Unsupervised Pre-training for Speech Recognition (wav2vec and 2.0)

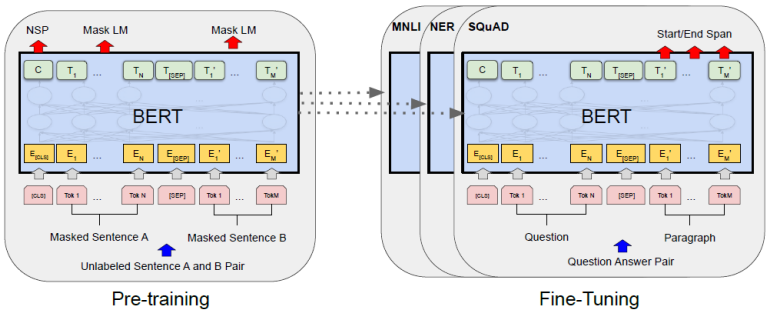
성원용

Pre-training

- Unsupervised (self-supervised) 방법으로 좋은 embedding 공간을 구축
- Text 등 다른 domain에서 이미 많이 사용된다
  - Word2vec: skip-gram 등을 이용하여 단어의 embedding 을 학습
  - BERT: transformer 구조를 이용하여 context-aware embedding 을 구축
  - ViLBERT: Vision을 위한 BERT
- BERT 의 영향이 가장 크다

BERT Masked LM training

- BERT (Bidirectional Encoder Representations from Transformers)
- We use vast amount of plain text (no labeling).
- By Pre-training, we get neural networks that can understand text (Sentence embedding, contextualized word-embedding, and so on)
- For fine-tuning, we only use a limited amount of labeled data.



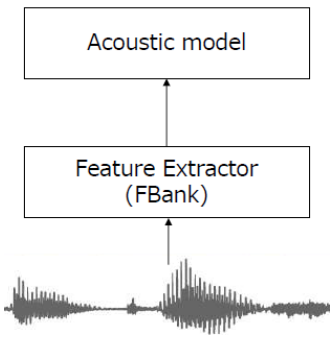
Masked LM training (Cloze task)

- The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In all of our experiments, we mask 15% of all WordPiece tokens in each sequence at random.
- The loss is calculated as the difference between the output probability distributions for each output 'token', and the true one-hot encoded *labels*.
- In contrast to denoising auto-encoders (Vincent et al., 2008), we only predict the masked words rather than reconstructing the entire input.

Wav2vec의 목적

- 종래의 HMM 을 이용한 음성인식
  - Hand-labeled feature를 이용
  - Filter-bank, MFCC, +delta, + normalization, +VTLN, +fMLLR
  - 일반적으로 작은 training data
- End-to-end 음성인식 방법 (CTC, LAS)
  - 좋은 성능을 위해 많은 labeled data (보통 10,000 시간 단위)
- Resource limited language 를 위한 ASR 설계: Unlabeled data 를 이용하는 self-supervised 방법의 pre-training 방법에 의하여 좋은 특징을 추출. 이 추출된 특징을 이용하고 약간의 labeled data를 이용 최적학습

종래의 acoustic model



- Feature extractor – Mel frequency filter bank, MFCC (mel-frequency cepstral coefficients) 등 사람 (전문가)이 설계한 feature
- 이 것을 unlabeled data를 이용하여 자동으로 생성하자.
- Representation learning

오늘 다룰 논문 3편

- Representation learning with contrastive predictive coding (DeepMind)
- Wav2vec: Unsupervised pretraining for speech recognition (Facebook AI)
- VQ-WAV2VEC: Self-supervised learning of discrete speech representations (Facebook)
- Wav2vec 2.0 (Facebook AI)

논문 1: Representation learning with contrastive predictive coding (DeepMind)

- Self-supervised: Encoder의 출력에서 context를 고려하여 미래의 embedding을 예측, 좋은 feature를 이용해야 이 것이 가능할 것이다.
- 입력은 음성 waveform (filter bank 가 아니다)
- 특징(feature) + linear classifier 에 의한 음소인식

