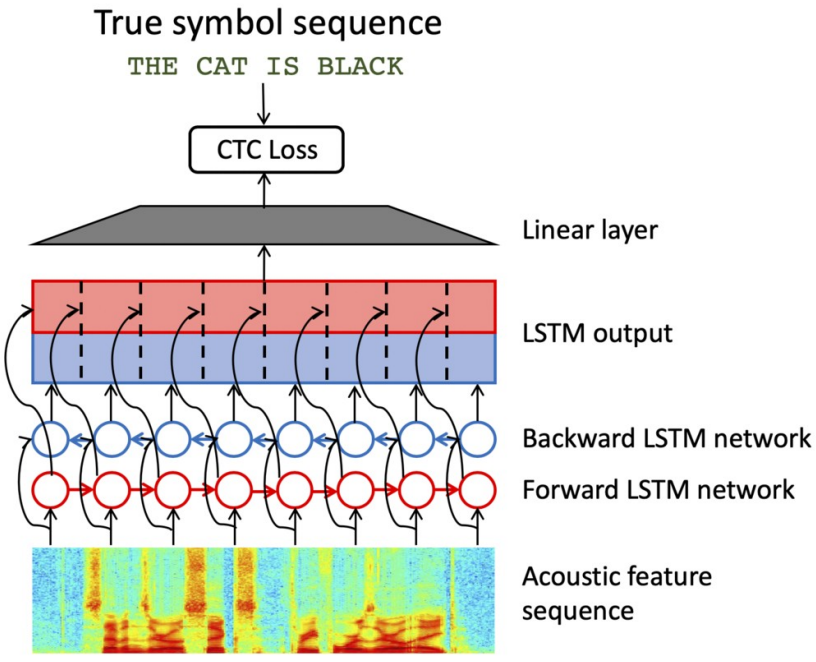
Natural Language and Speech Processing
Lecture 13: Self-supervised Models for Speech

Tanel Alumäe

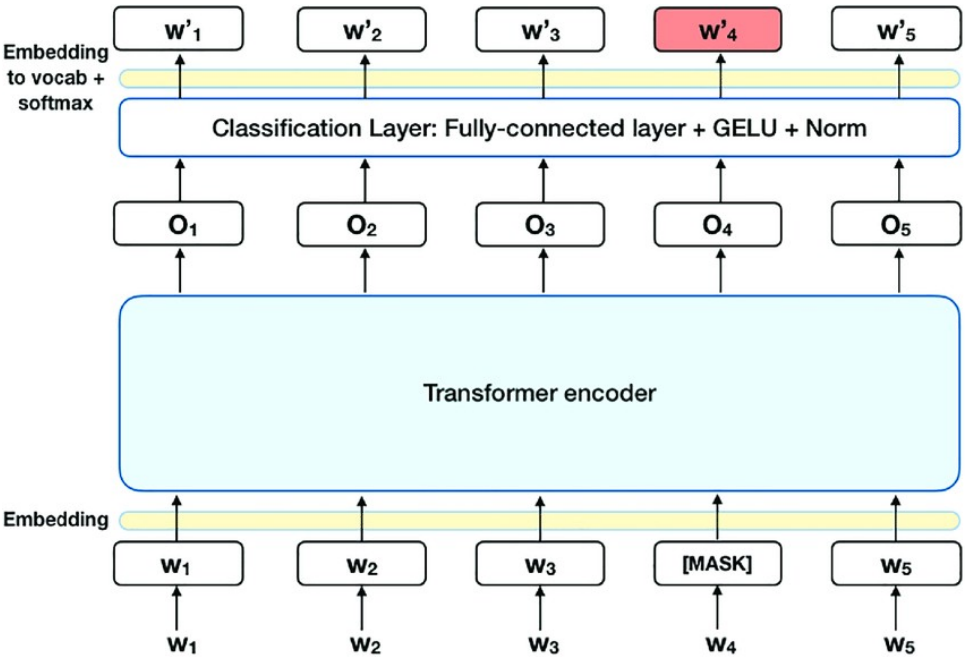
Limitations of supervised learning

- Deep learning = learning hierarchical speech representations (=features)
- Standard approach: supervised learning using large annotated corpus
- Limitations:
  - Expensive
  - Time consuming
  - Requires humans
- Our brain does not use supervised learning only
- Self-supervised learning extremely successful in NLP (BERT!)
  - Can we do the same with speech?



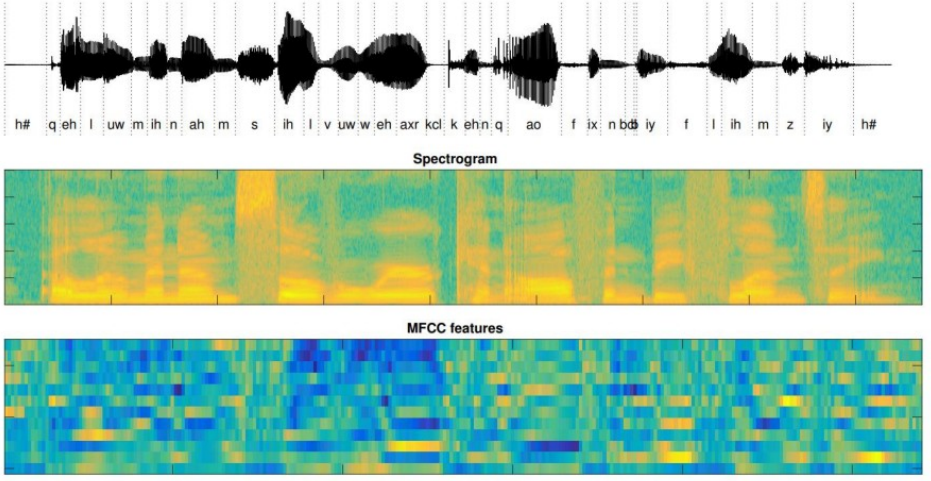
Self-supervised learning

- Self-supervised learning: the supervision is extracted from the signal itself!
- In general, self-supervised learning is done by applying some transform to the input and using the original input as the supervision
- E.g. BERT: **mask** some words in the input, and train the model to restore them



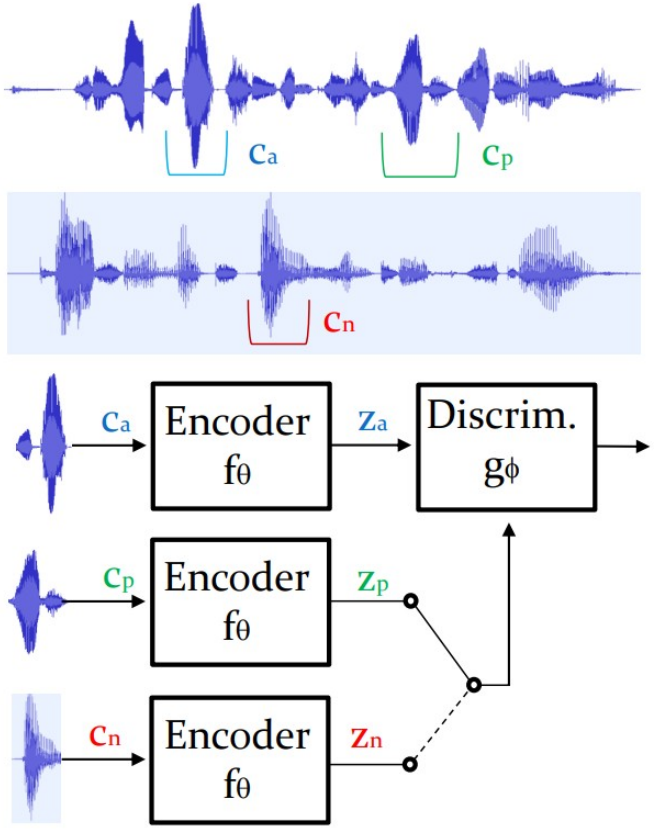
Processing waveforms...

