Report on pre-trained transformers for speech models

1. HUBERT:

HUBERT stands for Hidden Unit BERT. HUBERT was proposed to deal with three major problems of Self Supervised speech recognition approaches:

a. There are multiple sound units in each input utterance
b. There is no lexicon of input sound units during the pre-training phase
c. Sound units have variable lengths with no explicit segmentation

HUBERT utilizes an offline clustering step to provide aligned target labels for a BERT-like prediction loss.

A key ingredient of this approach is applying the prediction loss over the masked regions only, which forces the model to learn a combined acoustic and language model over the continuous inputs.

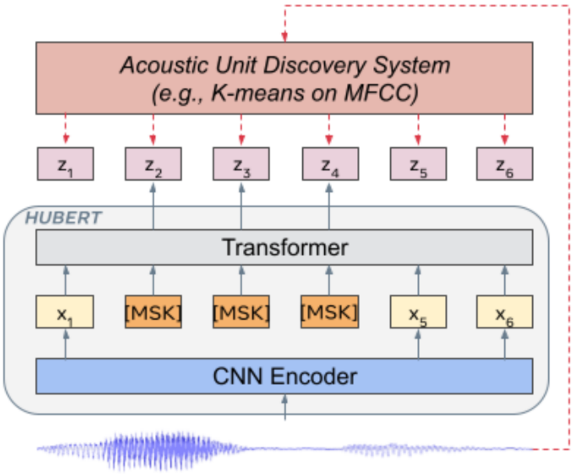


Fig. 1: The HuBERT approach predicts hidden cluster assignments of the masked frames (y_2, y_3, y_4 in the figure) generated by one or more iterations of k-means clustering.

HuBERT relies primarily on the consistency of the unsupervised clustering step rather than the intrinsic quality of the assigned cluster labels.

Starting with a simple k-means teacher of 100 clusters, and using two iterations of clustering, the HuBERT model either matches or improves upon the state-of-the-art wav2vec 2.0 performance on the Librispeech (960h) and Libri-light (60,000h) benchmarks with 10min, 1h, 10h, 100h, and 960h fine-tuning subsets. Using a 1B parameter model, HuBERT shows up to 19% and 13% relative WER reduction on the more challenging dev-other and test-other evaluation subsets.

teacher	C	dev-other WER (%)			
		steps=100k	250k	400k	800k
K-means	50	18.68	13.65	12.40	11.82
	100	17.86	12.97	12.32	11.68
[51]	13.5k	26.6			

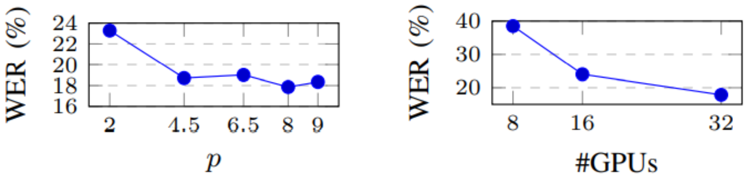


Fig. 3: Varying masking probability p (left) and effective batch size through the number of GPUs (right).

2. SEW

SEW stands for Squeezed and Efficient Wav2Vec. The paper focuses on performance-efficiency trade-offs in pre-trained models for automatic speech recognition (ASR). The focus is on wav2vec 2.0 and formalizing several architectural designs that influence both the model performance and its efficiency. SEW is pre-trained model architecture with significant improvement in both performance and efficiency dimensions across a variety of training setups.

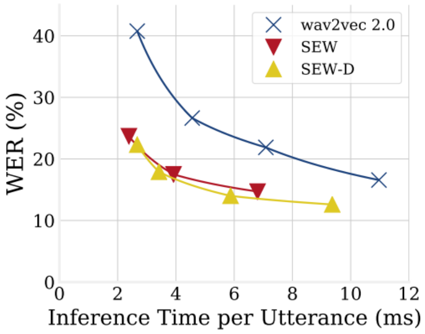


Figure 1: Word error rate (WER) and average utterance inference time on LibriSpeech (dev-other) of wav2vec 2.0 and our SEW and SEW-D models fine-tuned with 100h labeled data for 100K updates.

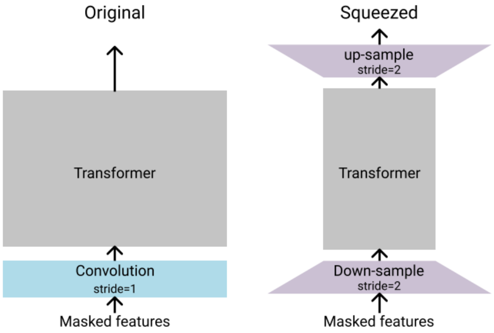


Figure 4: Original vs. squeezed context network. The sequence length is halved by the down-sampling layer.