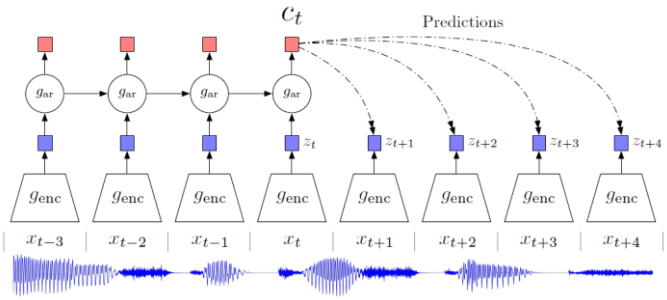


Figure 1: Overview of Contrastive Predictive Coding, the proposed representation learning appro

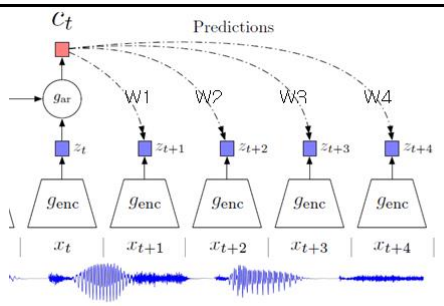
- x -> z -> c
  In text, x is an input text, z is an non-contextualized embedding (word-vector), c is the contextualized vector.
- z is the frame-level embedding, c is the multiple frame embedding (contextualized)
- How to design z? Maximize the mutual information between x and c

$$I(x;c) = \sum_{x,c} p(x,c) \log \frac{p(x|c)}{p(x)}.$$

Mutual information

- Mutual information is one of many quantities that measures how much one random variables tells us about another.

$$I(X;Y) = \sum_{x,\,y} p(x,\,y) log(\frac{p(x,\,y)}{p(x)p(y)})$$



edictive Coding, the proposed representation learning appro

$$f_k(x_{t+k}, c_t) \propto \frac{p(x_{t+k}|c_t)}{p(x_{t+k})} \tag{2}$$

where $\propto$ stands for 'proportional to' (i.e. up to a multiplicative constant). Note that the density ratio f can be unnormalized (does not have to integrate to 1). Although any positive real score can be used here, we use a simple log-bilinear model:

$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right), \tag{3}$$

In our experiments a linear transformation $W_k^T c_t$ is used for the prediction with a different W_k for every step k. Alternatively, non-linear networks or recurrent neural networks could be used.

By using a density ratio $f(x_{t+k}, c_t)$ and inferring z_{t+k} with an encoder, we relieve the model from modeling the high dimensional distribution x_{t_k}. Although we cannot evaluate $p(x)$ or $p(x|c)$ directly,

Training for CPC

- $W_k c_t$ 와 z^T_{t+K} 의 inner product 를 크게 만듦.

$$f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right),$$

we will call InfoNCE. Given a set $X = \{x_1, \ldots x_N\}$ of N random samples containing one positive sample from $p(x_{t+k}|c_t)$ and $N-1$ negative samples from the 'proposal' distribution $p(x_{t+k})$, we optimize:

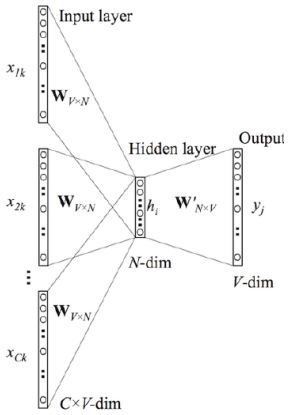
$$\mathcal{L}_N = -\underset{X}{\mathbb{E}}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right] \tag{4}$$

Negative sampling

- Softmax computation of large vocabulary problem.

$$\mathbf{p(z)_i} = \frac{\mathbf{e^{z_i}}}{\sum_{\mathbf{j=1}}^{\mathbf{|V|}} \mathbf{e^{z_j}}}$$

- Computing V is too costly, and most of them will have almost zero values. So, sample only K (작은 숫자), and updates only them.
- This is approximation, but can speed-up the training process



NCE(Noise Contrastive Estimation)

- Estimate the denominator term, and makes it a binary classification problem.
- Assume the distribution of negative samples, Q
- Sample k negative samples, and estimate the denominator with Q, and makes it a binary classification problem (logistic regression)

$$\mathbf{BCE} = \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i}}^{\mathbf{N}} \mathbf{l_i log}(\hat{\mathbf{p}}_\mathbf{i}) + (\mathbf{1} - \mathbf{l_i})\mathbf{log}(\mathbf{1} - \hat{\mathbf{p}}_\mathbf{i})$$

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.