- Self-supervised learning is challenging for speech
- Sequences are very high-dimensional
  - Text: ~10 word per sentence
  - Speech: 16000 samples per second, ~160K per utterance
  - Filterbanks: 40 * 100 features per second, ~40K per utterance



Some early models: Local Info Max (LIM)

- Train two models in parallel: Encoder and Disciminator
- Train by sampling random chunks from large and diverse unlabelled training data (containing speech from a very large variety of speakers):
  - Choose random chunk C_a (anchor)
  - Choose another random chunk from the same sentence C_p (positive)
  - Choose a random chunk from another random sentence C_n (negative)
- Training:
  - Process C_a, C_p, C_n using the Encoder
  - The Discriminator should figure out if their two inputs come from the same or different sentences
- Encoder and discriminator are jointly trained – both cooperate to solve this task better!
- The encoder can be later used as a feature extractor for the supervised speaker recognition task

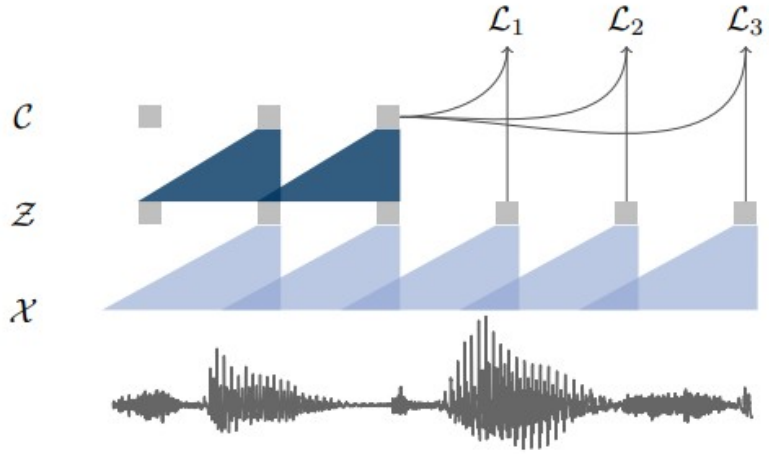


LIM

- Improves speaker recognition results
- Strengths:
  - LIM highlights high-quality speaker representations
  - LIM is simple and efficient (local information only)
- Problems:
  - The LIM representations are very **task-specific**

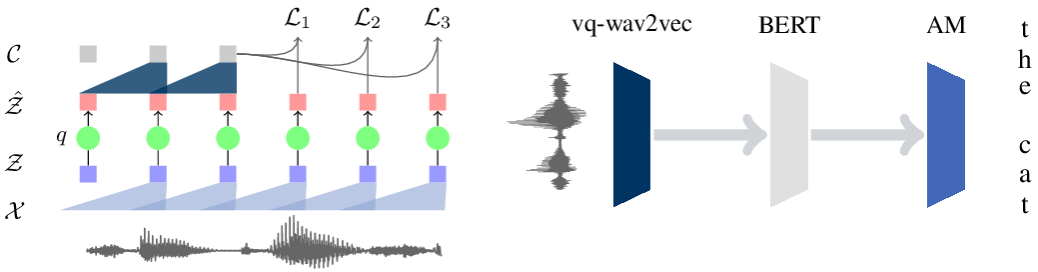
Wav2vec 1.0

- First neural network (5-layer convolutional network with „causal” convolutions) transfers raw audio X to representation (vector) Z
- Second network takes multiple z vectors and transforms them into C, using causal convolutions
  - Total receptive field of the second network is 210 ms
- Contrastive loss training objective: distinguish a sample z_{i+k} that is *k* steps in the future from distractor samples $\check{z}$ drawn randomly from the **same audio file**
  - K=1..12 was found best
  - They found that drawing distractor samples from other audio files gives inferior results
- After training, use the first neural network as a feature extractor (instead of filterbank features)
- Results: up to 36% relative improvement in word error rate when only a few hours of transcribed data is available
- Why not just train the model to predict the z (e.g., by mean squared error?)
  - Encoder (first network) learns to output constant vector z=[0..0] for all inputs, and the loss will be 0!
  - Needs some negative samples!



vq-wav2vec + BERT

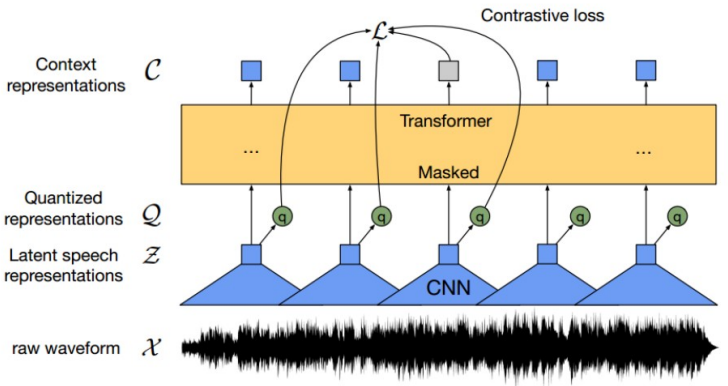
- vq-wav2vec: add a quantization step between feature encoder output Zt and aggregator network C to get Qt (using either Gumbel Softmax or K-Means)
  - That is vector z is mapped to cluster IDs [0,... 100000]
- Rest same as wav2vec
- Why discretize?
  - Can apply a host of techniques developed by NLP community (since textual input is discrete)
- Once vq-wav2vec is trained, pre-train BERT by asking it to predict masked input tokens (just like in NLP)
  - Note that we don't need any labels yet
- Finally, add a classification layer on top and train it in supervised fashion using labels for ASR (or any task)
  - Simply using codewords as inputs cannot outperform baseline (since only finite possibilities) but adding a BERT on top helps outperform wav2vec!
- Question: Why not train vq-wav2vec and BERT together?
  - Answer: Tokens input to BERT must be fixed



(a) vq-wav2vec (b) Discretized speech training pipeline

wav2vec 2.0

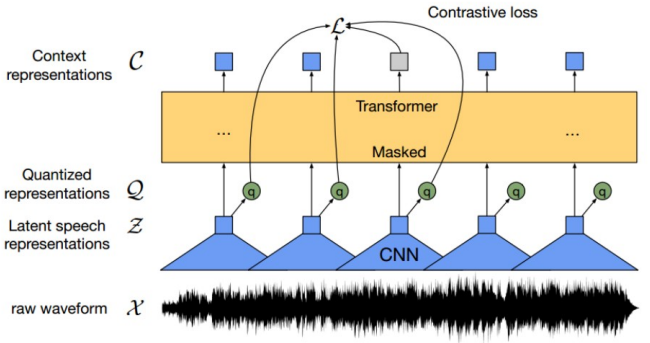
- Compared to wav2vec 1.0:
  - No more causal convolutions (i.e., model also sees into the future)
  - Second network is replaced by a transformer
  - Has quantization like vq-wav2vec + BERT, but trained end-to-end
- Now quite similar to BERT
  - Idea: Let's suppose we have an utterance (audio) „dog eats a bone"
  - We hide a random part of it: „dog eat_____ne"
  - In order to restore „_____" -> „s a bo" (that is, the corresponding audio*), the model has to learn a lot about how speech typically occurs, and also about the language/words
  - This ability could be useful also for speech recognition!



