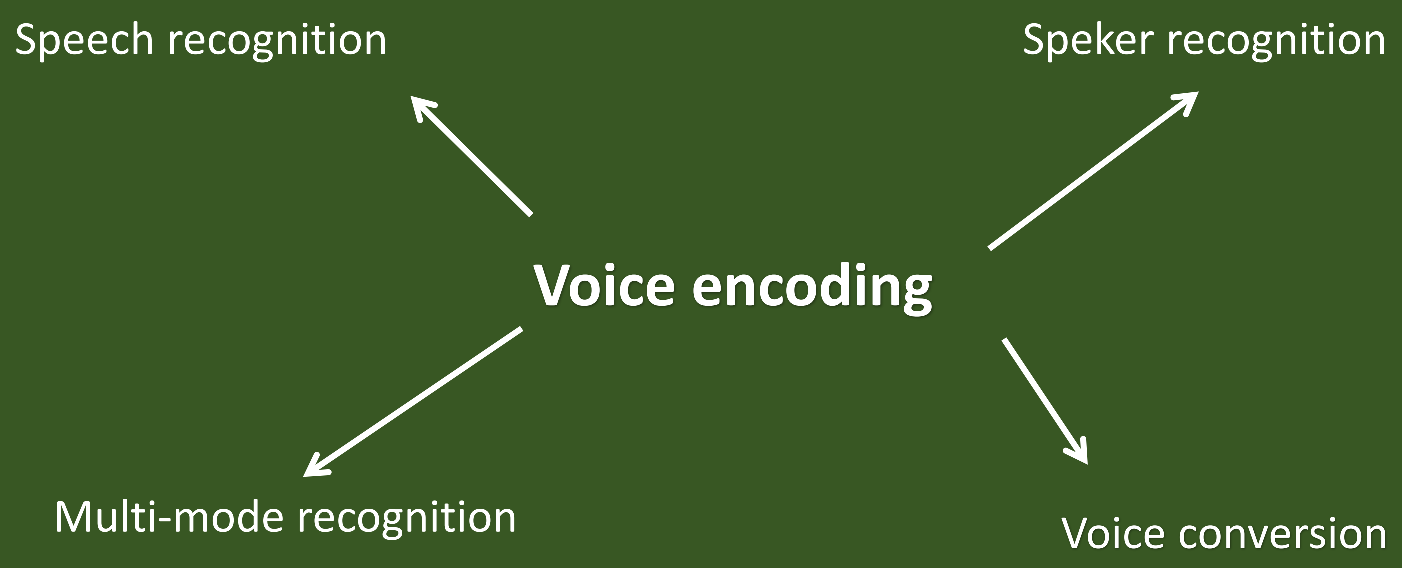
Wav2Vec

A Framework for Self-Supervised Learning of Speech Representations

分享人：算法2组-魏万顺

背景

语音领域几个任务


Speech recognition
Speker recognition
Voice encoding
Multi-mode recognition
Voice conversion

自监督学习

一堆无监督的数据，但是通过数据本身的结构或者特性，人为构造标签。

有了标签之后，就可以类似监督学习一样进行训练。

- word2vec - NLP
- transformer - NLP and other
- bert - NLP
- SimCse - NLP
- Vit - CV
- CPC - Audio
- wav2vec - Audio

关键论文

- CPC https://arxiv.org/pdf/1807.03748.pdf
- wav2vec https://arxiv.org/pdf/1904.05862.pdf
- vq-wav2vec https://arxiv.org/pdf/1910.05453.pdf
- wav2vec2.0 https://arxiv.org/abs/2006.11477v3

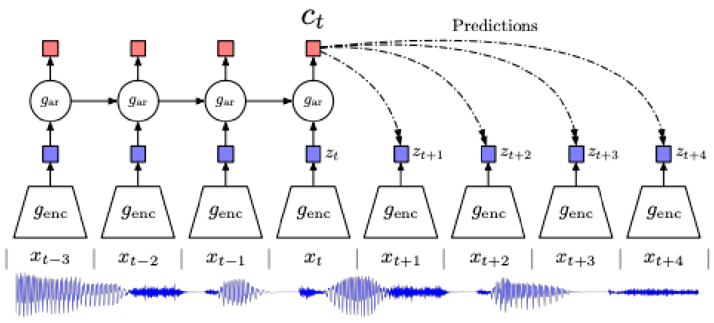
Blog

- https://maelfabien.github.io/machinelearning/wav2vec/#
- https://wandb.ai/tulasi1729/self-supervised-learning-in-audio/reports/Self-Supervised-Learning-in-Audio-and-Speech--VmlldzozODA3OTU

重要技术点介绍

CPC

- *** 卷积网络提取音频特征，抛弃mel滤波器组和mfcc特征**
- RNN作为context network
- 1个正样本，N倍负样本
- 下游对接标准声学模型



$$\mathcal{L}_N = -\mathbb{E}_X\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$$

