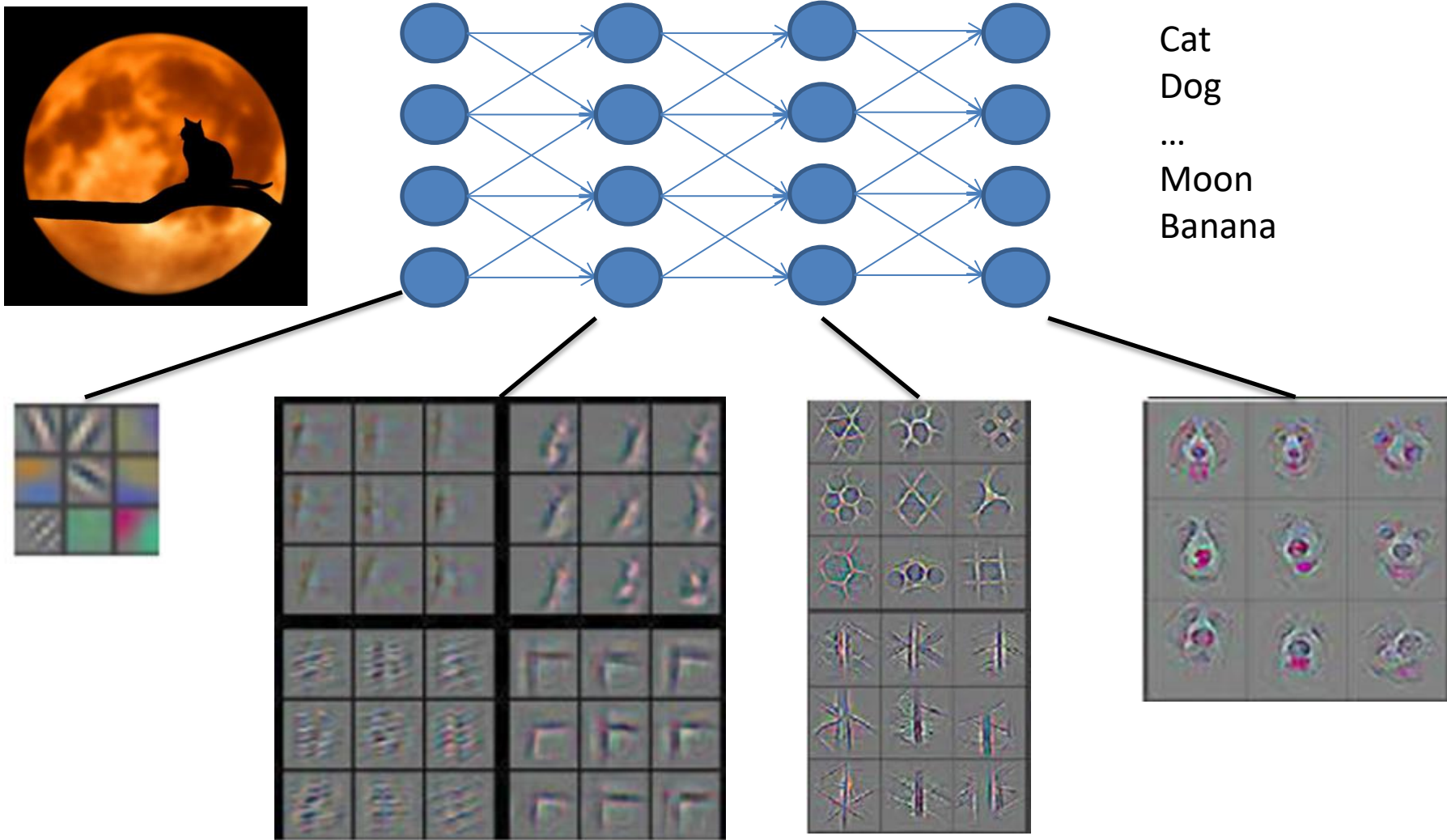


Unsupervised speech representation learning using WaveNet autoencoders

<https://arxiv.org/abs/1901.08810>

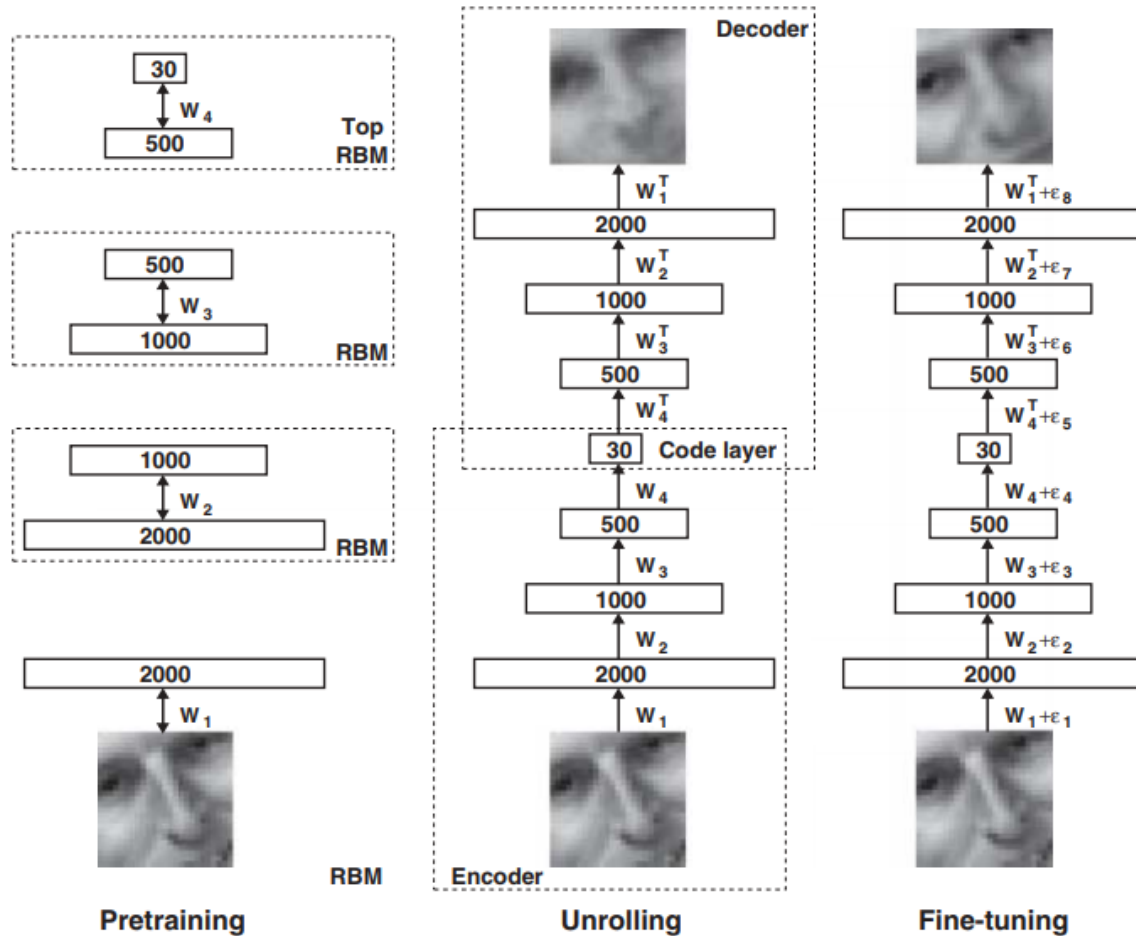
Jan Chorowski
University of Wrocław
06.06.2019