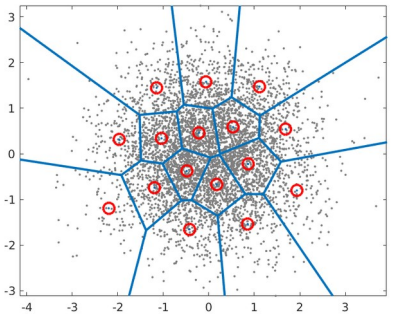
- Convolutional feature encoder $f : \mathcal{X} \mapsto \mathcal{Z}$
- Transformer $g : \mathcal{Z} \mapsto \mathcal{C}$
- Quantization module $\mathcal{Z} \mapsto \mathcal{Q}$

Architecture of wav2vec 2.0

- Input raw waveform
- CNN: audio -> features
  - Series of conv layers
  - Typically 512-dim vector after every 20 ms
- Resulting vector z goes to two branches:
  - Quantization module: z -> q
    - z's occurring in training data are clustered and mapped to cluster centre IDs
    - Cluster centers: the codebook
  - Transformer: z -> c
    - Series of Transformer encoder layers (like BERT)
    - But some of the z's are masked (i.e., replaced with a zero vector)
      - Masking is done using p=0.065, M=10
      - That is: a frame is chosen with a probability 6.5% and next 10 frames are hidden
      - Just masking one frame would be too easy!
- The model is trained to restore the masked frames
  - Actually, the quantized versions q
  - The output c is trained to be similar to q



- Convolutional feature encoder $f : \mathcal{X} \mapsto \mathcal{Z}$
- Transformer $g : \mathcal{Z} \mapsto \mathcal{C}$
- Quantization module $\mathcal{Z} \mapsto \mathcal{Q}$

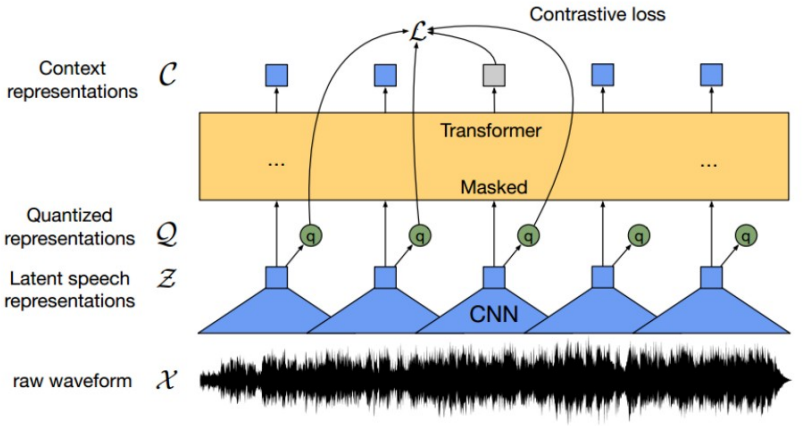


Training wav2vec2

Loss function:

$$\mathcal{L}_m = -\log \frac{\exp(\text{sim}(c_t, q_t)/\kappa)}{\sum_{\tilde{q} \sim Q_t} \exp(\text{sim}(c_t, \tilde{q})/\kappa)}$$

- Compute similarity (cosine similarity) between c_t and q_t
- Also, compute similarity between c_t and the quantized representations of 100 randomly drawn distractors
  - Distractors are uniformly sampled from other masked timesteps of the same utterance
- Apply *softmax* to the similarities
- So:
  - The model must learn to reproduce c that similar to the masked quantized representation q
  - And c must be dissimilar to other q-s
- Secondary term: *diversity loss*
  - Encourages the model to use codebook elements equally frequently



- Convolutional feature encoder $f : \mathcal{X} \mapsto \mathcal{Z}$
- Transformer $g : \mathcal{Z} \mapsto \mathcal{C}$
- Quantization module $\mathcal{Z} \mapsto \mathcal{Q}$

Diversity loss

- Regularization

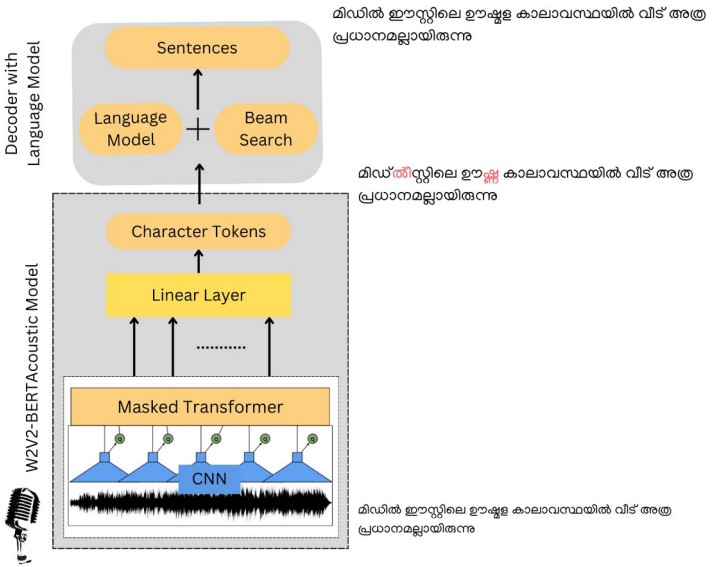
$$\mathcal{L}_d = \frac{1}{GV} \sum_{g=1}^{G} -H(\bar{p}_g) = \frac{1}{GV} \sum_{g=1}^{G} \sum_{v=1}^{V} \bar{p}_{g,v} \log \bar{p}_{g,v}$$

Contrastive loss + Diversity loss

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

Finetuning wav2vec2.0 for ASR

- Finetuning a pretrainedwav2vec2.0 model for speech recognition is simple
- Just add a softmax layer for outputtokens, and train using CTC loss
- Usually, convolutional speech encoder layer is frozen
- Transformer is trained using a lower learning rate than the freshly added softmax layer
- Language model can be used to make CTC decoding more accurate
- SpecAugment is applied during training



Wav2vec 2.0: results

- Now, results depend on two training datasets:
  - Untranscribed training set (potentially very large, since cheap)
  - Transcribed dataset (small, because expensive to produce)
- Wav2vec2.0 produces promising results with only 10 minutes of training data
  - Given 60k of very similar unlabelled data, WER (test/other) is 8.2%
- Also, with 60k hours of unlabelled data and 100 h of labelled data
  - Only labelled data: WER (test/other) 18.6%
  - Semi-supervised learning (pseudo-labelling) on 60k hours: WER 7.1%
  - Wav2vec, pretrained on 60k hours: WER 4.0%

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

Multilingual wav2vec2.0

- What happens if wav2vec2 model is pretrained on massive amounts of multilingal speech?
- XLS-R wav2vec2 model:
  - Trained on 436K hours of speech from 128 languages
  - Audiobooks, Mozilla CommonVoice, VoxPopuli (European Parliament audio), **VoxLingua107** (YouTube, compiled by us!)
- The resulting model is useful for a variety of tasks:
  - Speech recognition
  - Speech translation
  - Speaker recognition
  - Language identification

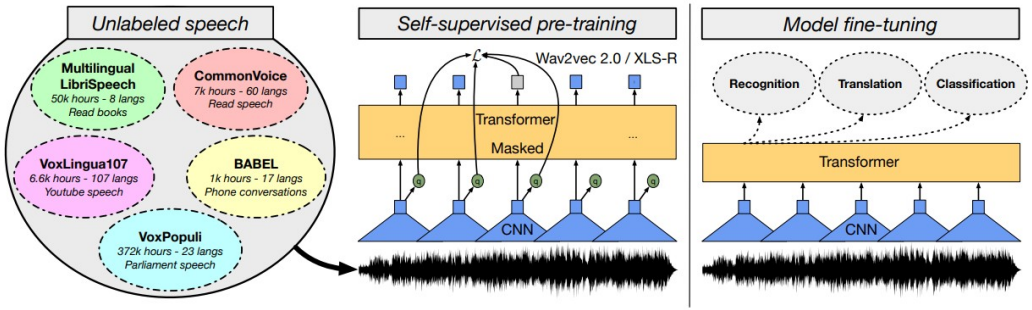


Figure 1: **Self-supervised cross-lingual representation learning.** We pre-train a large multilingual wav2vec 2.0 Transformer (XLS-R) on 436K hours of unannotated speech data in 128 languages. The training data is from different public speech corpora and we fine-tune the resulting model for several multilingual speech tasks.

