report average WER and standard deviation on combined dev-clean/other of Librispeech for three seeds of training.

	avg. WER	std.
Baseline ($p = 0.075$, $\alpha = 0.1$)	7.97	0.02
Continuous inputs, continuous targets	8.58	0.08
+ MLP on targets	8.51	0.05
+ Separate encoders	8.90	0.01
receptive field 30ms	7.99	0.06
diversity penalty		
$\alpha = 0$	8.48	0.08
$\alpha = 0.05$	8.34	0.08
$\alpha = 0.2$	8.58	0.45
Conv pos emb, kernel 256	8.14	0.05
No gradient to encoder from quantizer	8.41	0.08
Negatives		
$K = 200$ same utterance	8.12	0.05
$K = 50$ same utterance + $K = 50$ from batch	8.79	0.06
Sample negatives from any time step	8.07	0.02
No Gumbel noise	8.73	0.42
Codebook		
G=4, V=18	9.02	0.38
G=8, V=8	8.13	0.07
Predict exactly U time steps from edges		
$U = 1$	9.53	0.91
$U = 5$	8.19	0.07
$U = 10$	8.07	0.07
$U = 15$	7.89	0.10
$U = 20$	7.90	0.01

We also investigated predicting only time steps immediately next to the last unmasked time step for each span. This enables to better control the difficulty of the pre-training task. Given the leftmost or rightmost unmasked time step next to a masked span, we compute the contrastive loss only for the first U masked time steps next to these unsmasked spans. Predicting only up to one time step performs poorly because there is little training signal from each utterance and predicting more time steps performs better but does not significantly outperform predicting all masked time steps. Increasing the number of training updates helps but this increases training time.