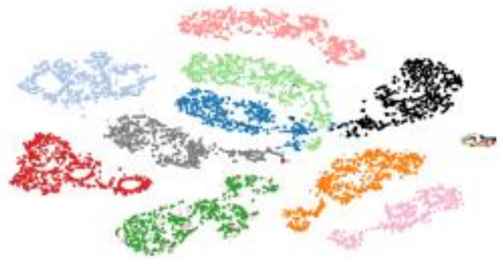


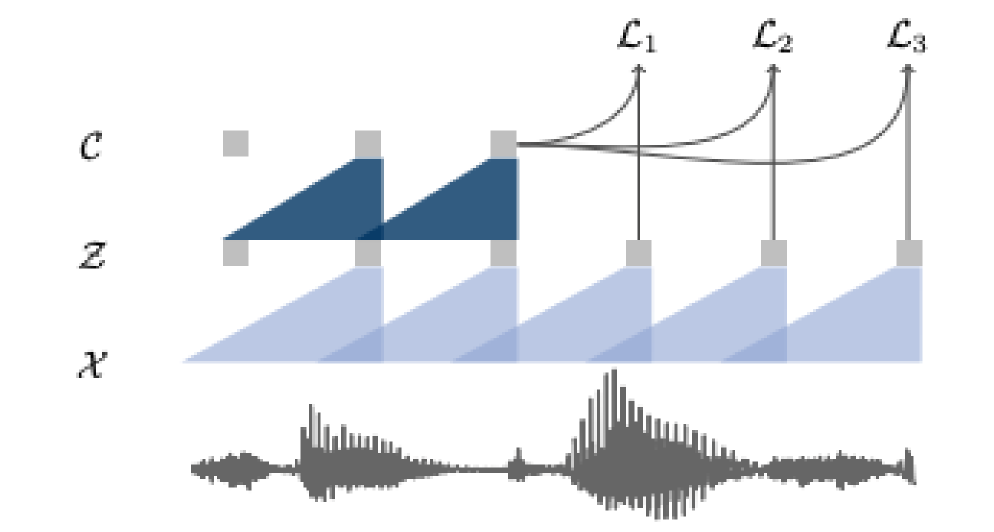
Figure 2: t-SNE visualization of audio (speech) representations for a subset of 10 speakers (out of 251). Every color represents a different speaker.

Method	ACC
Phone classification	
Random initialization	27.6
MFCC features	39.7
CPC	64.6
Supervised	74.6
Speaker classification	
Random initialization	1.87
MFCC features	17.6
CPC	97.4
Supervised	98.5

Table 1: LibriSpeech phone and speaker classification results. For phone classification there are 41 possible classes and for speaker classification 251. All models used the same architecture and the same audio input sizes.

wav2vec

- * 引入对比损失，引入对比负样本
- CNN作为context network



$$\mathcal{L}_k = -\sum_{i=1}^{T-k}\left(\log\sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i)) + \lambda \mathop{\mathbb{E}}_{\tilde{\mathbf{z}}\sim p_n}\left[\log\sigma(-\tilde{\mathbf{z}}^\top h_k(\mathbf{c}_i))\right]\right)$$