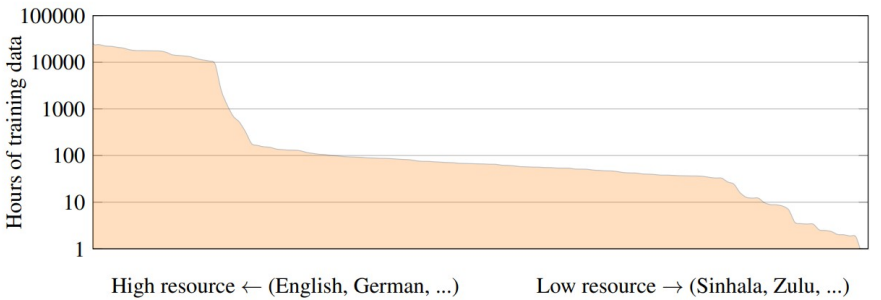


Figure 2: Illustration of the unlabeled training data distribution across the 128 languages of XLS-R.

Multilingual wav2vec2.0: results

- They trained 3 models:
  - 300M params
  - 1B params
  - 2B params
- ASR: beats previous highly tuned ensemble results
- Speech translation:
  - The large models are especially good
  - Actually combines wav2vec2 as encoder and mBART's decoder

Table 7: Speech recognition results on BABEL in terms of word error rate (WER) on Assamese (as), Tagalog (tl), Swahili (sw), Lao (lo) and Georgian (ka).

	as	tl	sw	lo	ka
Labeled data	55h	76h	30h	59h	46h
Previous work					
Alumäe et al. (2017)	-	-	-	-	32.2
Ragni et al. (2018)	-	40.6	35.5	-	-
Inaguma et al. (2019)	49.1	46.3	38.3	45.7	-
XLSR-10 (Conneau et al., 2021)	44.9	37.3	35.5	32.2	-
XLSR-53 (Conneau et al., 2021)	44.1	33.2	26.5	-	31.1
This work					
XLS-R (0.3B)	42.9	33.2	24.3	31.7	28.0
XLS-R (1B)	40.4	30.6	21.2	30.1	25.1
XLS-R (2B)	**39.0**	**29.3**	**21.0**	**29.7**	**24.3**

Table 3: Speech translation: results for X → English directions on CoVoST-2 in terms of average BLEU for 21 directions grouped into high/mid/low-resource labeled data directions. Appendix A lists detailed results for all languages.

	high	mid	low	Avg.
Prior work				
XLSR-53 (Conneau et al., 2021)	30.3	11.1	3.2	10.3
VP-100K (Wang et al., 2021b)	27.7	13.2	4.6	11.1
XMEF-En (Li et al., 2021b)	32.4	16.8	4.0	12.4
XMEF-X (Li et al., 2021b)	34.2	20.2	5.9	14.7
This work				
XLS-R (0.3B)	30.6	18.9	5.1	13.2
XLS-R (1B)	34.3	25.5	11.7	19.3
XLS-R (2B)	**36.1**	**27.7**	**15.1**	**22.1**

Multilingual wav2vec2.0: results

- Also helps with spoken language identification
- Will experiment with this in the lab

Table 12: Language identification on VoxLingua107. We report the error rate on the development set spanning 33 languages.

	Error Rate (%)		
	0...5 sec	5...20 sec	Average
Previous work			
Valk & Alumäe (2020)	12.3	6.1	7.1
Ravanelli et al. (2021)	-	-	6.7
This work			
wav2vec 2.0 LV-60K (300M)	11.5	6.3	7.2
XLS-R (0.3B)	**9.1**	**5.0**	**5.7**

W2V-BERT

- Chung et al, “W2V-BERT: COMBINING CONTRASTIVE LEARNING AND MASKED LANGUAGE MODELING FOR SELF-SUPERVISED SPEECH PRE-TRAINING” (Google)
- Main difference to wav2vec2:
  - Encoder is omitted, input is filterbank features
    - Model training is faster and loss in accuracy is minimal
  - Transformer replaced with Conformer (next slide)
  - Most important: uses BERT-style masked LM loss (pure cross-entropy softmax) in addition to the contrastive loss

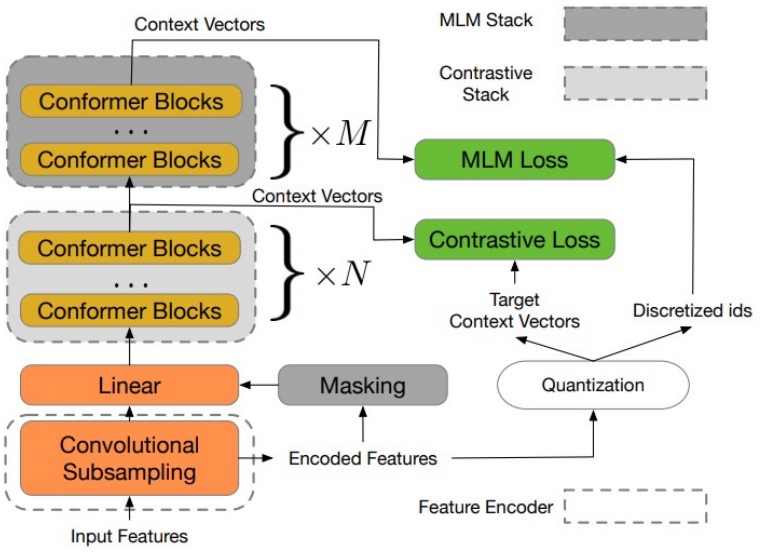


Fig. 1: Illustration of the w2v-BERT pre-training framework. w2v-BERT is composed of a feature encoder, a contrastive module, and a masked language modeling (MLM) module, where the latter two are both a stack of conformer blocks. N and M denote the number of conformer blocks in the two modules, respectively.

