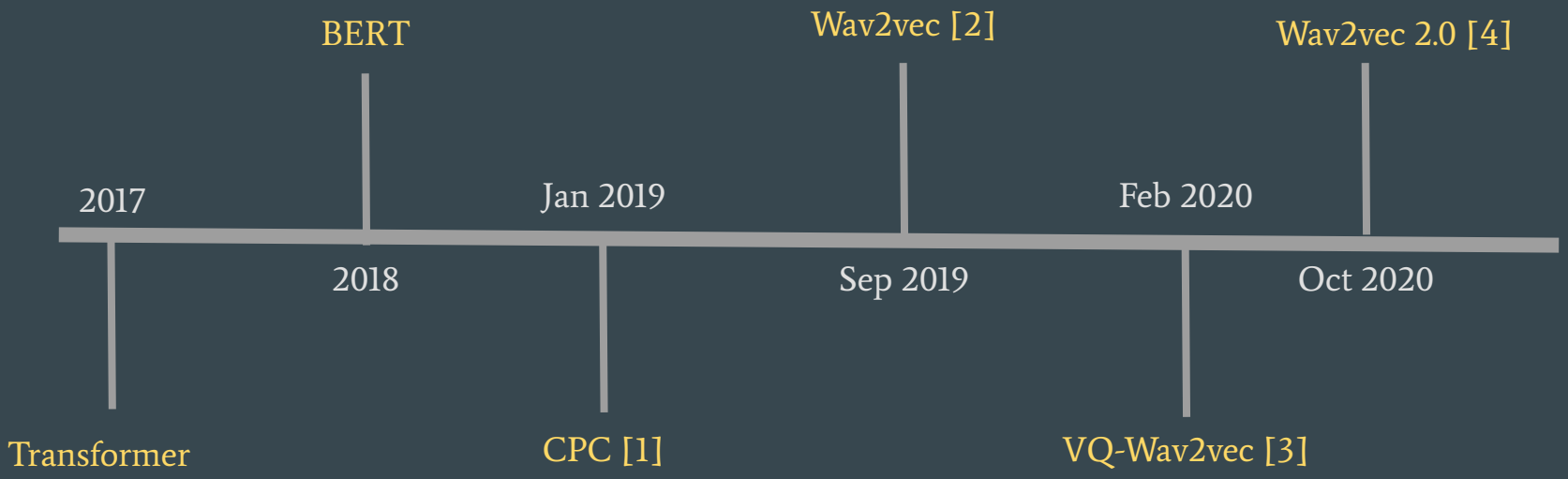
Wav2vec 2.0

Presentation by Mithilesh Vaidya, DAP Lab, EE IITB

Introduction

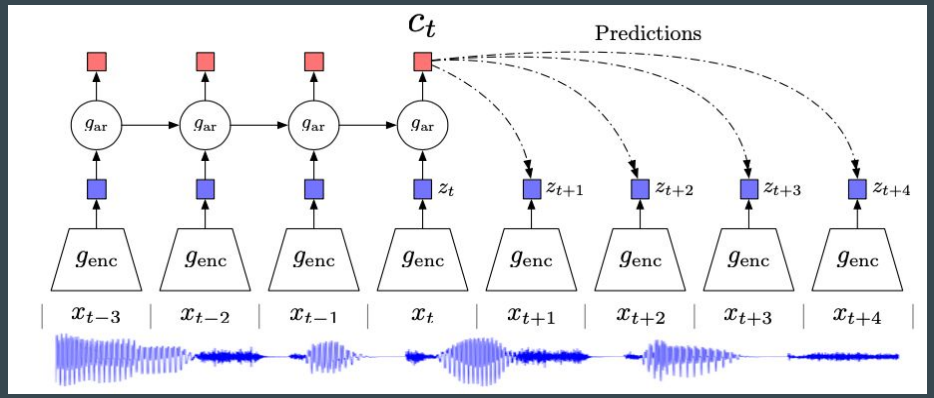
- Why?
  - Any deep learning task requires **significant** amount of **high-quality** data
  - Bottleneck for low-resource tasks
- Trend: Pre-train on large dataset + Fine-tune on task dataset
  - NLP: models are built on top of BERT (+variants)
    - **Pre-trained** transformer trained on gigantic datasets e.g. Wikipedia
    - Fine-tune an additional layer on top of BERT for given task
  - CV: ImageNet pre-trained feature extractor at input
- Wav2vec 2.0 (w2v2) is speech's answer to BERT!
- What?: extract robust speech representations from raw waveform
- I like to think of it as efficient (lossy) data compression
- Best part: Pre-trained models available [5]