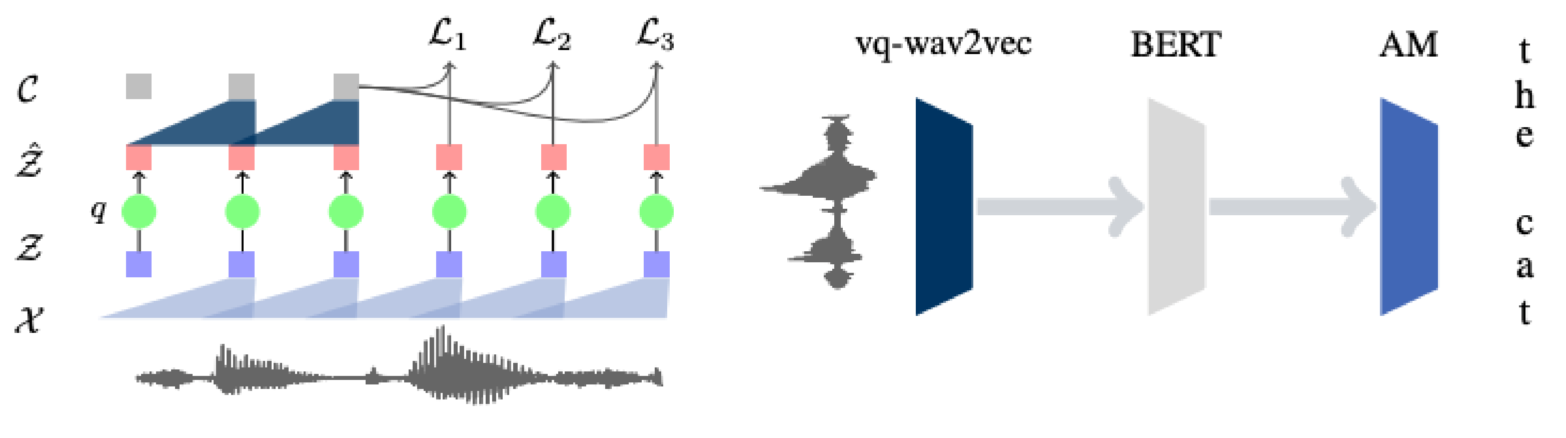
			nov93dev LER	nov93dev WER	nov92 LER	nov92 WER
Deep Speech 2 (12K h labeled speech; Amodei et al., 2016)			-	4.42	-	3.1
Trainable frontend (Zeghidour et al., 2018a)			-	6.8	-	3.5
Lattice-free MMI (Hadian et al., 2018)			-	5.66†	-	2.8†
Supervised transfer-learning (Ghahremani et al., 2017)			-	4.99†	-	2.53†
4-GRAM LM (Heafield et al., 2013)						
Baseline	–	–	3.32	8.57	2.19	5.64
wav2vec	Librispeech	80 h	3.71	9.11	2.17	5.55
wav2vec	Librispeech	960 h	2.85	7.40	1.76	4.57
wav2vec	Libri + WSJ	1,041 h	2.91	7.59	1.67	4.61
wav2vec large	Librispeech	960 h	2.73	6.96	1.57	4.32
WORD CONVLM (Zeghidour et al., 2018b)						
Baseline	–	–	2.57	6.27	1.51	3.60
wav2vec	Librispeech	960 h	2.22	5.39	1.25	2.87
wav2vec large	Librispeech	960 h	2.13	5.16	1.02	2.53
CHAR CONVLM (Likhomanenko et al., 2019)						
Baseline	–	–	2.77	6.67	1.53	3.46
wav2vec	Librispeech	960 h	2.14	5.31	1.15	2.78
wav2vec large	Librispeech	960 h	2.11	5.10	0.99	2.43

Table 1: Replacing log-mel filterbanks (Baseline) by pre-trained embeddings improves WSJ performance on test (nov92) and validation (nov93dev) in terms of both LER and WER. We evaluate pre-training on the acoustic data of part of clean and full Librispeech as well as the combination of all of them. † indicates results with phoneme-based models.

vq-wav2vec

- * 引入两种可微的 discrete representations 方法
  - * 引入CodeBook
  - K-means
  - * GUMBEL-SOFTMAX

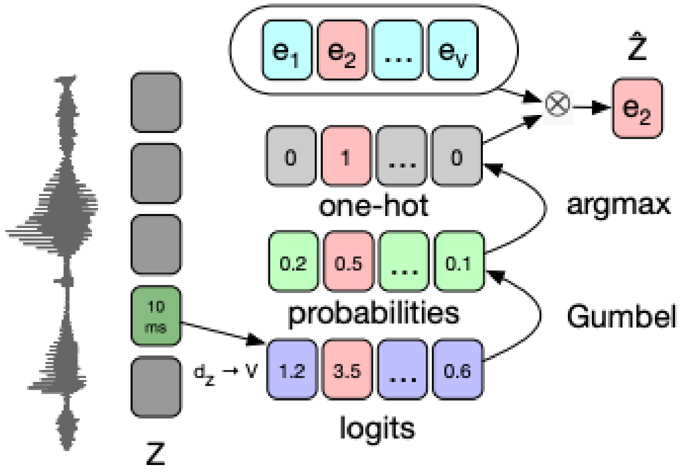
- * 下游用 bert 作为语义网络



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

GUMBEL-SOFTMAX

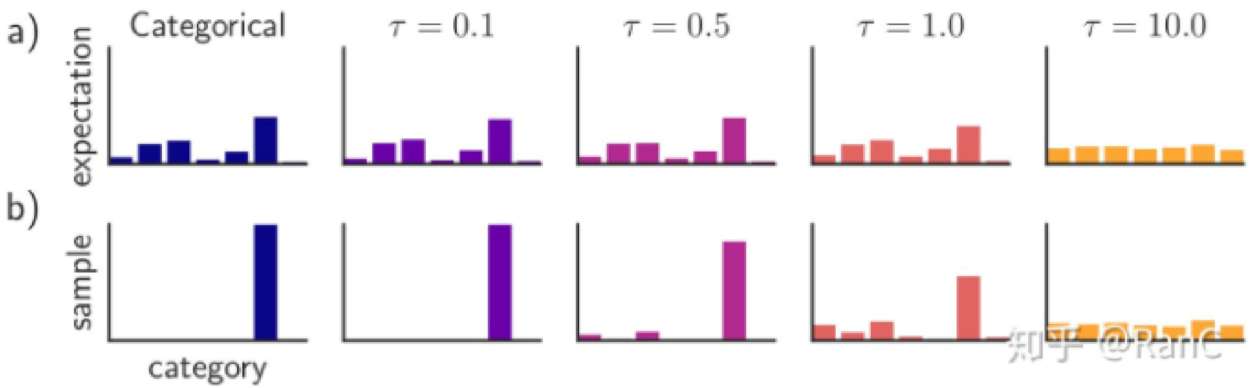
- 通过对Feature encoder层的变换，生成接近于onehot矩阵的分布，同时保证可导，进一步可对codebook进行选择