Conformer

- Adds a convolutional module in the transformer blocks
- Currently most common architecture in state-of-the-art speech recognition systems
- Why convolution:
  - Local dependencies are very important in speech, so convolution allows the model to exploit them more easily than pure transformer

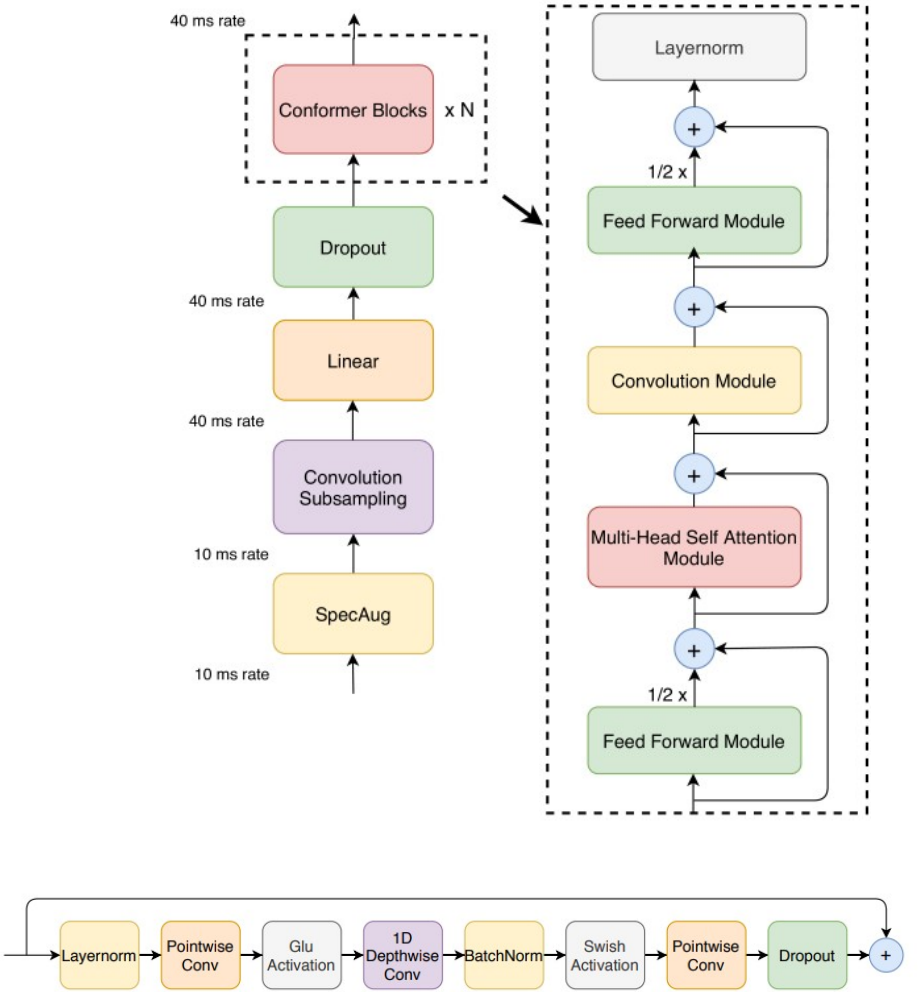


Figure 2: ***Convolution module.*** *The convolution module contains a pointwise convolution with an expansion factor of 2 projecting the number of channels with a GLU activation layer, followed by a 1-D Depthwise convolution. The 1-D depthwise conv is followed by a Batchnorm and then a swish activation layer.*

W2V-BERT: MLM loss

MLM (masked language model loss):

- Exactly as in BERT
- Codebook entries will get and ID
- The MLM head of the model will classify *c* according to codebook IDs
  - Softmax over all IDs
  - Cross-entropy loss
- Contrastive loss is still needed
  - Otherwise the model learns to cheat: all frames will be put into the same codebook and MLM head of the model will learn that and will classify everything perfectly
- So, now 3 loss functions:
  contrastive loss + diversity loss + MLM loss

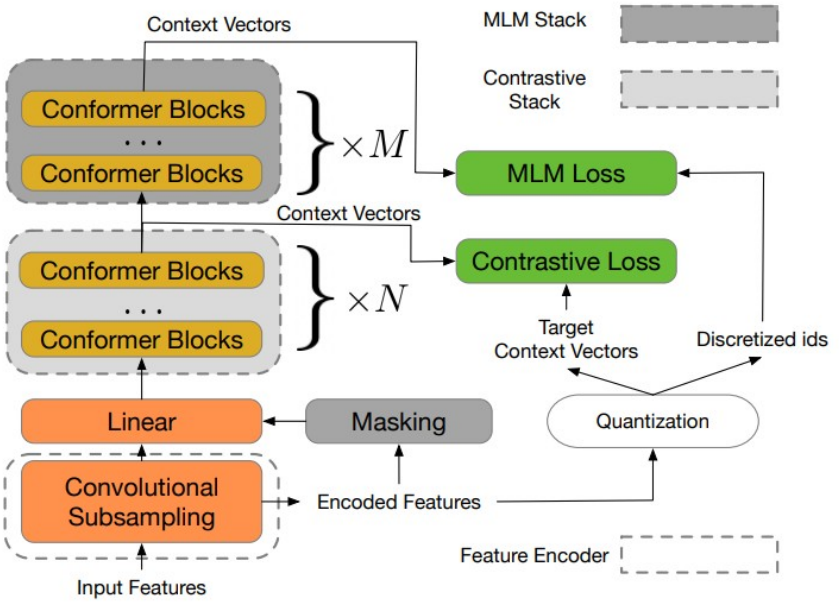


Fig. 1: Illustration of the w2v-BERT pre-training framework. w2v-BERT is composed of a feature encoder, a contrastive module, and a masked language modeling (MLM) module, where the latter two are both a stack of conformer blocks. N and M denote the number of conformer blocks in the two modules, respectively.

W2V-BERT: results

Experiment:

- 60k hours (Libri-Light unlab-60k) of unlabelled data for pretraining
- 100 hr labelled Librispeech corpus for final finetuning
- W2V-BERT much better than wav2vec2 ja w2-Conformer XL (same as wav2vec2, but Conformer, and more parameters)

Table 3: WERs (%) when using the LibriSpeech 100hr subset as supervised data. For all methods, both self-training and LM fusion are not used. References are where the numbers are quoted from.

Method	dev	dev-other	test	test-other
Baseline				
wav2vec 2.0 [23]	3.3	6.5	3.1	6.3
w2v-Conformer XL [21]	2.5	4.7	2.6	4.9
w2v-BERT XXL (Ours)	2.3	4.0	2.3	4.3

JUST: Joint unsupervised and supervised training

- W2V-BERT and the added speech recognition model are all trained jointly
  - Uses RNN-T model (see lecture 2 weeks ago)
- Now, already 4 loss functions
  - Contrastive loss
  - Diversity loss
  - MLM loss
  - RNN-T loss

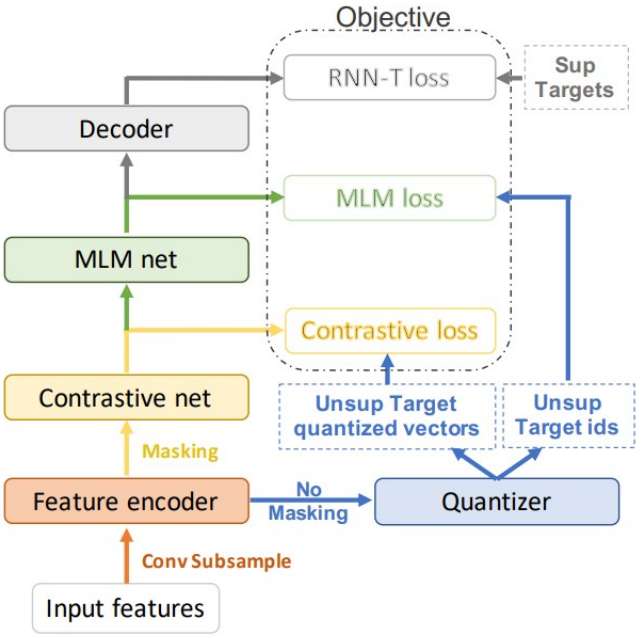


Fig. 1: An overview of our JUST framework. Feature encoder, contrastive net, MLM net and decoder are stacked sequentially. The output of each module constitutes a loss in the objective function. Target vectors and ids in the blue boxes are for unsupervised losses. Supervised targets in the grey box are for RNN-T loss.