Timeline


2017
2018
Transformer
BERT
Jan 2019
CPC [1]
Wav2vec [2]
Sep 2019
Feb 2020
VQ-Wav2vec [3]
Wav2vec 2.0 [4]
Oct 2020

Contrastive Predictive Coding (CPC)

- (lossy) Data Compression: predict future e.g. LPC for speech
- Unsupervised learning: predict contextual/missing data
  e.g. word2vec for LM: a word is known by the company it keeps
- Alternative: transfer learning (low-level representations) but could be still task-dependent e.g. emotion recognition might ignore phonemic content in the first few layers itself
- No such prior in predictive coding



All figures in this presentation are screenshots from respective papers

Components:

- Encoder: compress high-dimensional data into a vector
- AR: bring more context into account for increasing predictive power (aggregator is technically more correct)

CPC training

How is it trained?

- Pick k random samples *Zj*
- Minimise Contrastive loss: $\mathcal{L}_{\mathrm{N}} = -\underset{X}{\mathbb{E}}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)}\right]$
- f(.) is any scoring function e.g. $f_k(x_{t+k}, c_t) = \exp\left(z_{t+k}^T W_k c_t\right)$
- Natural question: Why not simply reduce MSE?
  - Collapse i.e. encoder can output constant embedding i.e. *Zt = 0* (since E2E training)
  - Loss will be 0!
  - Need some *negative samples*
  - Question: Push negative samples away and positive closer (like Triplet loss)?
- Why this exact form (distractor classification loss)?
  -> Backed by theoretical explanation of maximising mutual information

CPC attributes

- Key difference between CPC and AE:
  - Don’t reconstruct entire input
  - Why? Unnecessarily complexity required to model distribution of original signal
    E.g. WaveNet reconstructs entire waveform for generative modelling!
  - Not required: goal is to discard noise and capture high-level information
- Flexible: Encoder and AR model could be anything (GRU, Transformers, etc.)
- Applicable in various domains:
  - Audio (ASR, speaker identification)
  - Language (movie sentiment, question-type classification)
  - Vision (ImageNet classification)
  - RL
- Can use either representation for downstream tasks
  - *Ct*: long-range dependencies
  - *Zt:* otherwise
- Example: replace MFCCs with Ct/Zt for ASR and add linear layer on top to get phone predictions

From CPC to wav2vec

- CPC is a general framework
- Wav2vec = CPC applied specifically for ASR
- Encoder (x -> z): 5-layer convolutional network with
  - Kernels: (10, 8, 4, 4, 4)
  - Strides: (5, 4, 2, 2, 2)
  - Receptive field: 30 ms of data at 16 KHz, 10 ms hop
- Context (z -> c):
  - 9 CNN layers with kernel size = 3 and stride = 1
  - Total receptive field = 210 ms
- Small and large models with different number of layers
- Other hyperparameters: # time steps to predict and # negative samples (sweet spots for both: 12 and 10 resp.)

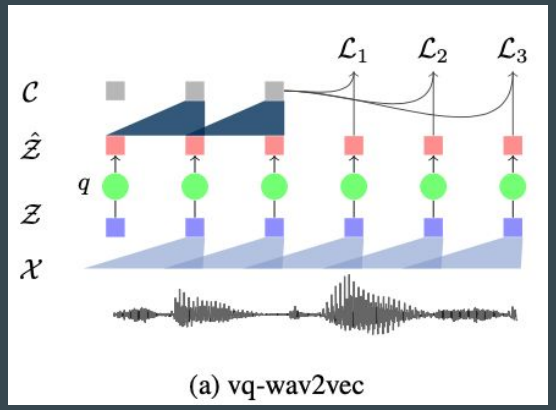
From wav2vec to vq-wav2vec

What?

- Add a quantization step between feature encoder output *Zt* and aggregator network C to get *Qt* (using either Gumbel Softmax or K-Means)
- Rest same as wav2vec

Why discretize?