Deep Model = Hierarchy of Concepts



Deep Learning history: 2006

2006: Stacked RBMs

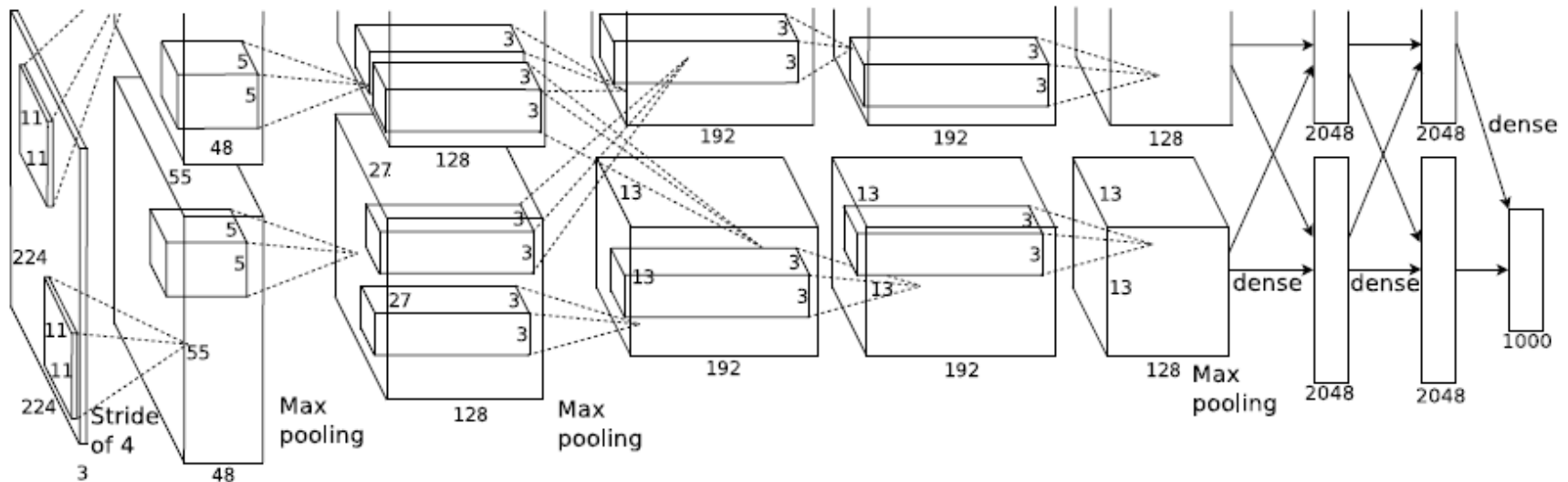
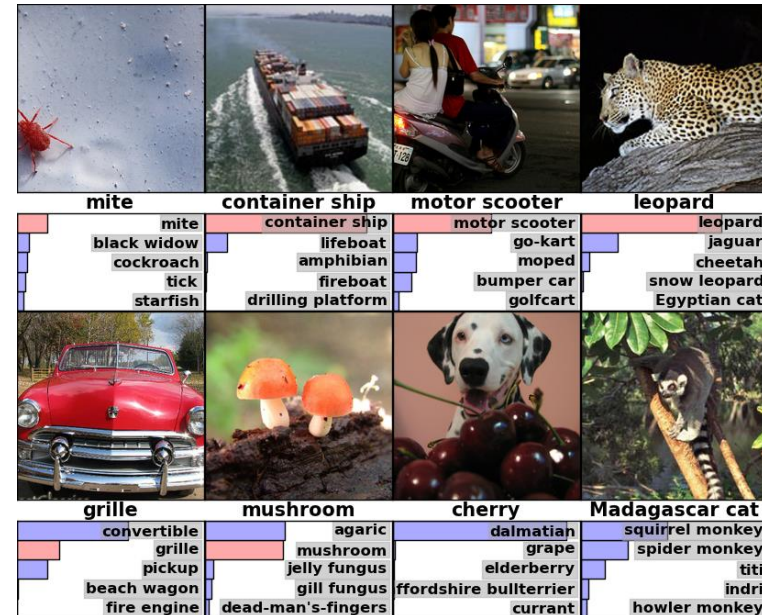


Deep Learning history: 2012

2012: Alexnet

SOTA on Imagenet

Fully supervised training



Deep Learning Recipe

1. Get a massive, labeled dataset $D = \{(x, y)\}$:
 - Comp. vision: Imagenet, 1M images
 - Machine translation: EuroParlamanet data, CommonCrawl, several million sent. pairs
 - Speech recognition: 1000h (LibriSpeech), 12000h (Google Voice Search)
 - Question answering: SQuAD, 150k questions with human answers
 - ...
2. Train model to maximize $\log p(y|x)$

Value of Labeled Data

- Labeled data is crucial for deep learning
- But labels carry little information:
 - Example:
An ImageNet model has 30M weights, but
ImageNet is about 1M images from 1000 classes
Labels: $1M * 10\text{bit} = 10\text{Mbits}$

Raw data: (128 x 128 images): ca 500 Gbits!