Method	ACC
#steps predicted	
2 steps	28.5
4 steps	57.6
8 steps	63.6
12 steps	64.6
16 steps	63.8
Negative samples from	
Mixed speaker	64.6
Same speaker	65.5
Mixed speaker (excl.)	57.3
Same speaker (excl.)	64.6
Current sequence only	65.2

Table 2: LibriSpeech phone classification ablation experiments. More details can be found in Section 3.1.

- We use five convolutional layers with strides [5, 4, 2, 2, 2], filter-sizes [10, 8, 4, 4, 4] and 512 hidden units with ReLU activations. The total downsampling factor of the network is 160 so that there is a feature vector for every 10ms of speech, which is also the rate of the phoneme sequence labels obtained with Kaldi. We then use a GRU RNN [17] for the autoregressive part of the model, gar with 256 dimensional hidden state. The output of the GRU at every timestep is used as the context c from which we predict 12 timesteps in the future using the contrastive loss

Application of CPC

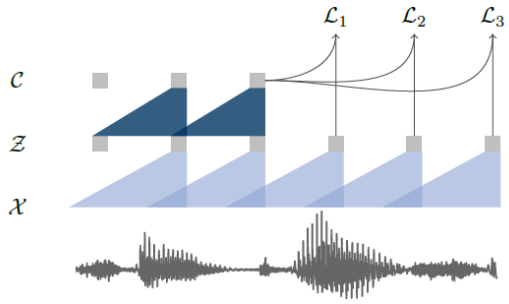
- 응용: 음성, 이미지, …
- Both z_t and c_t could be used as representation for downstream tasks.
- The autoregressive model output c_t can be used if extra context from the past is useful. One such example is speech recognition, where the receptive field of z_t might not contain enough information to capture phonetic content. In other cases, where no additional context is required, z_t might instead be better. (음성은 당연히 c_t 가 더 좋다. c_t 는 contextualized feature)

논문 2: Wav2vec network

WAV2VEC: UNSUPERVISED PRE-TRAINING FOR SPEECH RECOGNITION

Steffen Schneider, Alexei Baevski, Ronan Collobert, Michael Auli
Facebook AI Research

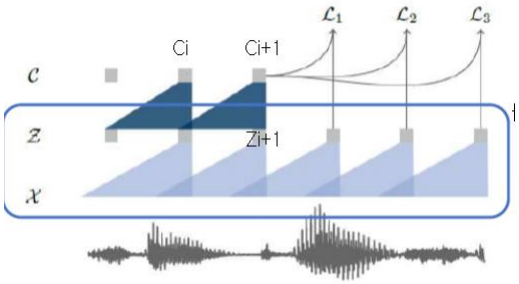
- CPC 구조의 음성인식에의 적용
- Unlabeled speech 를 이용한 pre-training
- Labeled data (WSJ 등)를 이용한 fine-tuning



- Encoder: The output of the encoder is a low frequency feature representation zi, which encodes about **30 ms** of 16 kHz of audio and the striding results in representations z_i every 10ms.
- Context network g : Z -> C to the output of the encoder network to mix multiple latent representations z_i. The context network has nine layers with kernel size three and stride one. The total receptive field of the context network is about **210 ms** (길이가 길지만 매우 긴 것은 아니다).
- The layers in both the encoder and context networks consist of a causal convolution with 512 channels, a group normalization layer and a ReLU nonlinearity.

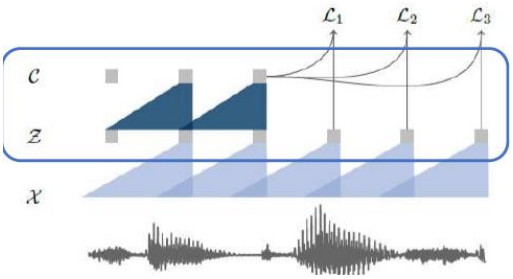
Encoder network (f, feature extraction)

- 음성의 waveform(파형. X)에서 hidden representation (embedding) Z 로 변환.
- One-dimensional convolutional network
  - 5 layers •
  - Kernel: (10,8,4,4,4)
  - Strides: (5, 4,2,2,2)

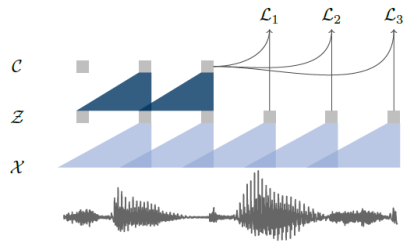


Context network (aggregation)