$$p_j = \frac{\exp(l_j + v_j)/\tau}{\sum_{k=1}^{V} \exp(l_k + v_k)/\tau},$$

- 当 τ 越小（如 0.1），代入 (2) 计算可知采样结果 $\mathbf{y}$ 就越倾向于接近一个真正的 one-hot vector，在本例中可能取值为 $[0.01; 0.01; 0.01; 0.97]$，类比真正的 one-hot vector $[0; 0; 0; 1]$
- 当 τ 越大（如 10.0），采样结果 $\mathbf{y}$ 在每一维度上的取值就越相似，导致 $y_i \approx 1/k$

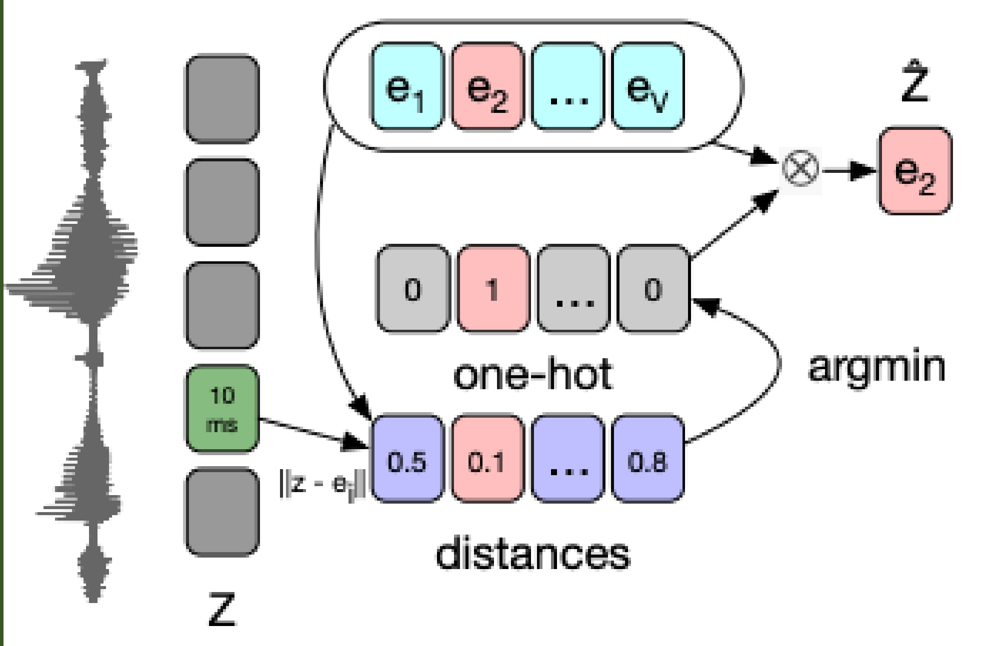
关于更多 τ 的讨论不妨直接看文中给出的实验结果，如下图：



K-MEANS

- 通过codebook与Z的比较生成欧氏距离向量，取最小值argmin 生成onehot矩阵

- 损失函数如图所示，在wav2vec对比损失基础上，增加两项，分别是固定codebook 仅关注Z和固定Z仅关注codebook



$$\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k^{\text{wav2vec}} + \left(\|\text{sg}(\mathbf{z}) - \hat{\mathbf{z}}\|^2 + \gamma \|\mathbf{z} - \text{sg}(\hat{\mathbf{z}})\|^2 \right)$$

CodeBook

- 类似于 embedding 结构，ont-hot 输入 vector 输出
- 可学习

灵魂拷问

- 为什么要使用 embedding 结构 ？
- 如果单纯只是为了降维，为什么不用其他网络结构 ？

ASR task language model

We consider two types of language models (LM): a 4-gram model and a Transformer trained on the Librispeech LM corpus.