- Speech made up of distinct human sounds (e.g. phones) (Question: What about para-linguistic content which may not have discrete equivalents?)
- Can apply a host of techniques developed by NLP community (since textual input is discrete)



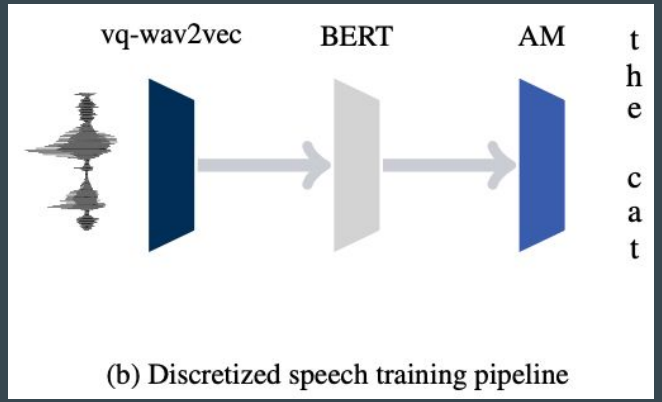
(a) vq-wav2vec

vq-wav2vec

- Hyperparameters of encoder and context network similar to wav2vec
- BERT has 12 layers with 768 dimension
- 2 codebooks, each with 320 entries are used, giving rise to $320^2 = 102{,}400$ possible combinations i.e. discrete units
- Why not directly initialise 102,400 entries?
  -> Mode collapse i.e. only some entries are used. 640 entries means much fewer parameters but at the same time, being more expressive since combinations can be powerful. (check Appendix for results)

vq-wav2vec summary

- 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!
- In summary, discrete + BERT (vq-wav2vec) > continuous + CNN (wav2vec)
- Question: Why not train vq-wav2vec and BERT together? My guess: Tokens input to BERT must be fixed



(b) Discretized speech training pipeline

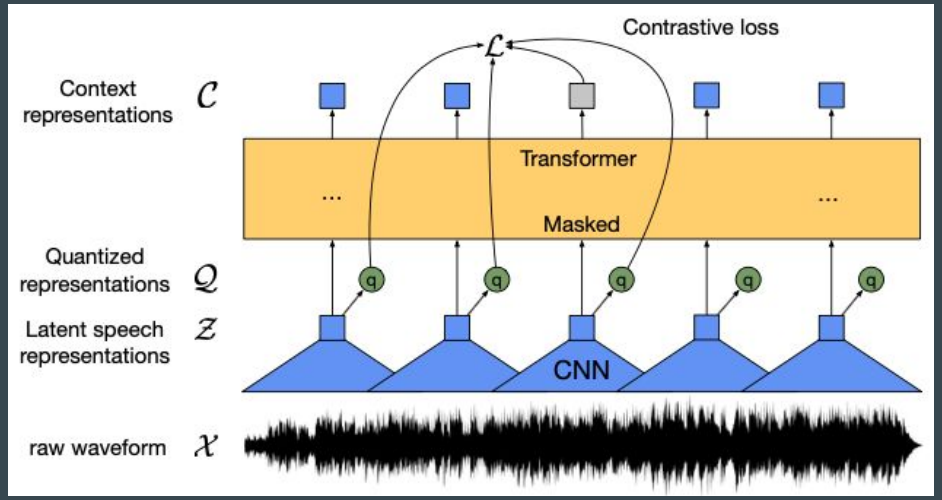
From vq-wav2vec to wav2vec 2.0

2 key changes:

- End-to-end training
- Replace context network with Transformer

Method:

- Raw audio -> CNN -> latent *Zt*
- *Zt* -> Transformer -> Contextual embeddings *Ct*
- Encodings *Zt* quantized to get *Qt* (only for pre-training)



Transformer

- Transformers are everywhere! (NLP, even Vision)
- Context CNN (z -> c) in wav2vec replaced by a Transformer
- Self-attention over entire sequence >> simple CNNs i.e. leads to better contextualised embeddings
- Similar to BERT, mask time steps randomly and predict quantized representations (called targets) of masked time steps
- Not sure if new task (predicting randomly masked time steps) is harder or easier than predicting only the future

Quantization

- Done ONLY during pre-training
- *Zt* is discretized by choosing one entry from each of the G codebooks using product quantization and concatenating them
- Contrastive loss is similar to wav2vec (here, sim refers to cosine):

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

- Additional diversity loss to encourage all codewords are used
- Thus, we are pushing context representations towards the discrete domain while still allowing more information since continuous in nature!
  Question: Why not use a simple affine transform on *Ct* before computing similarity with quantized representation (like wav2vec)?