- One-dimensional convolutional network that handles context information of Embedding
  - 9 layers
  - Kernel: 3
  - Strides: 1
  - 210ms window in total



Loss and training



We train the model to distinguish a sample $\mathbf{z}_{i+k}$ that is k steps in the future from distractor samples $\tilde{\mathbf{z}}$ drawn from a proposal distribution p_n, by minimizing the contrastive loss for each step $k = 1, \ldots, K$:

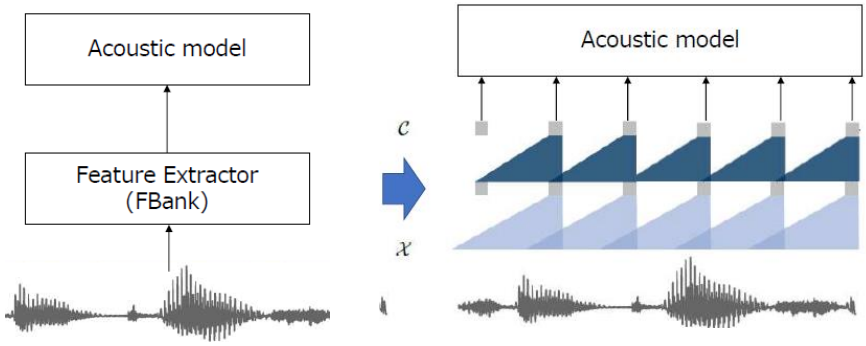
$$\mathcal{L}_k = -\sum_{i=1}^{T-k} \left(\log \sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i)) + \lambda \mathop{\mathbb{E}}_{\tilde{\mathbf{z}} \sim p_n} \left[\log \sigma(-\tilde{\mathbf{z}}^\top h_k(\mathbf{c}_i)) \right] \right) \quad (1)$$

where we denote the sigmoid $\sigma(x) = 1/(1+\exp(-x))$, and where $\sigma(\mathbf{z}_{i+k}^\top h_k(\mathbf{c}_i))$ is the probability of $\mathbf{z}_{i+k}$ being the true sample. We consider a step-specific affine transformation $h_k(\mathbf{c}_i) = W_k \mathbf{c}_i + \mathbf{b}_k$ for each step k, that is applied to $\mathbf{c}_i$ (van den Oord et al., 2018). We optimize the loss $\mathcal{L} = \sum_{k=1}^{K} \mathcal{L}_k$, summing (1) over different step sizes. In practice, we approximate the expectation by sampling ten negatives examples by uniformly choosing distractors from each audio sequence, i.e., $p_n(\mathbf{z}) = \frac{1}{T}$, where T is the sequence length and we set λ to the number of negatives.[2]

After training, we input the representations $\mathbf{c}_i$ produced by the context network to the acoustic model instead of log-mel filterbank features.

음성인식에의 응용

- Ci 를 filter-bank 등을 대치하여 acoustic model 을 만드는데 사용한다.



Corpus

- For phoneme recognition on TIMIT, the standard train, dev and test split where the training data contains just over three hours of audio data.
- Wall Street Journal (WSJ; Woodland et al. (1994)) comprises about 81 hours of transcribed audio data. We train on si284, validate on nov93dev and test on nov92.
- Librispeech contains a total of 960 hours of clean and noisy speech for training. For pre-training, we use either the full 81 hours of the WSJ corpus, an 80 hour subset of clean Librispeech, the full 960 hour Librispeech training set or a combination of all of them.
- To train the baseline acoustic model we compute 80 log-mel filterbank coefficients for a 25 ms sliding window with stride 10 ms.
- Final models are evaluated in terms of both word error rate (WER) and letter error rate (LER).

TIMIT fine-tuning (phoneme)

- The final representation is projected to a 39- dimensior phoneme probability.

negatives	dev PER	train time (h)
1	16.3	6.1
2	15.8	6.3
5	15.9	8.2
10	15.5	10.5
20	15.7	15.3