Value of **Unlabeled** Data

“The brain has about 10^{14} synapses and we only live for about 10^9 seconds. So we have a lot more parameters than data. This motivates the idea that we must do a lot of unsupervised learning since the perceptual input (including proprioception) is the only place we can get 10^5 dimensions of constraint per second.”

Geoff Hinton

Unsupervised learning recipe

1. Get a massive ~~labeled~~ dataset $D = \{x\}$
Easy, unlabeled data is nearly free
2. Train model to...???

What is the task?

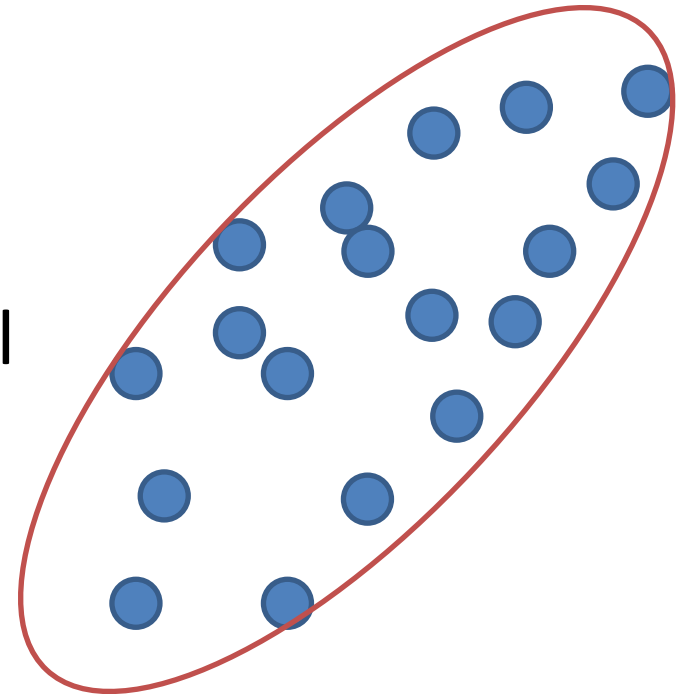
What is the loss function?

Unsupervised learning by modeling data distribution

Train the model to
minimize $-\log p(x)$

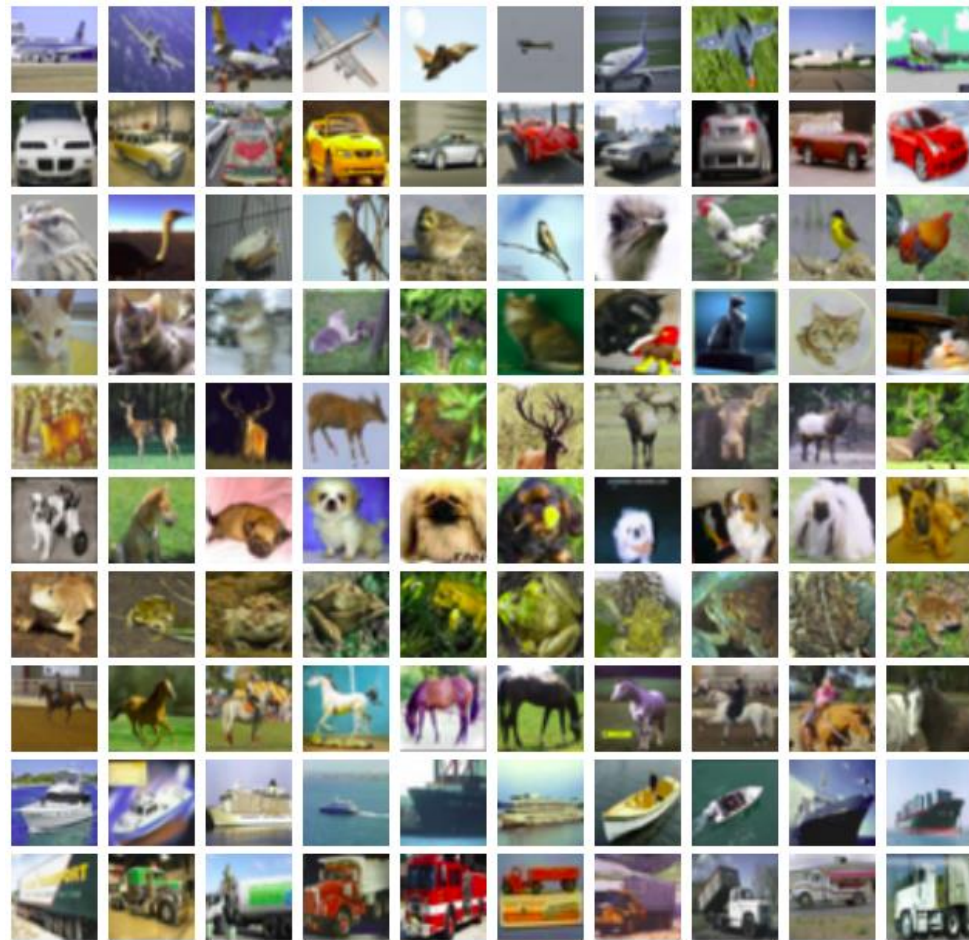
E.g. in 2D:

- Let $D = \{x: x \in \mathbb{R}^2\}$
- Each point is an 2-dimensional vector
- We can draw a point-cloud
- And fit some known distribution, e.g. a Gaussian



Learning high dimensional distributions is hard

- Assume we work with small (32x32) images
- Each data point is a real vector of size $32 \times 32 \times 3$
- Data occupies only a tiny fraction of $\mathbb{R}^{32 \times 32 \times 3}$
- Difficult to learn!