- wav2vec 作用为编码（encoding）其中的 transformer 结构对语音进行表征
- ASR task 中的 transformer 语言模型，指的是通过MLM等 NLP 预训练方式得到的模型，训练过程音频不参与

vq-wav2vec 模型表现

	nov93dev		nov92	
	LER	WER	LER	WER
Deep Speech 2 (12K h labeled speech; Amodei et al., 2016)	-	4.42	-	3.1
Trainable frontend (Zeghidour et al., 2018)	-	6.8	-	3.5
Lattice-free MMI (Hadian et al., 2018)	-	5.66†	-	2.8†
Supervised transfer-learning (Ghahremani et al., 2017)	-	4.99†	-	2.53†
NO LM				
Baseline (log-mel)	6.28	19.46	4.14	13.93
wav2vec (Schneider et al., 2019)	5.07	16.24	3.26	11.20
vq-wav2vec Gumbel	7.04	20.44	4.51	14.67
+ BERT base	**4.13**	**13.40**	**2.62**	**9.39**
4-GRAM LM (Heafield et al., 2013)				
Baseline (log-mel)	3.32	8.57	2.19	5.64
wav2vec (Schneider et al., 2019)	2.73	6.96	1.57	4.32
vq-wav2vec Gumbel	3.93	9.55	2.40	6.10
+ BERT base	**2.41**	**6.28**	**1.26**	**3.62**
CHAR CONVLM (Likhomanenko et al., 2019)				
Baseline (log-mel)	2.77	6.67	1.53	3.46
wav2vec (Schneider et al., 2019)	2.11	5.10	0.99	2.43
vq-wav2vec Gumbel + BERT base	**1.79**	**4.46**	**0.93**	**2.34**

Table 1: WSJ accuracy of vq-wav2vec on the development (nov93dev) and test set (nov92) in terms of letter error rate (LER) and word error rate (WER) without language modeling (No LM), a 4-gram LM and a character convolutional LM. vq-wav2vec with BERT pre-training improves over the best wav2vec model (Schneider et al., 2019).

	nov93dev		nov92	
	LER	WER	LER	WER
NO LM				
wav2vec (Schneider et al., 2019)	5.07	16.24	3.26	11.20
vq-wav2vec Gumbel	7.04	20.44	4.51	14.67
+ BERT small	4.52	14.14	2.81	9.69
vq-wav2vec k-means (39M codewords)	5.41	17.11	3.63	12.17
vq-wav2vec k-means	7.33	21.64	4.72	15.17
+ BERT small	4.31	13.87	2.70	9.62
4-GRAM LM (Heafield et al., 2013)				
wav2vec (Schneider et al., 2019)	2.73	6.96	1.57	4.32
vq-wav2vec Gumbel	3.93	9.55	2.40	6.10
+ BERT small	2.67	6.67	1.46	4.09
vq-wav2vec k-means (39M codewords)	3.05	7.74	1.71	4.82
vq-wav2vec k-means	4.37	10.26	2.28	5.71
+ BERT small	2.60	6.62	1.45	4.08

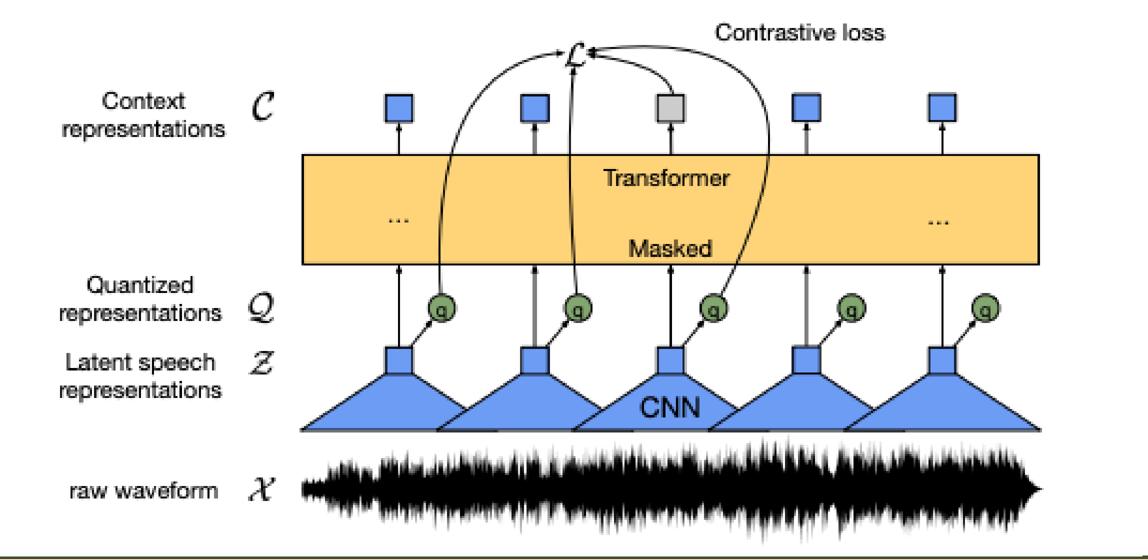
Table 2: Comparison of Gumbel-Softmax and k-means vector quantization on WSJ (cf. Table 1).