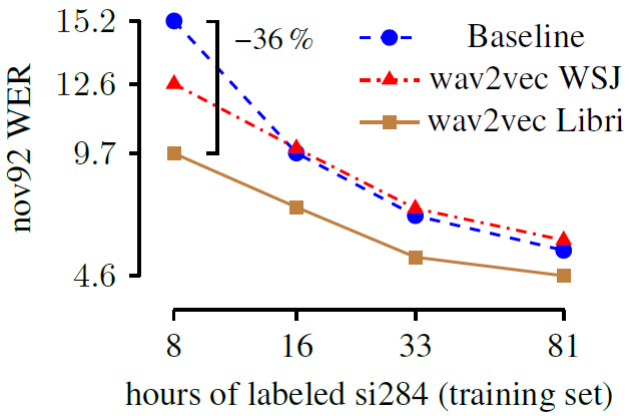
	dev	test
CNN + TD-filterbanks (Zeghidour et al., 2018a)	15.6	18.0
Li-GRU + MFCC (Ravanelli et al., 2018)	–	16.7 ± 0.26
Li-GRU + FBANK (Ravanelli et al., 2018)	–	15.8 ± 0.10
Li-GRU + fMLLR (Ravanelli et al., 2018)	–	14.9 ± 0.27
Baseline	16.9 ± 0.15	17.6 ± 0.11
wav2vec (Librispeech 80h)	15.5 ± 0.03	17.6 ± 0.12
wav2vec (Librispeech 960h)	13.6 ± 0.20	15.6 ± 0.23
wav2vec (Librispeech + WSJ)	$\mathbf{12.9 \pm 0.18}$	$\mathbf{14.7 \pm 0.42}$

Table 2: Results for phoneme recognition on TIMIT in terms of PER. All our models use the CNN-8L-PReLU-do0.7 architecture (Zeghidour et al., 2018a).

			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., 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

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

- WSJ – 31 grapheme-based (alphabet and repetition, ‘, ., |)



훈련

- Wav2Vec은 Word2Vec처럼 해당 입력이 positive 쌍인지 negative 쌍인지 이진 분류(binary classification)하는 과정에서 학습. Positive 쌍은 그림1처럼 입력 음성의 i 번째 context representation (Ci) 와 i+1 번째 hidden representation (Zi+1). 네거티브 쌍은 입력 음성의 i 번째 context representation Ci 와 현재 배치의 다른 음성의 hidden representation들 가운데 랜덤으로 추출한 것.
- 학습이 진행될 수록 포지티브 쌍 관계의 representation은 벡터 공간에서 가까워지고, 네거티브 쌍은 멀어진다. 다시 말해 encoder network f 와 context network g 는 입력 음성의 다음 시퀀스가 무엇일지에 관한 정보를 음성 피처에 잘 녹여낼 수 있게 됩니다.

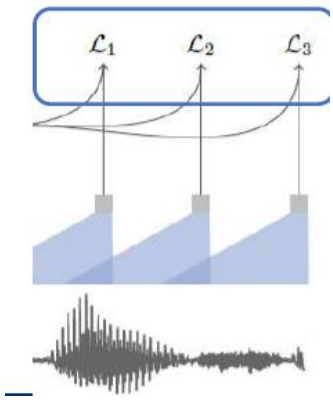
CPC objective function

$$\mathcal{L}_k = -\sum_i \underbrace{(\log \sigma(\boldsymbol{z}_{i+k}^\top h_k(\boldsymbol{c}_i))}_{①} + \underbrace{\lambda \mathbb{E}_{\tilde{\boldsymbol{z}} \sim p_n}[\log \sigma(-\tilde{\boldsymbol{z}}^\top h_k(\boldsymbol{c}_i))]}_{②}$$

$$h_k(\boldsymbol{c}_i) = W_k \boldsymbol{c}_i + b_k$$

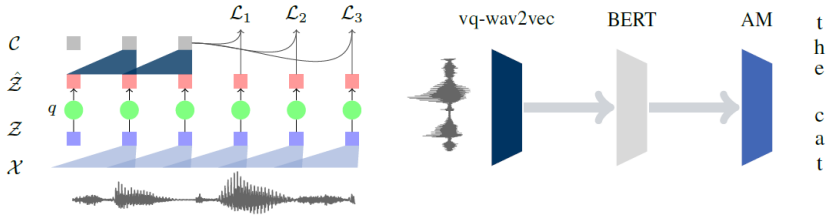
(1) 미래의 z_{i+k}와 상관관계가 높다면, $\mathcal{L}_k$가 작아진다.