Model design and trade-off:

Pre-training:

LibriSpeech 960h is used as training data for unsupervised pre-training, leaving 1% as a validation set for pre-training. Hyperparameters used are W2V2-base2 .To speed up and reduce the cost of the experiments, pre-training all models for 100K updates similar to Hsu et al. (2020). 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. (2020b) use 64 GPUs, we set gradient accumulation steps to 8 to simulate their 64-GPU pre-training with 8 GPUs.

Fine-tuning

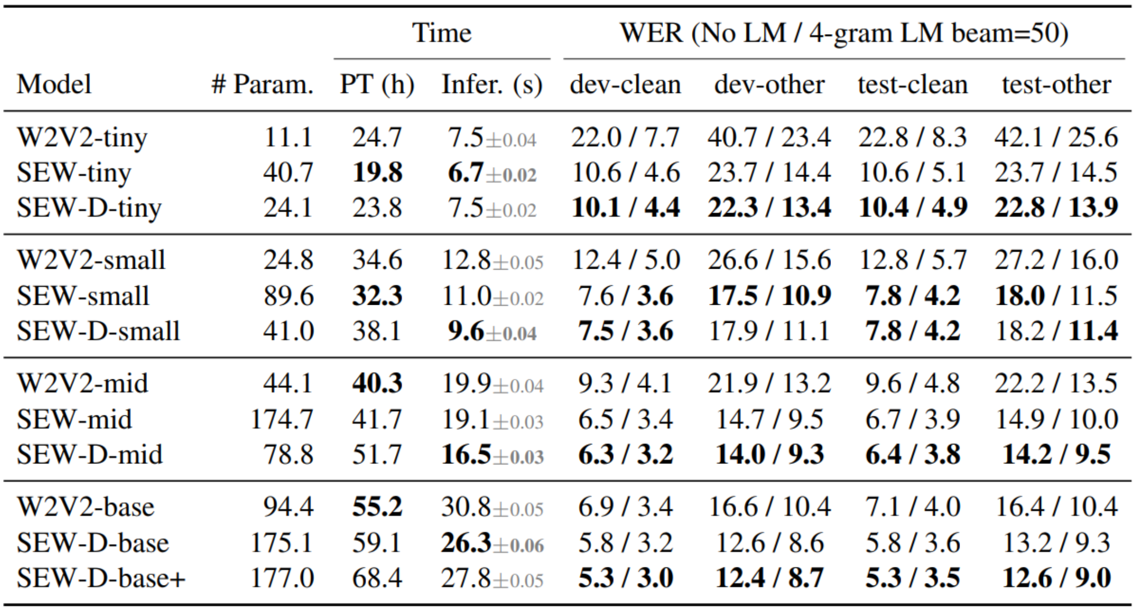
A linear classifier is added to the top of the context network and fine-tune the model using a CTC objective on LibriSpeech train-clean 100h set for 80K updates using the same set of hyper-parameters as W2V2-base

Evaluation:

The use of CTC is used for greedy decoding for all experiments because it is faster than Viterbi decoding (Viterbi, 1967) 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 (word error rate). All evaluation is done on a NVIDIA V100 GPU with FP32 operations, unless specified otherwise. When decoding with a language model (LM), we use the official 4-gram LM4 and wav2letter (Collobert et al., 2016) decoder5 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

Results:

SEW vs. SEW-D vs. W2V2 on LibriSpeech 100h-960h pre-training W2V2, SEW, and SEW-D on 960h LibriSpeech audios for 100K updates and fine-tune them on 100h labelled data