基于 WSJ 对比

gumbel-softmax 和 k-means对比

wav2vec2.0

- * 端到端 transformer 语义网络
- * 同时采用对比损失和多样性损失
- 精调负采样窗口和起始位
- 沿用 Gumbel softmax 对codebook 进行选择



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

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

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

Masking

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

- 任意位置置作为负样本（Masking）开始的概率为0.065
- masking 长度为 10 time-steps
- masking 有重叠

模型表现

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

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

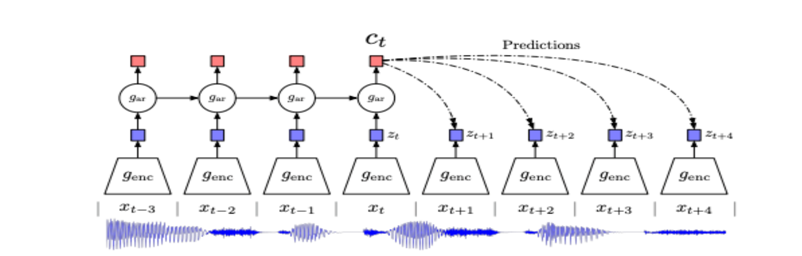
Model	Unlabeled data	LM	dev		test	
			clean	other	clean	other
Supervised						
CTC Transf [51]	-	CLM+Transf.	2.20	4.94	2.47	5.45
S2S Transf. [51]	-	CLM+Transf.	2.10	4.79	2.33	5.17
Transf. Transducer [60]	-	Transf.	-	-	2.0	4.6
ContextNet [17]	-	LSTM	1.9	3.9	1.9	4.1
Conformer [15]	-	LSTM	2.1	4.3	1.9	3.9
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