JUST: results

Training data: MLS (Multilingual Librispeech)

MLS dataset [12] is used as the benchmark in our experiments. It is derived from read audiobooks of LibriVox. There are 8 languages (namely English (**en**), German (**de**), Dutch (**nl**), French (**fr**), Spanish (**es**), Italian (**it**), Portuguese (**pt**) and Polish (**pl**)), with 44.5k hours of English and 6k hours for other languages combined. Some low-resource language like Polish only has 100 hours. Each utterance is 10-20 seconds long.

Method	External data	**de**	**en**	**es**	**fr**	**it**	**nl**	**pl**	**pt**	Avg	Avg (w/o **en**)
Monolingual [12]	-	7.10	6.76	6.68	6.58	11.78	13.09	21.66	20.52	11.8	12.5
+ 5-gram LM [12]	-	6.49	5.88	6.07	5.58	10.54	12.02	20.39	19.49	10.8	11.5
XLSR-53 [16]	Y	7.0	-	6.3	7.6	10.4	10.8	17.2	14.7	10.6	10.6
B0 (random init.) [13]	Y	5.5	6.1	5.8	6.9	11.9	11.9	15.4	16.2	10.0	10.5
B0 (15-language model init.) [13]	Y	5.0	6.6	4.7	6.1	10.1	11.1	10.9	15.5	8.8	9.1
E3 (15-language model init.) [13]	Y	4.3	**5.8**	4.2	**4.9**	8.8	9.9	10.4	15.2	7.9	8.2
JUST ($\beta = 0$)	N	5.5	6.9	4.1	6.0	9.3	10.3	11.3	9.4	7.8	8.0
w2v2 Pretrain ($\mathcal{L}_c$+$\alpha\mathcal{L}_d$)+pure Finetune ($\mathcal{L}_s$)	N	4.7	6.8	4.1	5.8	9.9	10.3	12.1	12.6	8.3	8.5
w2v-bert Pretrain ($\mathcal{L}_u$)+pure Finetune ($\mathcal{L}_s$)	N	4.3	6.6	3.8	5.0	9.1	9.9	8.1	14.6	7.7	7.8
w2v-bert Pretrain ($\mathcal{L}_u$)+JUST Finetune ($\mathcal{L}$)	N	4.2	6.6	4.0	5.0	9.0	9.5	7.6	15.1	7.6	7.8
w2v2 Joint Training ($\mathcal{L}_s$+β($\mathcal{L}_c$+$\alpha\mathcal{L}_d$))	N	4.6	6.7	4.1	5.7	8.9	9.9	9.8	9.3	7.4	7.5
JUST ($\mathcal{L}$)	N	4.6	6.8	3.9	5.7	9.1	9.9	9.1	8.6	7.2	7.3
JUST ($\mathcal{L}$)+pure Finetune ($\mathcal{L}_s$)	N	**4.1**	6.5	**3.7**	5.2	**8.2**	**9.5**	**6.6**	**8.0**	**6.5**	**6.5**

Baselines:

- Monolingual model trained only on this language data, no pretraining
- XLSR-53: multilingual wav2vec2, finetuned on this language data (very standard approach)
- JUST (β = 0): proposed multilingual model, but trained using only the RNN-T loss

JUST: results

- w2v-bert Pretrain + pure Finetune: traditional approach: first self-supervised pretraining, then finetuning
- JUST: trained with 4 loss functions
- JUST + pure Finetune: JUST, but finally finetuned using only the RNN-T loss

So:

- Joint training (WER 7.3) is better than serial pretraining+finetuning (WER 7.8)
- Joint training + finetuning is even better (WER 6.5)

Method	External data	de	en	es	fr	it	nl	pl	pt	Avg	Avg (w/o en)
Monolingual [12]	-	7.10	6.76	6.68	6.58	11.78	13.09	21.66	20.52	11.8	12.5
+ 5-gram LM [12]	-	6.49	5.88	6.07	5.58	10.54	12.02	20.39	19.49	10.8	11.5
XLSR-53 [16]	Y	7.0	-	6.3	7.6	10.4	10.8	17.2	14.7	10.6	10.6
B0 (random init.) [13]	Y	5.5	6.1	5.8	6.9	11.9	11.9	15.4	16.2	10.0	10.5
B0 (15-language model init.) [13]	Y	5.0	6.6	4.7	6.1	10.1	11.1	10.9	15.5	8.8	9.1
E3 (15-language model init.) [13]	Y	4.3	**5.8**	4.2	**4.9**	8.8	9.9	10.4	15.2	7.9	8.2
JUST ($\beta = 0$)	N	5.5	6.9	4.1	6.0	9.3	10.3	11.3	9.4	7.8	8.0
w2v2 Pretrain ($\mathcal{L}_c$+$\alpha\mathcal{L}_d$)+pure Finetune ($\mathcal{L}_s$)	N	4.7	6.8	4.1	5.8	9.9	10.3	12.1	12.6	8.3	8.5
w2v-bert Pretrain ($\mathcal{L}_u$)+pure Finetune ($\mathcal{L}_s$)	N	4.3	6.6	3.8	5.0	9.1	9.9	8.1	14.6	7.7	7.8
w2v-bert Pretrain ($\mathcal{L}_u$)+JUST Finetune ($\mathcal{L}$)	N	4.2	6.6	4.0	5.0	9.0	9.5	7.6	15.1	7.6	7.8
w2v2 Joint Training ($\mathcal{L}_s$+β($\mathcal{L}_c$+$\alpha\mathcal{L}_d$))	N	4.6	6.7	4.1	5.7	8.9	9.9	9.8	9.3	7.4	7.5
JUST ($\mathcal{L}$)	N	4.6	6.8	3.9	5.7	9.1	9.9	9.1	8.6	7.2	7.3
JUST ($\mathcal{L}$)+pure Finetune ($\mathcal{L}_s$)	N	**4.1**	6.5	**3.7**	5.2	**8.2**	**9.5**	**6.6**	**8.0**	**6.5**	**6.5**