(2) The correlation with negative examples
- Select 10 randomly from each data



논문 3: VQ-WAV2VEC: SELF-SUPERVISED LEARNING OF DISCRETE SPEECH REPRESENTATIONS

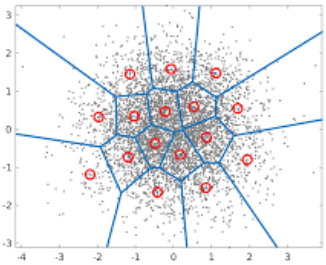
- 저자 : Alexei Baevski, Steffen Schneider, Michael Auli (Facebook)
- 양자화(Quantization) 한 후에 BERT의 방법을 이용
- wav2vec embedding을 양자화
- 양자화된 embedding을 BERT 의 입력으로 넣어서 특징을 추출



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

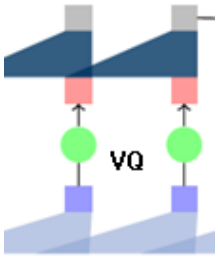
Vector quantization (<-> scalar quantization)

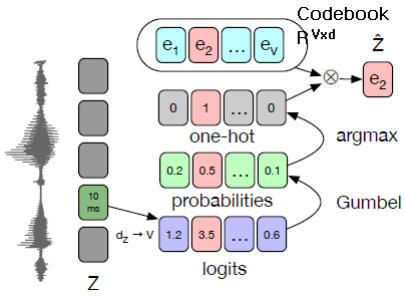
- It works by dividing a large set of points (vectors) into groups having approximately the same number of points closest to them. Each group is represented by its centroid point, as in k-means and some other clustering algorithms.
- 2-D vector quantization 의 예



Embedding의 양자화

- Using Gumbel softmax or K-means Gumbel softmax





Vector Quantization 모듈.

- 우선 z 를 선형변환해 logit를 만들고, 여기에 Gumbel Softmax와 argmax를 취해 one-hot 벡터를 만든다. 연속적인(continous) 변수 z 가 이산(discrete) 변수로 변환됐습니다. 이것이 바로 Vector Quantization.

- 이후 Embedding matrix를 내적해 z^ 를 만든다. 결과적으로는 Vector Quantization으로 V 개의 Embedding 가운데 e2 를 하나 선택.

- G개의 그룹을 사용. 실험 시 G=2

Gumbel softmax

- Codebook 훈련의 문제 – 양자화 때문에 gradient 가 계산이 안되어서 codebook의 변수를 훈련할 수 없다.
- Gumbel noise 를 섞으면 양자화 후의 분포가 일반 softmax 분포 비슷해진다.

of Jang et al. (2016). Given the dense representation $\mathbf{z}$, we apply a linear layer, followed by a ReLU and another linear which outputs $\mathbf{l} \in \mathbb{R}^V$ logits for the Gumbel-Softmax. At inference, we simply pick the largest index in l. At training, the output probabilities for choosing the j-th variable are

$$p_j = \frac{\exp(l_j + v_j)/\tau}{\sum_{k=1}^{V} \exp(l_k + v_k)/\tau}, \tag{2}$$

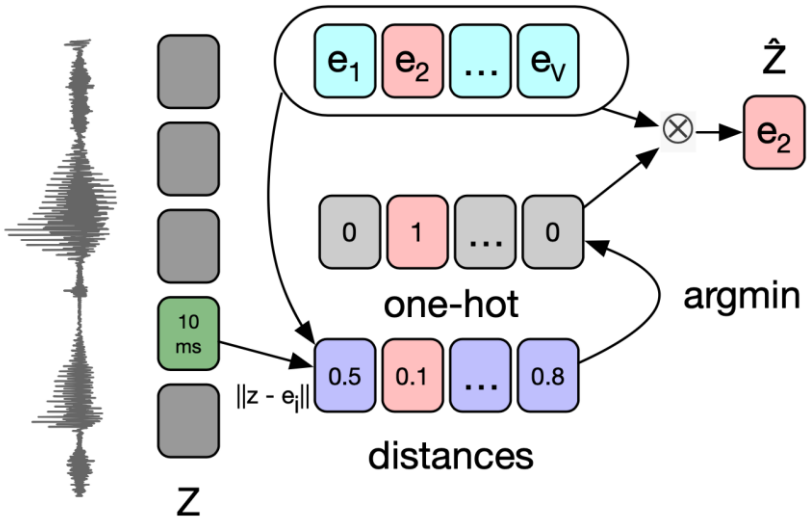
where $v = -\log(-\log(u))$ and u are uniform samples from $\mathcal{U}(0, 1)$. During the forward pass, $i = \text{argmax}_j p_j$ and in the backward pass, the true gradient of the Gumbel-Softmax outputs is used.