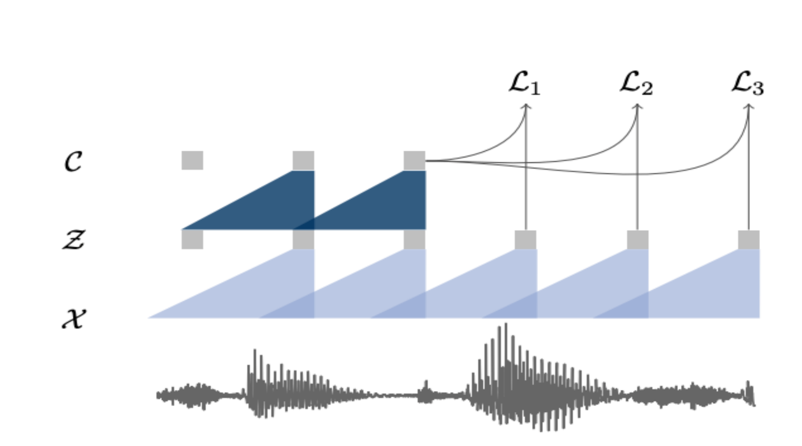
错误样本分析与启发

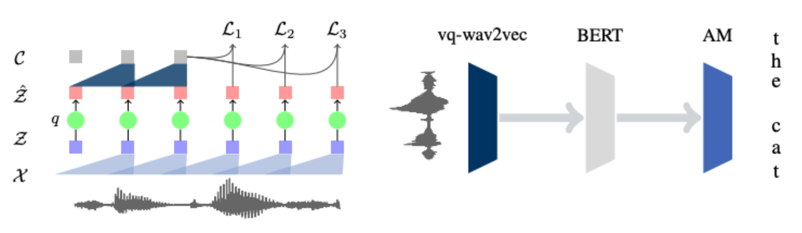
10m LARGE LV-60k	1h LARGE LV-60k	10h LARGE LV-60k	100h LARGE LV-60k	960h LARGE LV-60k	960h LARGE from scratch
all → al (181)	too → to (26)	in → and (15)	a → the (13)	a → the (12)	and → in (20)
are → ar (115)	until → untill (24)	a → the (11)	and → in (10)	and → in (9)	a → the (16)
will → wil (100)	new → knew (22)	o → oh (10)	in → and (10)	macklewain → mackelwaine (7)	in → and (13)
you → yo (90)	door → dor (18)	and → in (9)	o → oh (8)	in → and (6)	the → a (10)
one → on (89)	says → sais (18)	mode → mod (9)	minnetaki → minnitaki (7)	o → oh (6)	in → an (8)
two → to (81)	soul → sol (17)	ursus → ersus (9)	randal → randall (7)	bozzle → bosell (5)	and → an (5)
well → wel (80)	bread → bred (16)	tom → tome (8)	christie → cristy (6)	criss → chris (5)	clarke → clark (4)
been → ben (73)	poor → pore (16)	randal → randol (7)	macklewain → mackelwane (6)	bozzle → bosel (4)	grethel → gretel (4)
upon → apon (73)	a → the (13)	the → a (7)	randal → randoll (6)	clarke → clark (4)	macklewain → mackelwaine (4)
good → god (67)	either → ither (13)	color → colour (6)	bozzle → bosall (5)	colored → coloured (4)	this → the (4)
see → se (66)	food → fud (13)	flour → flower (6)	kaliko → calico (5)	grethel → gretel (4)	an → and (3)
we → whe (60)	doubt → dout (12)	phoebe → feeby (6)	trevelyan → trevelian (5)	lige → lyge (4)	anyone → one (3)
little → litle (54)	earth → erth (12)	an → and (5)	an → and (4)	the → a (4)	bozzle → basell (3)
great → grate (53)	led → lead (12)	cucumbers → cucombers (5)	and → an (4)	and → an (3)	buns → bunds (3)
your → yor (53)	sea → see (12)	egg → eg (5)	anyone → one (4)	ann → marianne (3)	carrie → carry (3)
could → coud (51)	thee → the (12)	macklewain → macklewaine (5)	bozzle → bozall (4)	butte → bute (3)	criss → chris (3)
here → hear (51)	tom → tome (12)	magpie → magpi (5)	clarke → clark (4)	color → colour (3)	he's → is (3)
know → now (45)	add → ad (11)	milner → millner (5)	gryce → grice (4)	deucalion → ducalion (3)	his → is (3)
there → ther (45)	good → god (11)	stacy → staci (5)	i'm → am (4)	forcemeat → meat (3)	honor → honour (3)
three → thre (45)	heaven → heven (11)	trevelyan → trevellion (5)	in → ind (4)	gryce → grice (3)	lattimer → latimer (3)
still → stil (42)	mary → marry (11)	verloc → verlock (5)	letty → lettie (4)	honor → honour (3)	millet → mellet (3)
off → of (40)	randal → randel (11)	ann → an (4)	phoebe → phebe (4)	kearny → kirney (3)	pyncheon → pension (3)
don't → dont (37)	answered → ansered (10)	anyone → one (4)	the → a (4)	nuova → noiva (3)	tad → ted (3)
shall → shal (36)	blood → blod (10)	apartment → appartment (4)	ann → anne (3)	thing → anything (3)	thing → anything (3)
little → litl (35)	bozzle → bosel (10)	basin → bason (4)	awhile → while (3)	this → the (3)	trevelyan → trevelian (3)

观点：英文单词中音素与字符非一一对应，但是中文几乎全为一一对应音素，是否在10min训练更加有效？

网络结构演变

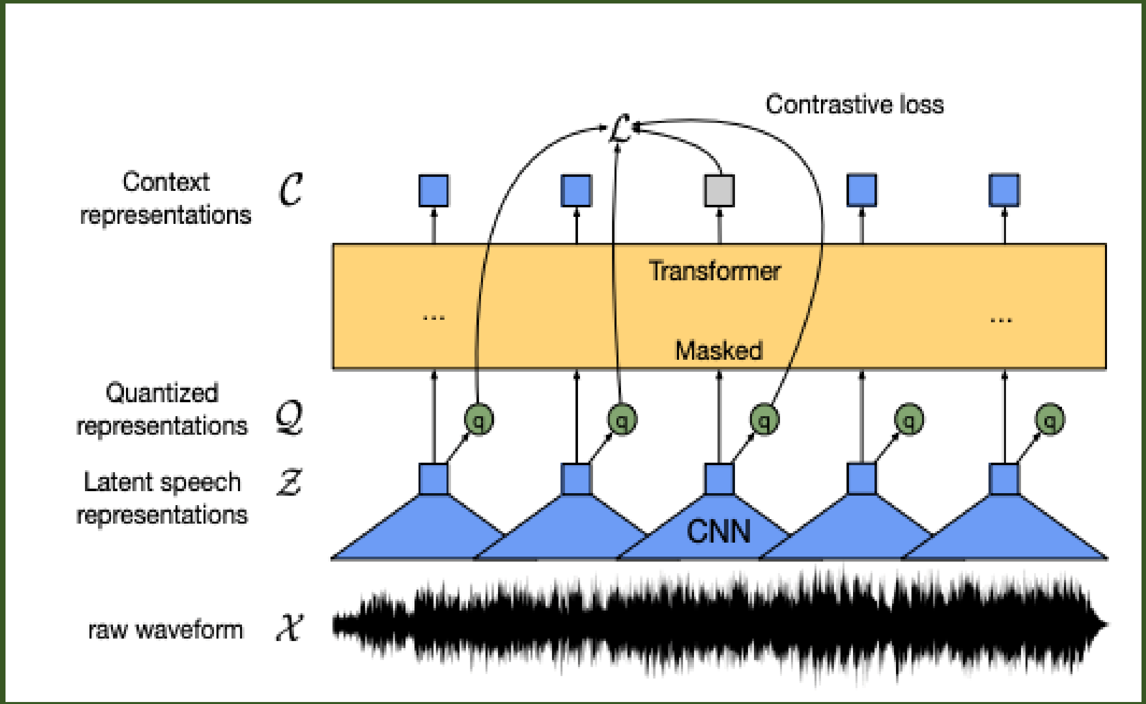






(a) vq-wav2vec

(b) Discretized speech training pipeline



Negative samples 补充

How do we select negative samples?