Experiments

- Feature encoder: 7-layer CNN with 512 channels and strides similar to wav2vec
- 25 ms receptive field, 20 ms hop
- Low resource: On Librispeech, after 53k hours of pre-training, WER drops from 16.3/25.2 (discrete BERT or vq-wav2vec) to 4.8/8.2 (test clean/other) on training with 10 minutes of labelled dataset!! Think about the possibilities of developing ASR for low-resource Indian languages!
- Also attains best performance when using all 960 hours of labelled dataset

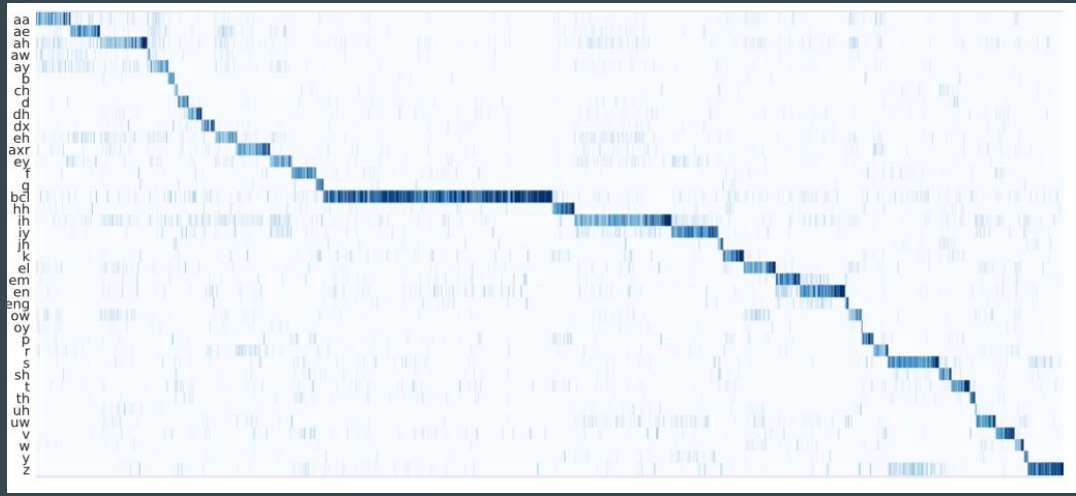
Ablation

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

- Quantizing *Zt* before feeding it to transformer may be suboptimal because it hasn't seen enough context to decide which phone it is. Only on passing it through a context network (transformer in this case), will *Ct* have a meaningful discrete equivalent
- "Continuous targets reduce the effectiveness of self-supervised training since targets can capture detailed artifacts of the current sequence, e.g. speaker and background information, which make the task easier and prevent the model from learning general representations beneficial to speech recognition"
- In other words, both continuous leads to some form of overfitting
- Question: For tasks such as emotion recognition, we actually need this para-linguistic information. Will last configuration perform better?

Latents and Phones

- Using TIMIT alignment information, activation of each discrete latent is calculated
- We then check co-occurrence with phone labels
- (i, j) corresponds to average activation of i^{th} phone with j^{th} latent (i: 1 to 39, j: 1 to 640)
- Nice intuitive plot which says that each latent (column) roughly corresponds to a phone



What about music?

Couldn’t find direct application of wav2vec on music datasets

Spijkervet, Janne, and John Ashley Burgoyne. "Contrastive Learning of Musical Representations." arXiv preprint arXiv:2103.09410 (2021)

- Different from wav2vec but same philosophy (self-supervised pre-training + FT on given dataset)
- Results presented for music classification
- Code available: https://spijkervet.github.io/CLMR/

Takeaways

- For low-resource: Pre-train on large unlabeled datasets in self-supervised fashion + Fine-tune on task at hand
- Loss function extremely crucial (I tend to stick to the simplest option i.e. work on better model architectures instead of novel loss functions)
- Enforcing bottlenecks (such as quantization) can improve performance (provided we have an intuitive backing for where to impose the bottleneck)

How do I get started?

Hugging Face (highly recommended since the community is very active):

- Model: Wav2Vec2
- Code walkthrough: Fine-Tune Wav2Vec2 for English ASR in Hugging Face with 🤗 Transformers

PyTorch: Speech Recognition with Wav2Vec2 — PyTorch Tutorials 1.10.1+cu102 documentation

Helpful blog article: Self-training and pre-training, understanding the wav2vec series

References

[1] Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." arXiv preprint arXiv:1807.03748 (2018)

[2] Schneider, Steffen, et al. "wav2vec: Unsupervised pre-training for speech recognition." arXiv preprint arXiv:1904.05862 (2019)

[3] Baevski, Alexei, Steffen Schneider, and Michael Auli. "vq-wav2vec: Self-supervised learning of discrete speech representations." arXiv preprint arXiv:1910.05453 (2019)

[4] Baevski, Alexei, et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations." arXiv preprint arXiv:2006.11477 (2020)

[5] https://huggingface.co/docs/transformers/model_doc/wav2vec2