Autoregressive Models

Decompose probability of data points in R^n into n conditional univariate probabilities:

$$\begin{aligned} p(x) &= p(x_1, x_2, \dots, x_n) \\ &= p(x_1)p(x_2|x_1) \dots p(x_n|x_1, x_2, \dots, x_{n-1}) \\ &= \prod_i p(x_i|x_{<i}) \end{aligned}$$

Autoregressive Example: Language modeling

Let x be a sequence of word ids.

$$p(x) = p(x_1, x_2, \dots, x_n) = \prod_i p(x_i | x_{<i})$$
$$\approx \prod_i p(x_i | x_{i-k}, x_{i-k+1}, \dots, x_{i-1})$$

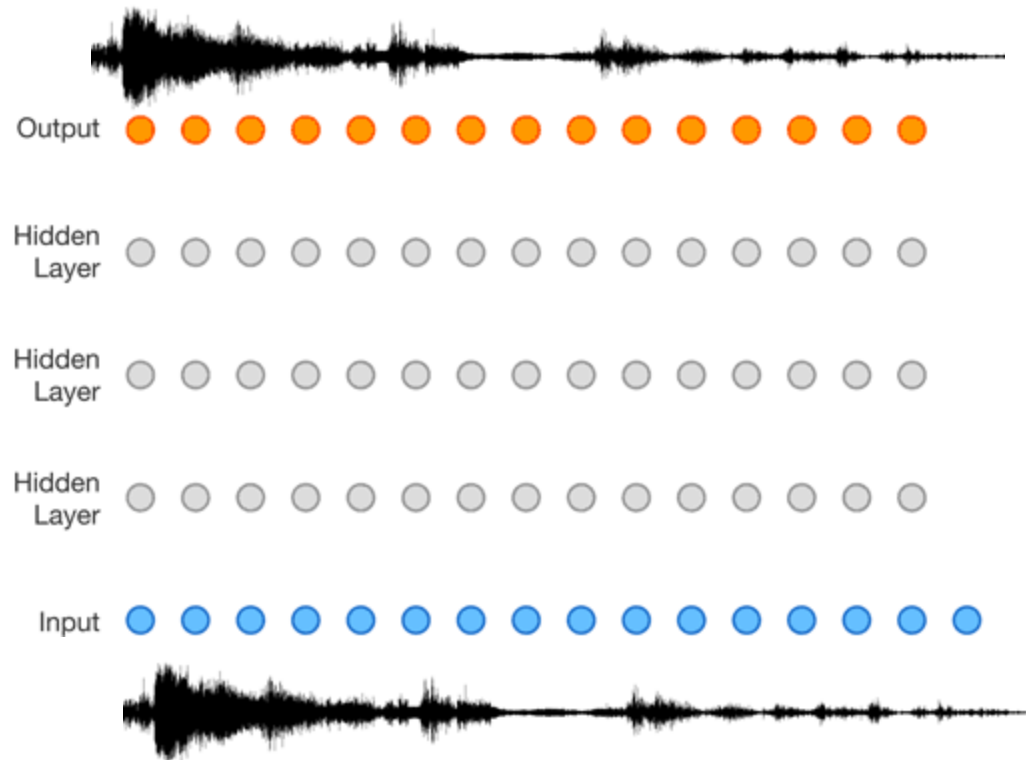
$p(\text{It's a nice day}) = p(\text{It}) * p(\text{'s} | \text{it}) * p(\text{a} | \text{'s}) \dots$

- Classical n-gram models: cond. probs. estimated using counting
- Neural models: cond. probs. estimated using neural nets

WaveNet: Autoregressive modeling of speech

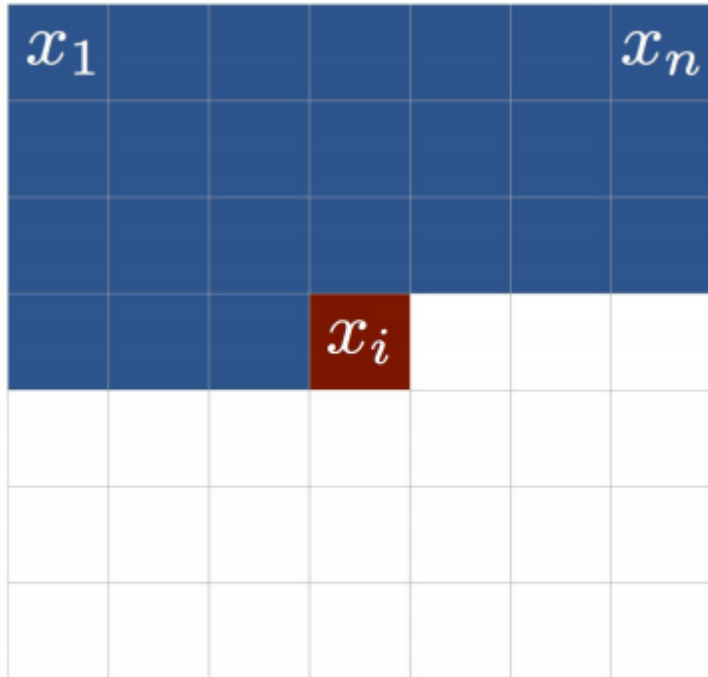
Treat speech as a sequence of samples!

Predict each sample base on previous ones.



PixelRNN:

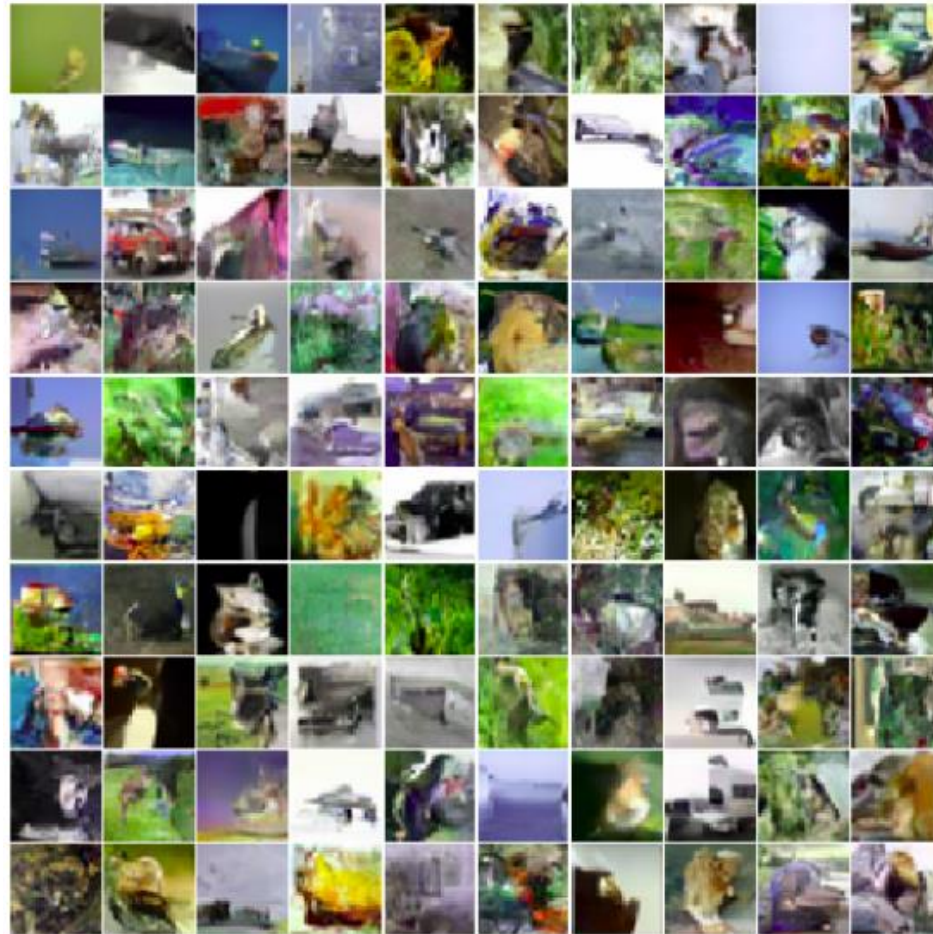
A “language model for images”