		Time		WER (No LM / 4-gram LM beam=50)			
Model	# Param.	PT (h)	Infer. (s)	dev-clean	dev-other	test-clean	test-other
W2V2-tiny	11.1	24.7	$7.5_{\pm 0.04}$	22.0 / 7.7	40.7 / 23.4	22.8 / 8.3	42.1 / 25.6
SEW-tiny	40.7	**19.8**	$\mathbf{6.7}_{\pm \mathbf{0.02}}$	10.6 / 4.6	23.7 / 14.4	10.6 / 5.1	23.7 / 14.5
SEW-D-tiny	24.1	23.8	$7.5_{\pm 0.02}$	**10.1 / 4.4**	**22.3 / 13.4**	**10.4 / 4.9**	**22.8 / 13.9**
W2V2-small	24.8	34.6	$12.8_{\pm 0.05}$	12.4 / 5.0	26.6 / 15.6	12.8 / 5.7	27.2 / 16.0
SEW-small	89.6	**32.3**	$11.0_{\pm 0.02}$	7.6 / **3.6**	**17.5 / 10.9**	**7.8 / 4.2**	**18.0** / 11.5
SEW-D-small	41.0	38.1	$\mathbf{9.6}_{\pm \mathbf{0.04}}$	**7.5 / 3.6**	17.9 / 11.1	**7.8 / 4.2**	18.2 / **11.4**
W2V2-mid	44.1	**40.3**	$19.9_{\pm 0.04}$	9.3 / 4.1	21.9 / 13.2	9.6 / 4.8	22.2 / 13.5
SEW-mid	174.7	41.7	$19.1_{\pm 0.03}$	6.5 / 3.4	14.7 / 9.5	6.7 / 3.9	14.9 / 10.0
SEW-D-mid	78.8	51.7	$\mathbf{16.5}_{\pm \mathbf{0.03}}$	**6.3 / 3.2**	**14.0 / 9.3**	**6.4 / 3.8**	**14.2 / 9.5**
W2V2-base	94.4	**55.2**	$30.8_{\pm 0.05}$	6.9 / 3.4	16.6 / 10.4	7.1 / 4.0	16.4 / 10.4
SEW-D-base	175.1	59.1	$\mathbf{26.3}_{\pm \mathbf{0.06}}$	5.8 / 3.2	12.6 / 8.6	5.8 / 3.6	13.2 / 9.3
SEW-D-base+	177.0	68.4	$27.8_{\pm 0.05}$	**5.3 / 3.0**	**12.4 / 8.7**	**5.3 / 3.5**	**12.6 / 9.0**

Table 6: Libri-Speech 100h-960h semi-supervised setup pretrained for 100K updates.

Without an LM, compared with the W2V2-tiny, SEW-tiny reduces the WER by 53.5% (22.8% to 10.6%) and 43.7% (41.1% to 23.7%) on test-clean and test-other, while being faster. With an LM, WER improves by 38.6% and 43.4% on test-clean and test-other. Compared with the W2V2-mid, SEW-mid reduces WER by 30.2% (9.6% to 6.7%) and 32.9% (22.2% to 14.9%) with similar inference times. 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 takes longer to train (e.g., SEW-small vs. W2V2-mid or SEW-mid vs. W2V2-base). SEW-D has lower WER compared to SEW even with smaller width and half of the parameters. With large models, SEW-D is also more efficient. However, SEW-D-tiny is slower than SEW-tiny, due to the implementation difference

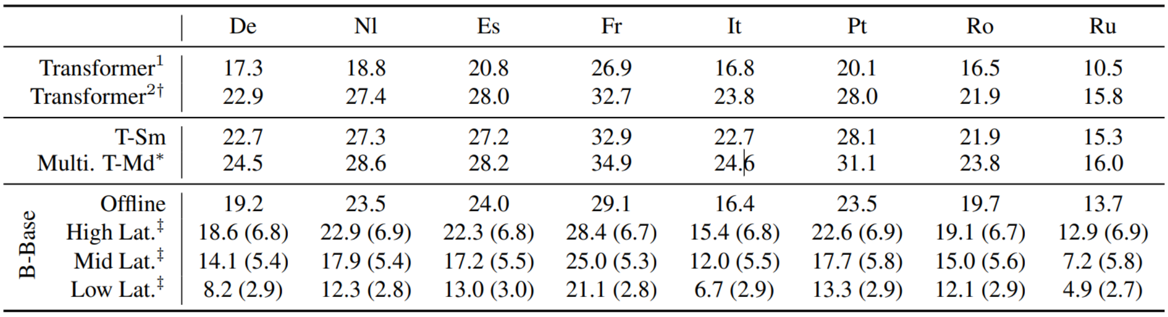
3. FAIRSEQ S2T

FAIRSEQ S2T is an extension for Speech2Text modelling task such as Speech recognition and Speech2Text translation.

Fairseq Models FAIRSEQ provides a collection of MT models and LMs that demonstrate state-of-the-art performance on standard benchmarks. They are open sourced with pre-trained models. FAIRSEQ also supports other tasks such as text summarization, story generation and self-supervised speech pre-training.