Self-supervised *vs* semi-supervised

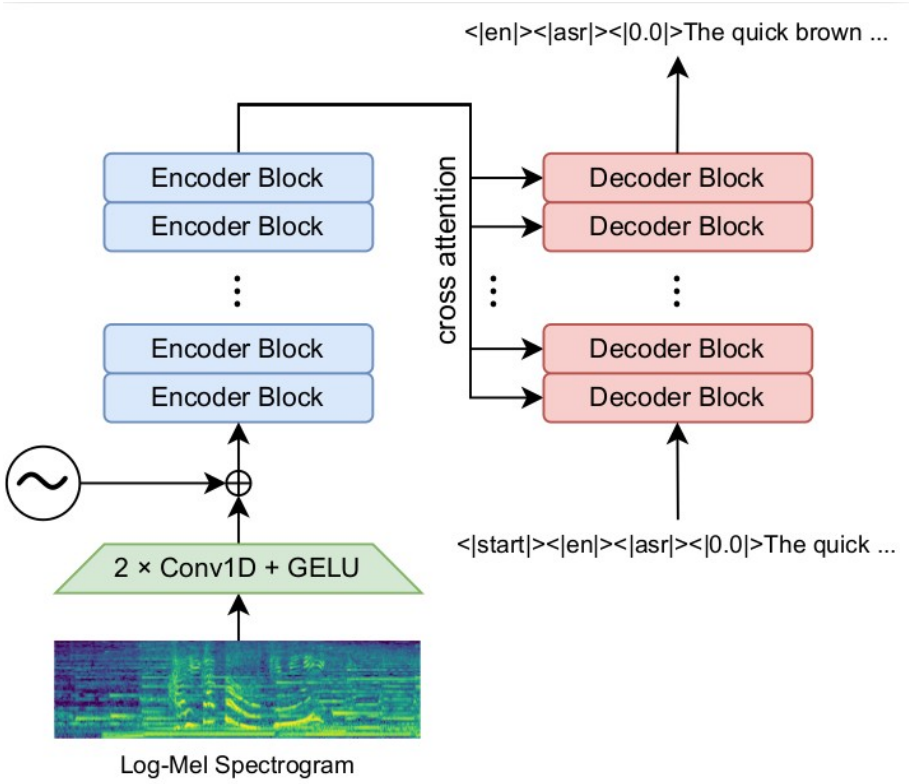
- Self-supervised training
  - Use contrastive loss (or smth similar) to pretrain a model
  - Followed by finetuning on labelled data
- Alternative: semi-supervised
  - Train a model on labelled data
  - Use it to pseudo-label unlabelled data (using the trained model + additional language model trained on external data)
  - Train a new model on labelled+pseudolabelled data
  - Repeat (optional)
- Turns out those approaches are complementary, so the best approach:
  - Pretrain using contrastive loss
  - Finetune using labelled data
  - Generate pseudo-labels for unlabelled data
  - Finetune a new model on labelled+pseudolabelled data
- For example: 960h unlabelled data, 10h labelled data
  - Pseudo-labelling only, using model trained on 10h: WER 26.0%
  - Wav2vec2: WER 6.1%
  - Encoder-decoder model, trained on pseudo-labelled data (labelled using wav2vec): WER 5.9%
  - Finetune wav2vec using pseudo-labels: WER 5.7%
- More effect with really small amount of labelled data (10 minutes)

Table 1: WER on the Librispeech dev and test sets for the Libri-light low-resource labeled data setups of 10 min, 1 hour and 10 hours. As unlabeled data we use the audio of Librispeech (LS-960) or the larger LibriVox (LV-60k). ST (s2s scratch) trains a sequence to sequence model with a word-piece vocabulary on the pseudo-labeled data from random initialization while as ST (ctc ft) fine-tunes wav2vec 2.0 with the pseudo-labels using CTC and a letter-based vocabulary. All results are with language models at inference time.

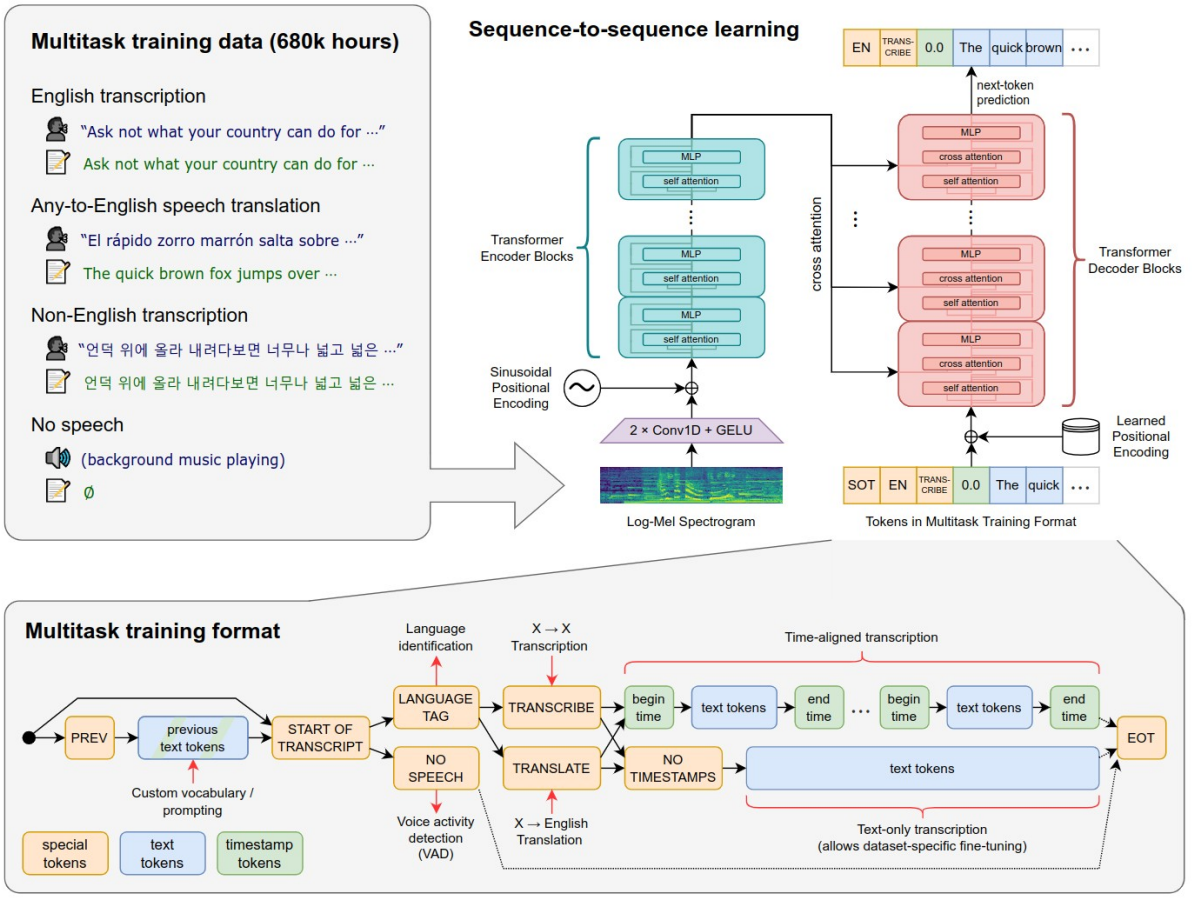
Model	Unlbld data	dev		test	
		clean	other	clean	other
10 min labeled					
Discr. BERT [27]	LS-960	15.7	24.1	16.3	25.2
wav2vec 2.0 [24]	LS-960	6.6	10.6	6.8	10.8
+ ST (s2s scratch)	LS-960	4.1	7.0	5.0	8.1
+ ST (ctc ft)	LS-960	3.6	6.6	4.0	7.2
wav2vec 2.0 [24]	LV-60k	5.0	8.4	5.2	8.6
+ ST (s2s scratch)	LV-60k	2.6	4.7	3.1	5.4
+ ST (ctc ft)	LV-60k	2.8	4.6	3.0	5.2
1h labeled					
Discr. BERT [27]	LS-960	8.5	16.4	9.0	17.6
wav2vec 2.0 [24]	LS-960	3.8	7.1	3.9	7.6
+ ST (s2s scratch)	LS-960	2.9	5.6	3.4	6.6
+ ST (ctc ft)	LS-960	2.8	5.5	3.1	6.3
10h labeled					
Discr. BERT [27]	LS-960	5.3	13.2	5.9	14.1
IPL [14]	LS-960	23.5	25.5	24.4	26.0
wav2vec 2.0 [24]	LS-960	2.9	5.7	3.2	6.1
+ ST (s2s scratch)	LS-960	2.5	5.1	3.5	5.9
+ ST (ctc ft)	LS-960	2.6	5.2	2.9	5.7