Pixels generated left-to-right,
top-to-bottom.

Cond. probabilities
estimated using recurrent or
convolutional neural
networks.

PixelCNN samples



Salimans et al, "A PixelCNN Implementation with Discretized Logistic Mixture Likelihood and Other Modifications"

Autoregressive Models Summary

The good:

- Simple to define (pick an ordering).
- Often yield SOTA log-likelihood.

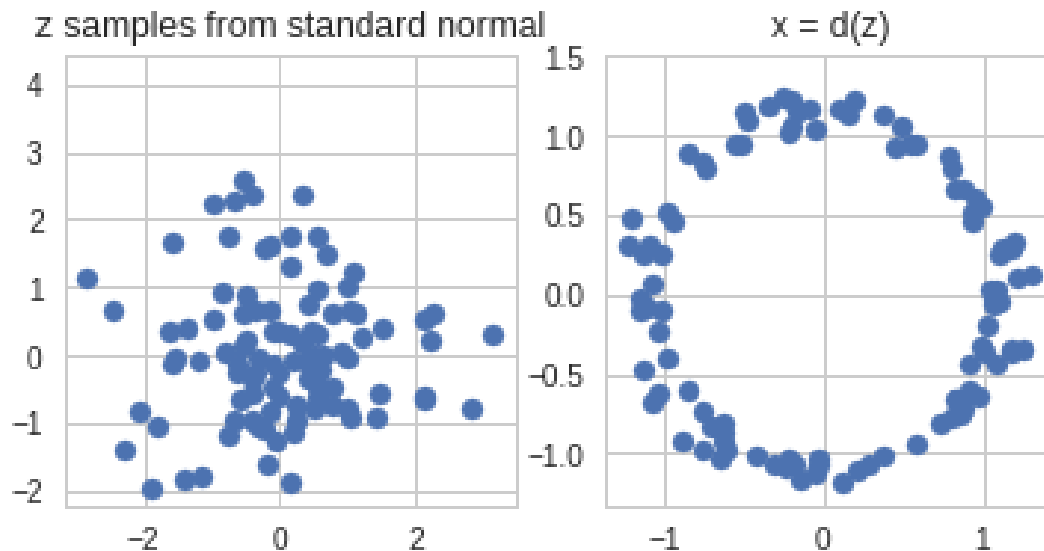
The bad:

- Training and generation require $O(n)$ ops.
- No compact intermediate data representation – not obvious how to use for downstream tasks.

Latent Variable Models

Intuition: to generate something complicated, do:

1. Sample something simple $z \sim \mathcal{N}(0,1)$
2. Transform it $x = \frac{z}{10} + \frac{z}{\|z\|}$



Variational autoencoder: A neural latent variable model

Assume a 2 stage data generation process:

$$z \sim \mathcal{N}(0,1)$$

prior $p(z)$ assumed to be simple

$$x \sim p(x|z)$$

complicated transformation

implemented with a neural network

How to train this model?

$$\log p(x) = \log \sum_z p(x|z)p(z)$$

This is often intractable!

ELBO: A lower bound on $\log p(x)$

Let $q(z|x)$ be any distribution. We can show that

$$\begin{aligned}\log p(x) &= \\ &= KL(q(z|x) \parallel p(z|x)) + \mathbb{E}_{z \sim q(z|x)} \left[\log \left(\frac{p(z|x)}{q(z|x)} p(x) \right) \right] \\ &\geq \mathbb{E}_{z \sim q(z|x)} \left[\log \left(\frac{p(z|x)}{q(z|x)} p(x) \right) \right] \\ &= \mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] - KL(q(z|x) \parallel p(z))\end{aligned}$$

The bound is tight for $p(z|x) = q(z|x)$.

ELBO interpretation

ELBO, or evidence lower bound:

$$\log p(x) \geq \mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] - KL(q(z|x) \parallel p(z))$$

where:

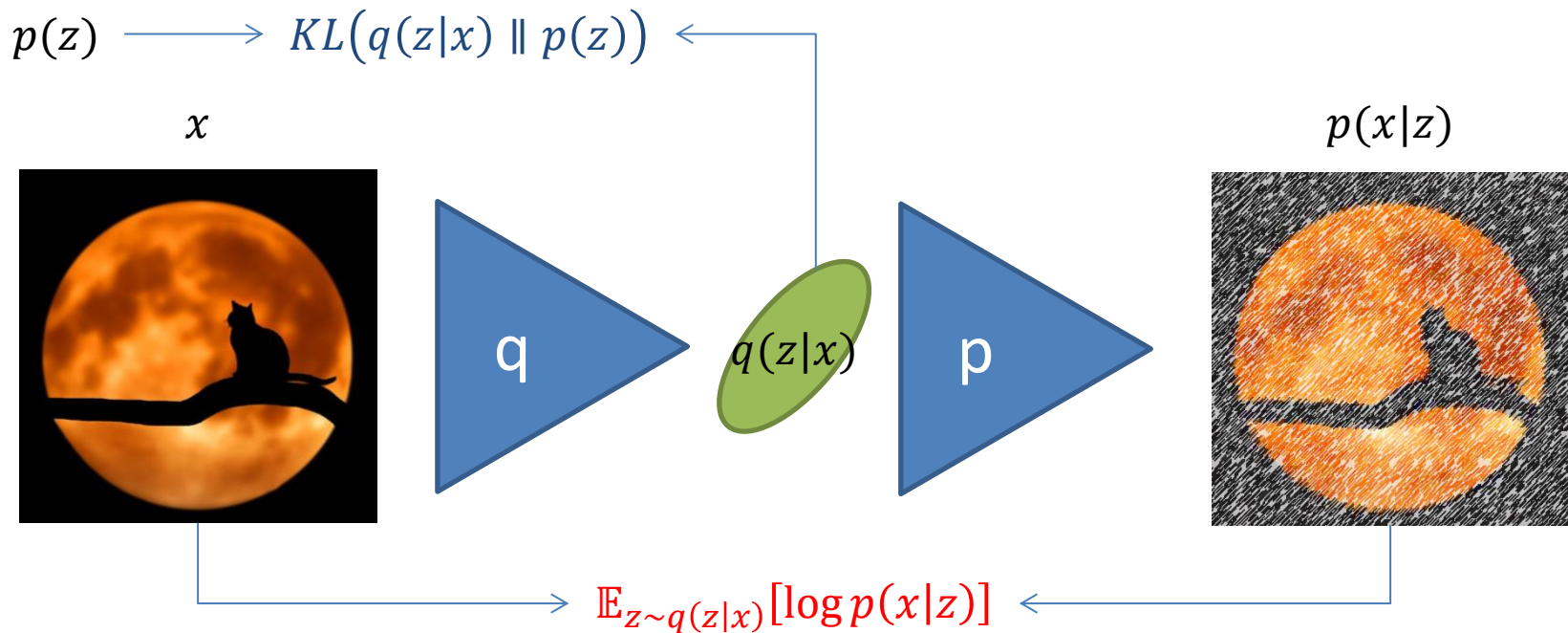
$\mathbb{E}_{z \sim q(z|x)} [\log p(x|z)]$ reconstruction quality:
how many nats we need to reconstruct x ,
when someone gives us $q(z|x)$

$KL(q(z|x) \parallel p(z))$ code transmission cost:

how many nats we transmit about x in $q(z|x)$ rather than $p(z)$

Interpretation: **do well at reconstructing x** , limiting the amount of information about x encoded in z .

The Variational Autoencoder



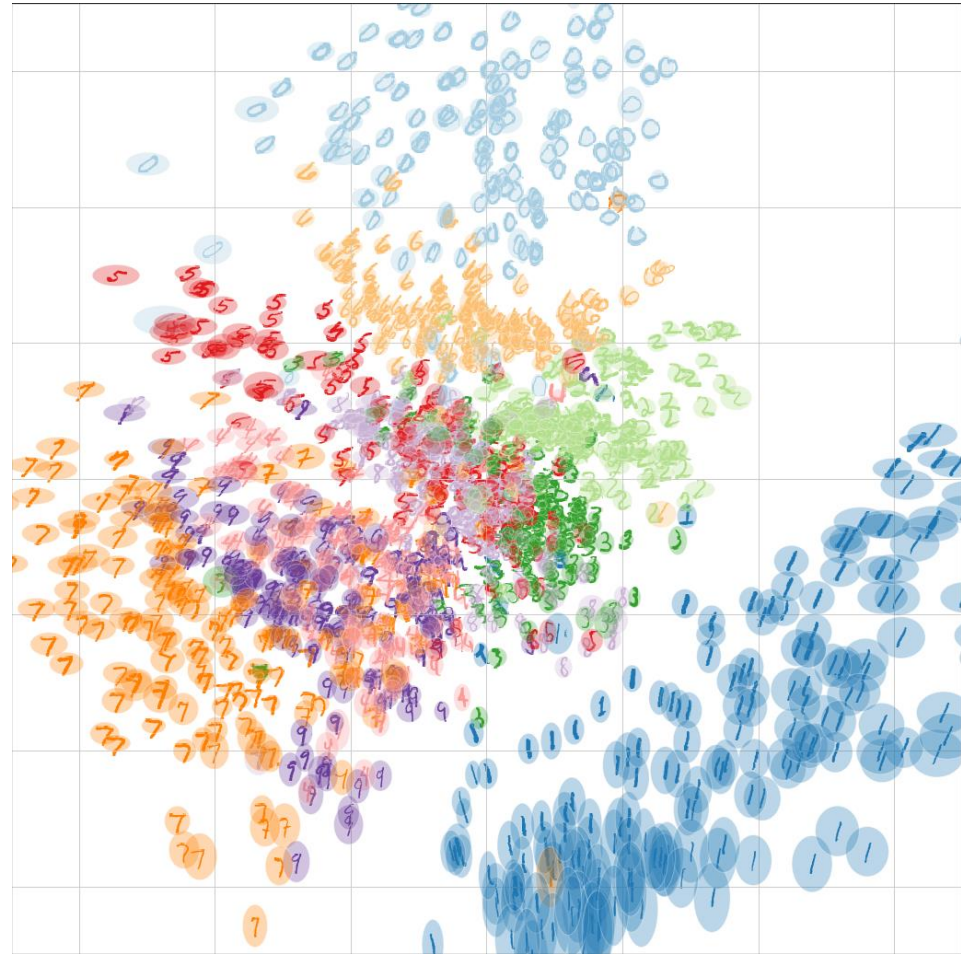
An input x is put through the q network to obtain a distribution over latent code z , $q(z|x)$.

Samples z_1, \dots, z_k are drawn from $q(z|x)$. They k reconstructions $p(x|z_k)$ are computed using the network p .