S2T extension FAIRSEQ S2T adds attention-based RNN, Transformer models, as well as the latest Conformer models for ASR and ST. It also supports the CTC criterion for ASR. For the simultaneous ST setting, it includes online models with widely used policies: monotonic attention, wait-k (Ma monotonic infinite lookback attention, and monotonic multi-head attention

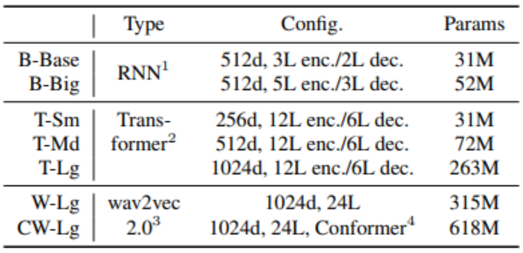
Evaluation Metrics FAIRSEQ S2T provides common automatic metrics for ASR, ST, and MT, including WER (word error rate), BLEU, and chrF It also integrates SIMULEVAL for simultaneous ST/MT metrics such as AL (average lagging) and DAL (differentiable average Lagging)



		De	Nl	Es	Fr	It	Pt	Ro	Ru
	Transformer[1]	17.3	18.8	20.8	26.9	16.8	20.1	16.5	10.5
	Transformer[2†]	22.9	27.4	28.0	32.7	23.8	28.0	21.9	15.8
	T-Sm	22.7	27.3	27.2	32.9	22.7	28.1	21.9	15.3
	Multi. T-Md*	24.5	28.6	28.2	34.9	24.6	31.1	23.8	16.0
B-Base	Offline	19.2	23.5	24.0	29.1	16.4	23.5	19.7	13.7
	High Lat.[‡]	18.6 (6.8)	22.9 (6.9)	22.3 (6.8)	28.4 (6.7)	15.4 (6.8)	22.6 (6.9)	19.1 (6.7)	12.9 (6.9)
	Mid Lat.[‡]	14.1 (5.4)	17.9 (5.4)	17.2 (5.5)	25.0 (5.3)	12.0 (5.5)	17.7 (5.8)	15.0 (5.6)	7.2 (5.8)
	Low Lat.[‡]	8.2 (2.9)	12.3 (2.8)	13.0 (3.0)	21.1 (2.8)	6.7 (2.9)	13.3 (2.9)	12.1 (2.9)	4.9 (2.7)

Table 2: FAIRSEQ S2T models on MuST-C. Test BLEU reported (for online models, AL is shown in parentheses). [1] Di Gangi et al. (2019). [2] Inaguma et al. (2020). [†] Applied additional techniques: speed perturbation, pre-trained decoder from MT and auxiliary CTC loss for ASR pre-training. [‡] Online models using beam size of 1 (instead of 5). [*] Trained jointly on all 8 languages.

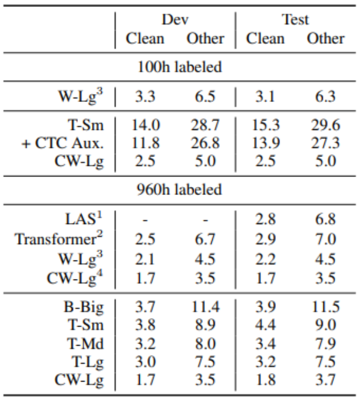
Model Architecture:



	Type	Config.	Params
B-Base	RNN[1]	512d, 3L enc./2L dec.	31M
B-Big		512d, 5L enc./3L dec.	52M
T-Sm	Trans-former[2]	256d, 12L enc./6L dec.	31M
T-Md		512d, 12L enc./6L dec.	72M
T-Lg		1024d, 12L enc./6L dec.	263M
W-Lg	wav2vec 2.0[3]	1024d, 24L	315M
CW-Lg		1024d, 24L, Conformer[4]	618M

Table 3: FAIRSEQ S2T models for benchmarking. For simplicity, we use the same (default) model hyper-parameters and learning rate schedule across all experiments. [1] Bérard et al. (2018). [2] Vaswani et al. (2017). [3] Baevski et al. (2020). [4] Gulati et al. (2020).