- Gumbel-Softmax Models. We use G = 2 groups and V = 320 latents per group and the linear layer projects the features produced by the encoder into G V = 640 logits. The Gumbel-Softmax produces a one-hot vector for each group G. The temperature is linearly annealed from 2 to 0.5 over the first 70% of updates and then kept constant at 0.5. This enables the model to learn which latents work best for each input before committing to a single latent. After training this model on 960h of Librispeech and quantizing the training dataset, we are left with 13.5k unique codewords combinations (out of V G = 102k possible codewords).

The dense feature vector $\mathbf{z} \in \mathbb{R}^d$ is first organized into multiple *groups* G into the matrix form $\mathbf{z}' \in \mathbb{R}^{G \times (d/G)}$. We then represent each row by an integer index, and hence can represent the full feature vector by the indices $\mathbf{i} \in [V]^G$, where V again denotes the possible number of *variables* for this particular group and each element $\mathbf{i}_j$ corresponds to a fixed codebook vector. For each of the G groups, we apply either one of the two VQ approaches (§3.1 and §3.2).

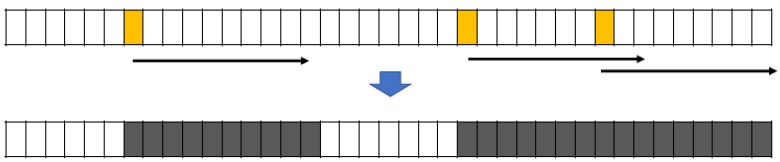
The codebook itself can be initialized in two possible ways: Codebook variables can be shared across groups, i.e., a particular index in group j would reference the same vector as the same index in group j'. This yields a codebook $\mathbf{e} \in \mathbb{R}^{V \times (d/G)}$. In contrast, not sharing the codebook variables yields a codebook of size $\mathbf{e} \in \mathbb{R}^{V \times G \times (d/G)}$. In practise, we observe that sharing the codebook variables generally yields competitive results to a non-shared representation.

K-means clustering



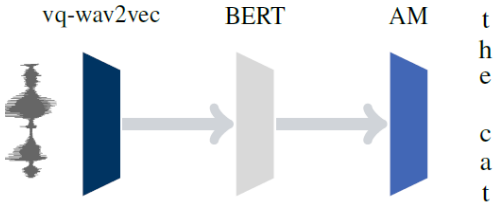
Use of BERT

BERT is learned by masked token prediction. However, it is too easy to estimate the embedding of 10ms if you do it normally, so set the starting point with a probability of p = 0.05 and set 10 frames from there. Mask.



음성인식에의 이용

- BERT의 출력을 음성인식의 feature-vector 에 이용
- BERT small 을 이용한 실험 (BERT large 너무 크다)



(b) Discretized speech training pipeline

성능비교 (WSJ corpus)

- Quantization에 의한 성능 하락을 BERT를 이용 복구.

	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**

BERT mask length and probabilities for TIMIT training

M	dev	test
1	14.94	17.38
5	13.62	15.78
10	12.65	15.28
20	13.04	15.56
30	13.18	15.64

(a) Mask length.

p	dev	test
0.015	12.65	15.28
0.020	12.51	14.43
0.025	12.16	13.96
0.030	11.68	14.48
0.050	11.45	13.62

(b) Mask probabilities.

able 5: TIMIT PER for (a) different mask sizes M with $pM = 0.15$ in BERT training and (b) mask robabilities p for a fixed mask length $M = 10$.

	dev PER	test PER
CNN + TD-filterbanks (Zeghidour et al., 2018)	15.6	18.0
Li-GRU + fMLLR (Ravanelli et al., 2018)	–	14.9
wav2vec (Schneider et al., 2019)	12.9	14.7
Baseline (log-mel)	16.9	17.6
vq-wav2vec, Gumbel	15.34	17.78
+ BERT small	9.64	**11.64**
vq-wav2vec, k-means	15.65	18.73
+ BERT small	9.80	11.40

Table 3: TIMIT phoneme recognition in terms of phoneme error rate (PER). All our mode CNN-8L-PReLU-do0.7 architecture (Zeghidour et al., 2018).