- Sampling randomly from the entire dataset and encode them, which is not efficient.
- Sampling randomly from the same minibatch.
- Sampling from the same sequence farther away from the target
- Sampling from the other examples of mini-batch.

Of course, we can use a mix of both from the 2nd point.

Which one to choose and when?

- Suppose we are drawing the negative samples from other examples of mini-batch. The dataset is a multi-speaker dataset then negative samples most likely to be of different speakers. The model will try to learn the representations of speaker identity.

- If we draw the negative samples from the same sequence, then the model will try to learn the representations that will capture phonetic information, prosody, etc. Models trained with this objective can be used for speech recognition downstream tasks.

实现

抱　　　　脸

https://huggingface.co

```
python transformers/examples/pytorch/speech-recognition/run_speech_recognition_ctc.py \
    --dataset_name="common_voice" \
    --model_name_or_path="facebook/wav2vec2-large-xlsr-53" \
    --dataset_config_name="tr" \
    --output_dir="./wav2vec2-common_voice-tr-demo" \
    --overwrite_output_dir \
    --num_train_epochs="15" \
    --per_device_train_batch_size="8" \
    --gradient_accumulation_steps="1" \
    --learning_rate="3e-4" \
    --warmup_steps="500" \
    --evaluation_strategy="steps" \
    --text_column_name="sentence" \
    --save_steps="400" \
    --eval_steps="100" \
    --layerdrop="0.0" \
    --save_total_limit="3" \
    --freeze_feature_extractor \
    --gradient_checkpointing \
    --chars_to_ignore , ? . ! - \; \: \"  “ %  ‘ ”  � \
    --dataloader_num_workers=2 \
    --do_train --do_eval
```

```
{'eval_loss': 0.3913697898387909, 'eval_wer': 0.33581860892656523, 'eval_runtime': 82.8417, 'eval_samples_per_second': 19.881, 'eval_steps_per_second': 2.487, 'epoch': 14.94}
100% 6500/6525 [3:41:03<00:29,  1.17s/it]
100% 206/206 [01:21<00:00,  2.95it/s]
100% 6525/6525 [3:41:31<00:00,  1.01s/it]

Training completed. Do not forget to share your model on huggingface.co/models =)


{'train_runtime': 13291.5575, 'train_samples_per_second': 3.925, 'train_steps_per_second': 0.491, 'train_loss': 0.4483499853821093, 'epoch': 15.0}
100% 6525/6525 [3:41:31<00:00,  2.04s/it]
Saving model checkpoint to ./wav2vec2-common_voice-tr-demo
Configuration saved in ./wav2vec2-common_voice-tr-demo/config.json
Model weights saved in ./wav2vec2-common_voice-tr-demo/pytorch_model.bin
Configuration saved in ./wav2vec2-common_voice-tr-demo/preprocessor_config.json
***** train metrics *****
  epoch                    =       15.0
  train_loss               =     0.4483
  train_runtime            = 3:41:31.55
  train_samples            =       3478
  train_samples_per_second =      3.925
  train_steps_per_second   =      0.491
11/22/2021 10:55:57 - INFO - __main__ - *** Evaluate ***
***** Running Evaluation *****
  Num examples = 1647
  Batch size = 8
[W pthreadpool-cpp.cc:90] Warning: Leaking Caffe2 thread-pool after fork. (function pthreadpool)
[W pthreadpool-cpp.cc:90] Warning: Leaking Caffe2 thread-pool after fork. (function pthreadpool)
100% 206/206 [01:21<00:00,  2.53it/s]
***** eval metrics *****
  epoch                   =       15.0
  eval_loss               =     0.3912
  eval_runtime            = 0:01:22.88
  eval_samples            =       1647
  eval_samples_per_second =     19.871
  eval_steps_per_second   =      2.485
```

eval_wer = 0.336

```
Dropping the following result as it does not have all the necessary fields:
{'dataset': {'name': 'COMMON_VOICE - TR', 'type': 'common_voice', 'args': 'Config: tr, Training split: train+validation, Eval split: test'}}
```

Openslr

https://www.openslr.org/resources.php

Papers with Code

https://paperswithcode.com

HAVE A TRY！！！

下一步的研究计划：

1. 中文ASR数据集测试
2. 中文语音数据集预训练
3. 与现有榜单对比差距

Thanks