Model Performance:



	Dev		Test	
	Clean	Other	Clean	Other
100h labeled				
W-Lg[3]	3.3	6.5	3.1	6.3
T-Sm	14.0	28.7	15.3	29.6
+ CTC Aux.	11.8	26.8	13.9	27.3
CW-Lg	2.5	5.0	2.5	5.0
960h labeled				
LAS[1]	-	-	2.8	6.8
Transformer[2]	2.5	6.7	2.9	7.0
W-Lg[3]	2.1	4.5	2.2	4.5
CW-Lg[4]	1.7	3.5	1.7	3.5
B-Big	3.7	11.4	3.9	11.5
T-Sm	3.8	8.9	4.4	9.0
T-Md	3.2	8.0	3.4	7.9
T-Lg	3.0	7.5	3.2	7.5
CW-Lg	1.7	3.5	1.8	3.7

Table 4: FAIRSEQ S2T models on LibriSpeech. Dev and test WER reported. [1] Park et al. (2019). [2] Synnaeve et al. (2019). [3] Baevski et al. (2020). [4] Zhang et al. (2020).

4. Wav2Vec

Wav2Vec is a framework for self-supervised learning of representations from raw audio data. Basically, it learns to efficiently represent the raw audio data as a vector space encoding.

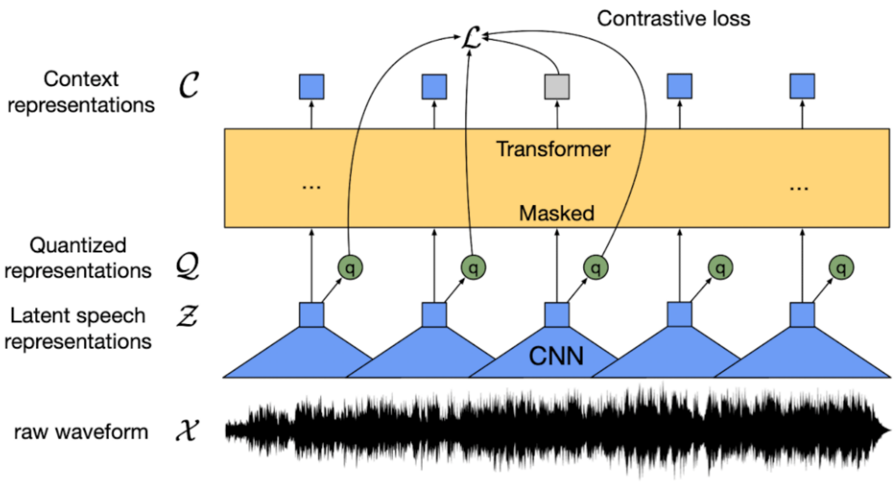


Illustration of the Wav2vec2 framework

A major advantage of this approach is that we end up training a generic audio model that could be used for multiple downstream tasks! And because of the self-supervised learning, we don't need access to huge amount of labelled data. In the paper, after pre-training on unlabelled speech, the model is fine-tuned on small labelled data with a Connectionist Temporal Classification (CTC) loss for speech recognition task.

Architecture

The complete architecture of the framework can be divided into 3 components, they are

Feature encoder: This is the encoder part of the model. It takes the raw audio data as input and outputs feature vectors. Input size is limited to 400 samples which is 20ms for 16kHz sample rate. The raw audio is first standardized to have zero mean and unit variance. Then it is passed to 1D convolutional neural network (temporal convolution) followed by layer normalization and GELU activation function. There could be 7 such convolution blocks with constant channel size (512), decreasing kernel width (10, 3x4, 2x2) and stride (5, 2x6). The output is list of feature vectors each with 512 dimensions.

Transformers: The output of the feature encoder is passed on to a transformer layer. One differentiator is use of relative positional embedding by using convolution layers, rather than using fixed positional encoding as done in original Transformers paper. The block size differs, as 12 transformers block with model dimension of 768 is used in BASE model but 24 blocks with 1024 dimension in LARGE version.

Quantization module: For self-supervised learning, we need to work with discrete outputs. For this, there is a quantization module that converts the continuous vector output to discrete representations, and on top of it, it automatically learns the discrete speech units.

This is done by maintaining multiple codebooks/groups (320 in size) and the units are sampled from each codebook are later concatenated *(320x320=102400 possible speech units)*. The sampling is done using Gumbel-SoftMax which is like argmax but differentiable.