VAE is an Information Bottleneck

Each sample is represented as a Gaussian

This discards information (latent representation has low precision)



VQVAE – deterministic quantization

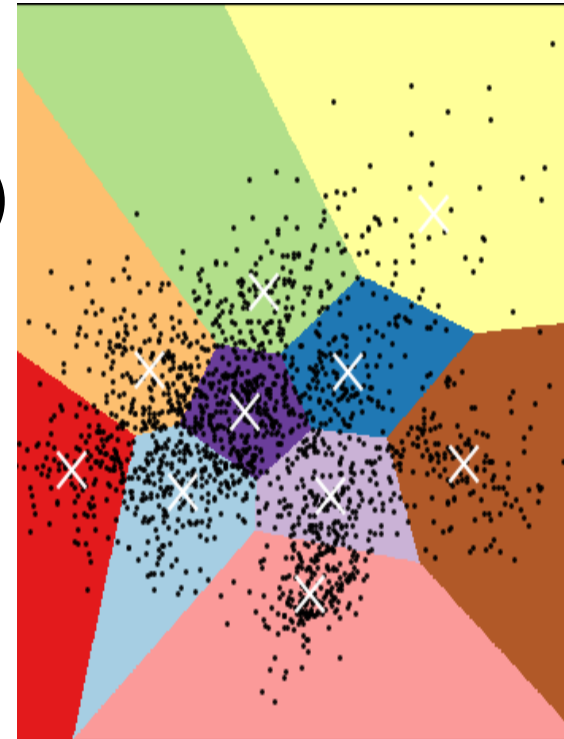
Limit precision of the encoding by quantizing (round each vector to a nearest prototype).

Output can be treated:

- As a sequence of discrete prototype ids (tokens)
- As a distributed representation (the prototypes themselves)

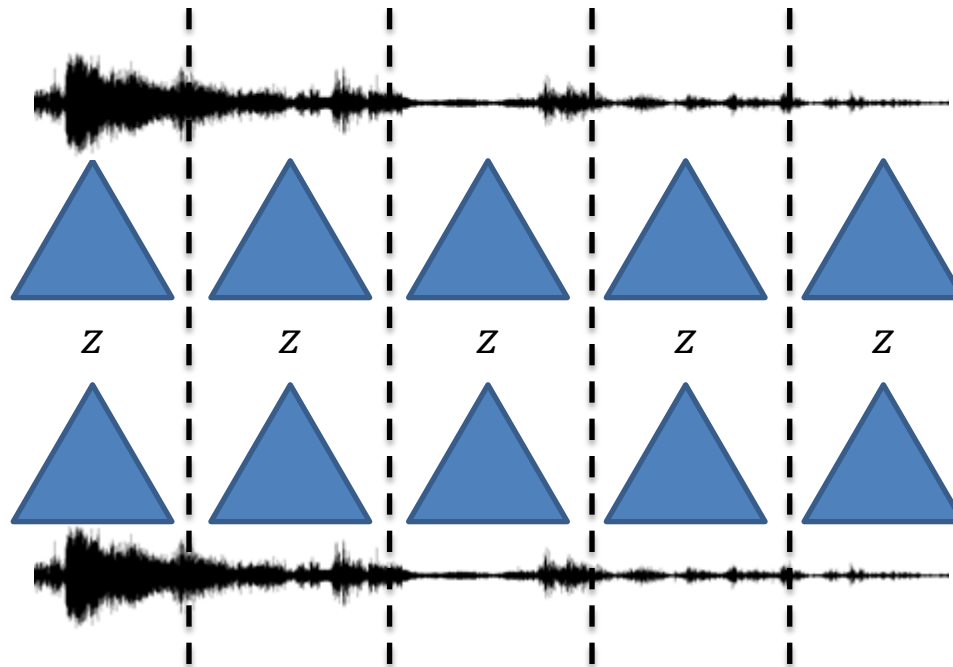
Train using the straight-through estimator,
with auxiliary losses:

$$\mathcal{L} = \log p(x | z_q(x)) + \| \text{sg}(z_e(x)) - e_{q(x)} \|_2^2 + \gamma \| z_e(x) - \text{sg}(e_{q(x)}) \|_2^2$$



VAEs and sequential data

To encode a long sequence, we apply the VAE to chunks:

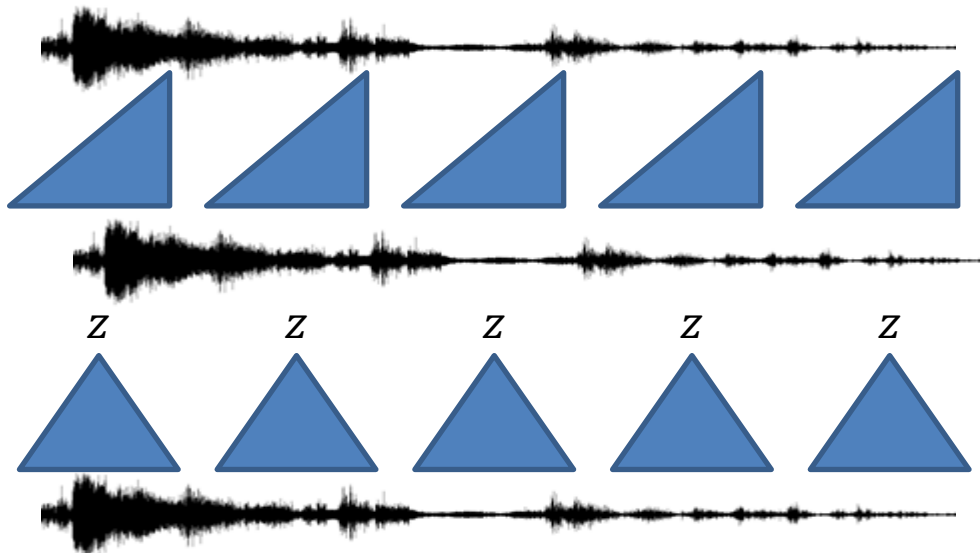


But neighboring chunks are similar!

We are encoding the same information in many z s!

We are wasting capacity!

WaveNet + VAE



A WaveNet reconstructs the waveform using the information from the past

Latent representations are extracted at regular intervals.

The WaveNet uses information from:

1. The past recording
2. The latent vectors z
3. Other conditioning, e.g. about speaker

The encoder transmits in z s only the information that is missing from the past recording .