논문 4: Wav2vec 2.0

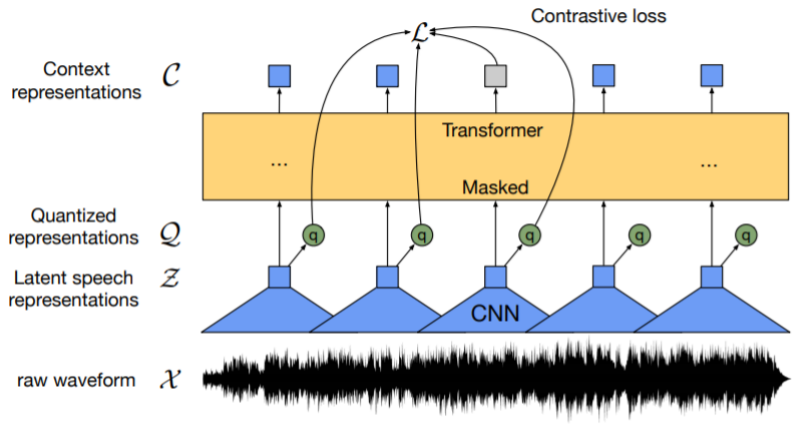
wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations

Alexei Baevski Henry Zhou Abdelrahman Mohamed Michael Auli

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

Facebook AI

- Transformer-based
- Design ASR with only 10 minutes of labeled data (low resource speech recognition) and pre-training on 53K hours of unlabeled data (5.7/10.1 WER on clean/noisy Librispeech testset)



- Vq-wav2vec 과 비교할 때 많이 간단해짐.
- Context network이 생략되고, quantized latent representation 이 직접 Transformer 로 들어감 (Transformer 가 context network 역할 도 함).
- 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.

Feature encoder and Transformer

- 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.
- 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.

Quantization

- Multiple codebooks and concatenate the results
- Gumbel softmax for choosing discrete codebook entries in a differentiable way.

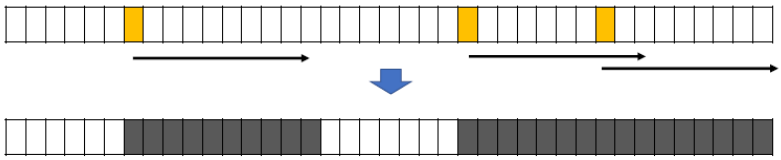
representations [5]. Product quantization amounts to choosing quantized representations from multiple codebooks and concatenating them. Given G codebooks, or groups, with V entries $e \in \mathbb{R}^{V \times d/G}$, we choose one entry from each codebook and concatenate the resulting vectors $e_1, \ldots,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

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

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.

Training

- 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.



Training loss

- Contrastive loss for masking and diversity loss for codebook entry diversity.

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

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

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

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

Contrastive loss

- Given context network output c_t centered over masked time step t, the model needs to identify the true quantized latent speech representation qt in a set of K+1 quantized candidate representations q~ =Q_t which includes qt and K distractors. Distractors are uniformly sampled from other masked time steps of the same utterance. The loss is defined as:

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

where we compute the cosine similarity sim(a, b) = $a^T b/||a||||b||$

c_t 는 가급적 q_t 에 가깝게, 그렇지만 다른 것 과는 멀게

Diversity loss and Penalty

- We encourage the equal use of the V entries in each of the G codebooks by maximizing the entropy of the averaged softmax distribution L over the codebook entries for each codebook p-g across a batch of utterances; the softmax distribution does not contain the gumbel noise nor a temperature.
- To stabilize the training, it is helpful to apply an L2 penalty to the activations of the final layer of the feature encoder but before the final layer normalization.

Pretraining

- 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

Fine-tuning

- Phoneme recognition task: adding a randomly initialized linear projection on top of the context network into C classes representing the vocabulary of the task [4].
- Grapheme output - 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

Datasets

- Unlabeled dataset for pretraining: Librispeech corpus [40] without transcriptions containing 960 hours of audio (LS-960) or the audio data from LibriVox (LV-60k).
- Finetuning – TIMIT and Librispeech

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

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

결론

- NLP에서 많이 이용되는 pre-training 방법을 응용
- Wav2vec – unlabeled data를 활용하여 pre-training 후 소량의 labeled data로 좋은 음성인식기 설계
- Vq-wav2vec – BERT 모델을 이용하여 성능을 높였다.
- Wav2vec 2.0 – VQ-wav2vec 을 간략화시켜서 context network 과 BERT 를 Transformer 구조로 대치하였다.