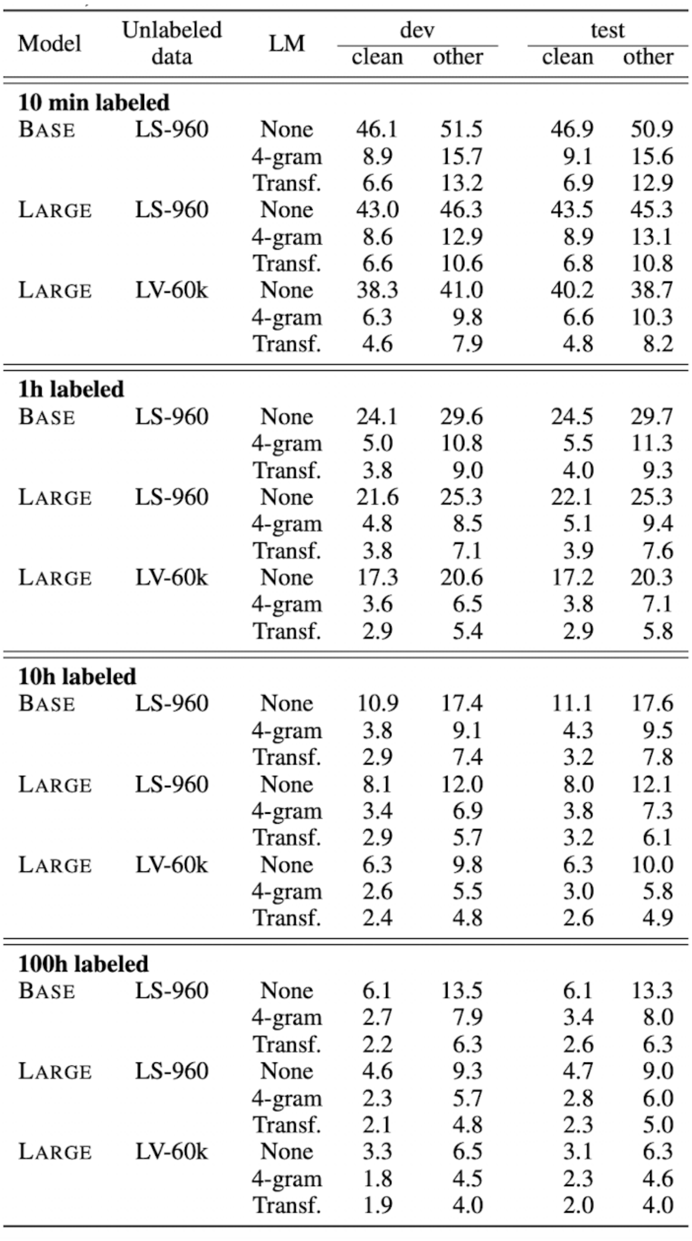
Training

- To pre-train the model, Wav2Vec2 masks certain portions of time steps in the feature encoder which is similar to the masked language model. The aim is to teach the model to predict the correct quantized latent audio representation in a set of distractors for each time step.
- The overall training objective is to minimize contrastive loss (Lm) and diversity loss (Ld) in L=Lm+αLd. Contrastive loss is the performance on the self-supervised task. Diversity loss is designed to increase the use of the complete quantized codebook representations and not only a select subset.
- For pretraining, the datasets used were (1) Librispeech corpus with 960 hours of audio data, (2) LibriVox 60k hours of audio data that was later subset to 53.2k hours. Only unlabelled data was used for pretraining.
- To make the model more robust to different tasks, we can finetune the model on different task-specific modifications and datasets. Here, the paper finetuned for ASR by adding a randomly initialized classification layer on top on the Transformer layer with a class size equal to the size of vocab. The model is optimized by minimizing the CTC loss.
- Adam was used as an optimization algorithm and the learning rate is warmed up till 10% of the training duration, then kept constant for the next 40%, and finally linearly decayed for the remaining duration. Also, for the first 60k updates only the output classifier was trained after which the Transformer is also updated. The feature encoder is kept frozen (not trained at all).

Results

There are two interesting points to note from the results of the Wav2Vec2 model,

- The model is able to learn ASR with as minimum as 10 mins of labelled data! As shown below, LARGE model pre-trained on LV-60k and finetuned on Librispeech with CTC loss is giving 4.6/7.9 WER! This is a very good news in case you want to finetune the model for your domain or accent!
- The choice of decoder can lead to improvement in performance. As obvious from the results, Transformer decoder is giving best performance, followed by n-gram and then CTC decoding. But also note that the CTC decoding will gives the best inference speed. The suggested decoder could be 4-gram, as it provides huge improvement in performance by fixing the spelling mistakes and grammar issues of CTC and is still faster than Transformer decoders.



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

WER on Librispeech dev/test data