OpenAI Whisper

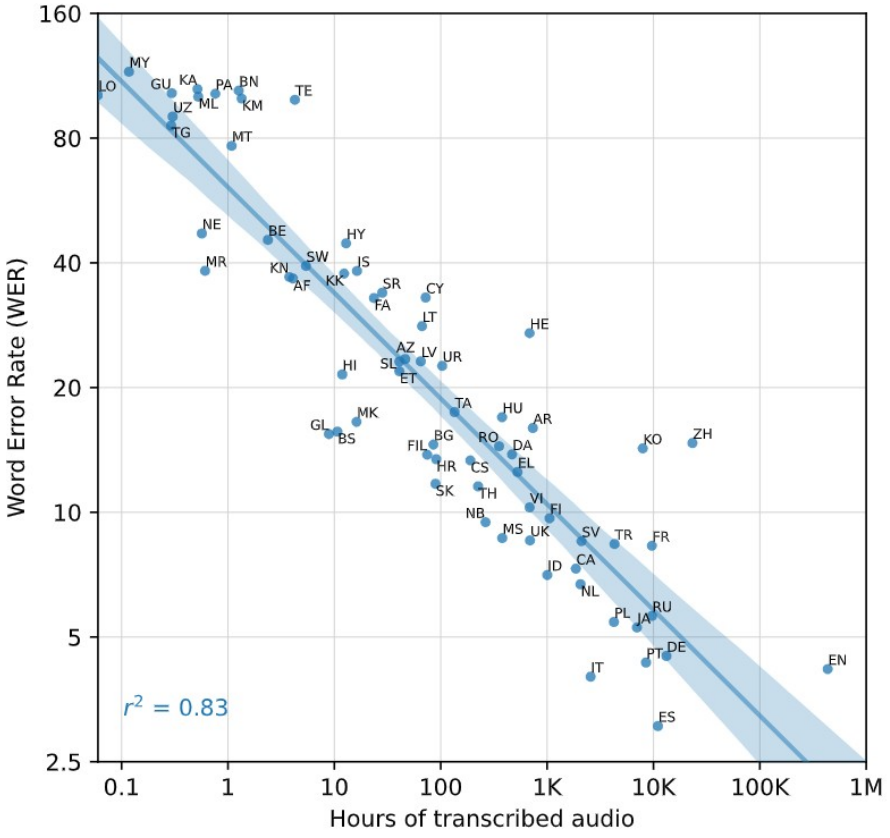
- Another popular base model for speech recognition is **OpenAI Whisper**
  - Trained on 700 000 h of multililingual transcribed audio collected from the web (?)
  - Can do speech recognition in ~100 languages (with varying quality)
  - Can also do translation to English
  - Architecture: seq2seq Transformer
    - Input: 80 dim filterbanks, passed through 2 convolutional layers
  - Different training and inference strategy than most previous models:
    - Uses fixed 30 second chunks, tolerates speaker changes



OpenAI Whisper, cont



OpenAI Whisper, cont



Model	Layers	Width	Heads	Parameters
Tiny	4	384	6	39M
Base	6	512	8	74M
Small	12	768	12	244M
Medium	24	1024	16	769M
Large	32	1280	20	1550M

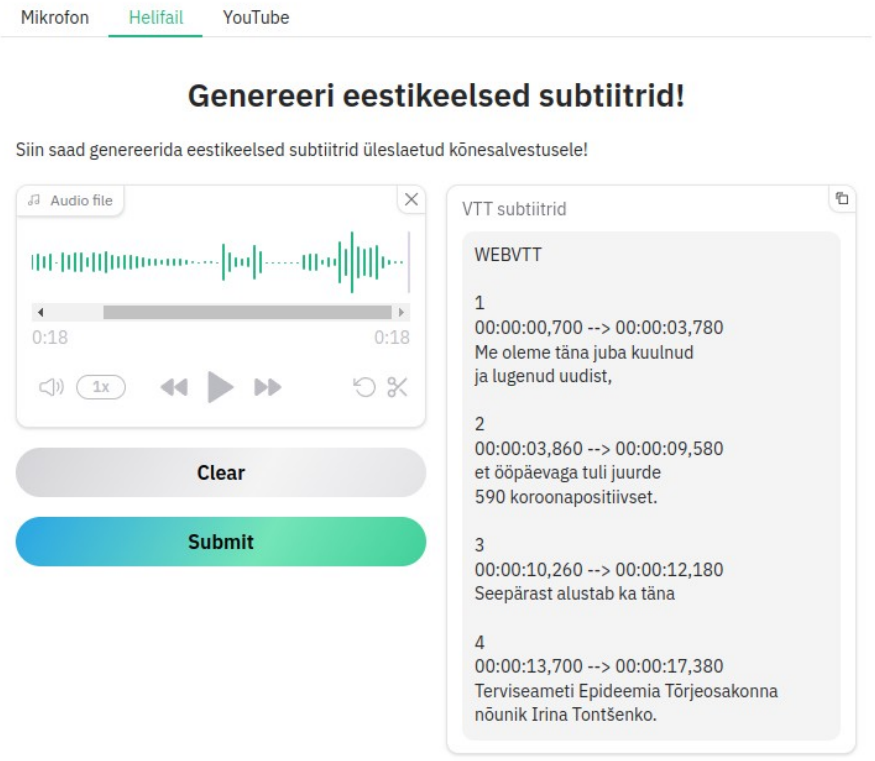
OpenAI Whisper, cont

- For many larger languages, Whisper works exceptionally well, especially on conversational data like podcasts
- For Estonian, the out-of-the-box performance is not that great
  - The transcripts are readabble but contain many „typos” which increase WER
  - After finetuning on out own Estonian data (~ 1400 h), it performs very well for Estonian
  - Model available: https://huggingface.co/TalTechNLP/whisper-large-v3-turbo-et-verbatim-2604
  - Especially good in correctly transcribing English terms and names that previous models struggle with (because the model has learned this from English data)

- Example of a Estonian tech podcast transcription (Algorütm)
  - kes Dubais on käinud, seal on üks suur see Sheikh Zayed Boulevard on ju, seal on nii-öelda alguses ja lõpus on siis ristuvad teed, et seal oleks ideaalne seda kasutada, sellepärast et noh, peamine liiklus on ühe tee peal ja siis noh, millal iganes liiklus kõrval tee peal tuleb, saad sa siis seda foore muuta ja neil oli isegi see automaatika …
  - … kas need business lounge täituvuse analüüsiks võetavad video freimid, kas seal on inimeste näod udustatud ? Jah, ütleme niimoodi, et et isegi mitte ei ole udustatud, vaid meie ei salvesta video data üldse ehk meie pipeline on niimoodi ehitatud, et ta töötab justkui selline funnel, et data tuleb sisse, tuvastused toimuvad ära ja nii-öelda kõik
  - … ja siis ma näitasin seda SQLi lauset, ma ise sellest enam aru ei saanud, sellest kolmandast versioonist, näitasin nagu, noh, inimesele, kes on ise kirjutanud, ta ütles, et ta ei oleks suutnud seda kirjutada . Et ja, ja ongi ja töötab ja ongi väga universaalne lahendus . Ja siis teine case oli siis see, et oli, oli vaja defineerida privacy by disain policy .

Whisper for subtitling

- Whisper is very good in adapting to the style of training set
- We trained another model on manually produced subtitles from Estonian TV (~800h)
- Resulting model can:
  - Produces subtitle blocks of limited size and accurate timestamps
  - Fixes grammatical errors, discards hesitations, rewrites word order and compresses in some cases
- Try: https://bark.cs.taltech.ee/subtitreeri/
- Demo: https://www.youtube.com/watch?v=bEow5vGIgZc



Questions?