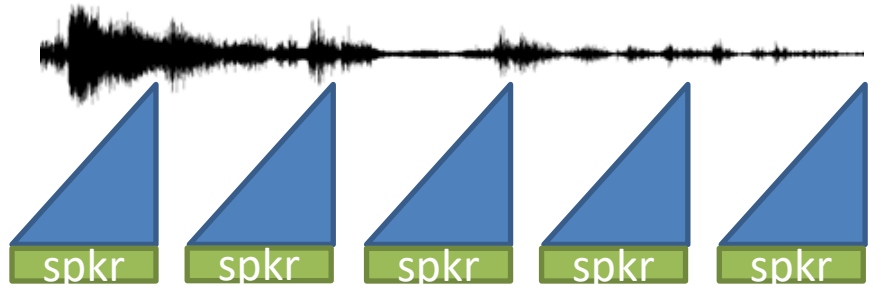
The whole system is a very low bitrate codec

(roughly 0.7kbits/sec, the waveform is 16k Hz* 8bit=128kbit/sec)

VAE + autoregressive models: latent collapse danger

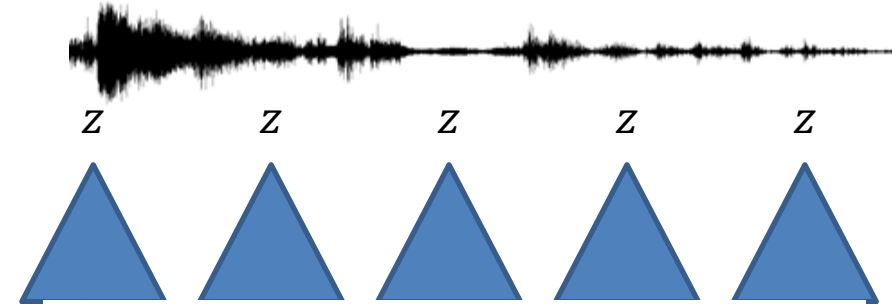
- Purely Autoregressive models: SOTA log-likelihoods
- Conditioning on latents:
information passed through bottleneck
lower reconstruction x-entropy
- In standard VAE model actively tries to
 - reduce information in the latents
 - maximally use autoregressive information=> Collapse: latents are not used!
- Solution: stop optimizing KL term
(free bits), make it a hyperparam (VQVAE)

Model description



WaveNet decoder conditioned on:

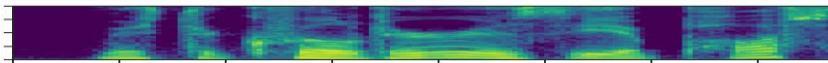
- latents extracted at 24Hz-50Hz
- speaker



3 bottleneck evaluated:

- Dimensionality reduction, max 32 bits/dim
- VAE, $KL(q(z|x) \parallel \mathcal{N}(0,1))$ nats (bits)
- VQVAE with K protos: $\log_2 K$ bits

Or



Input:

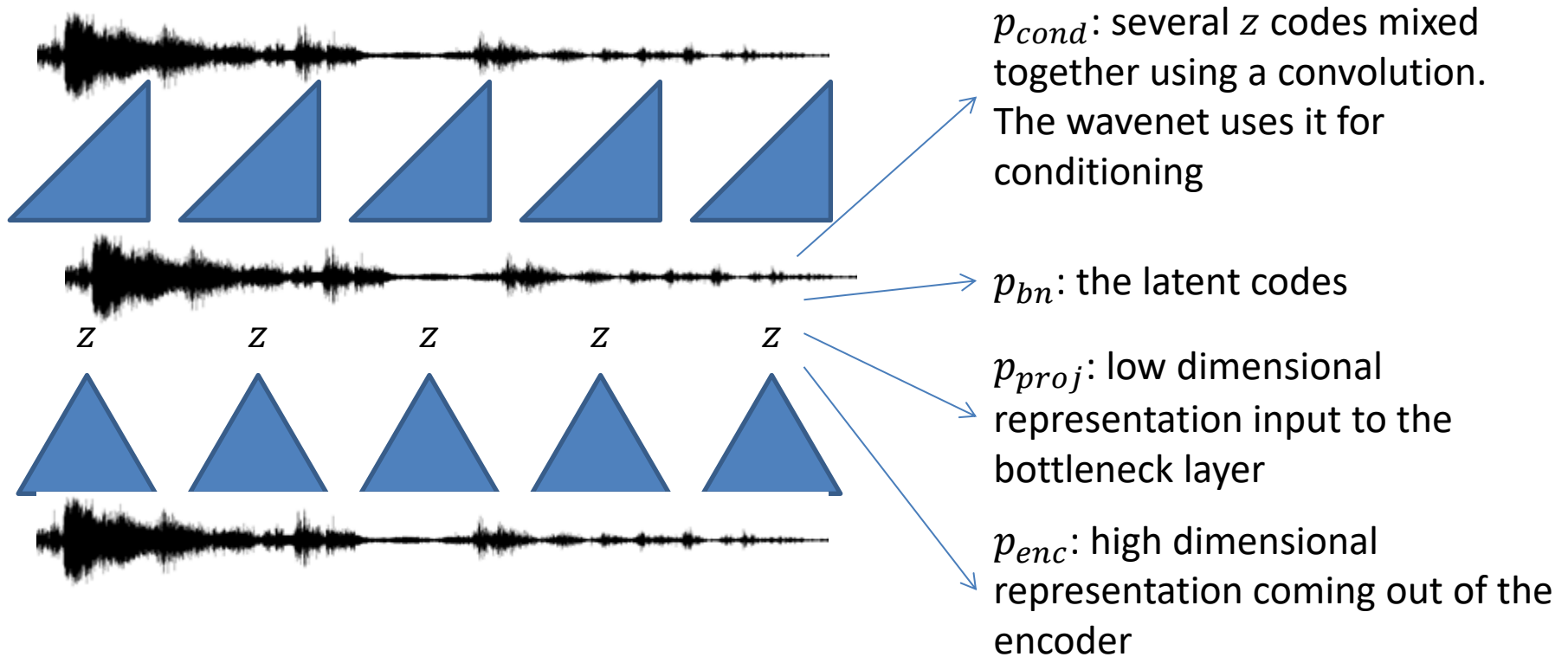
Waveforms, Mel Filterbanks, MFCCs

Hope: speaker separated form content.

Proof: <https://arxiv.org/abs/1805.09458>

Representation probing points

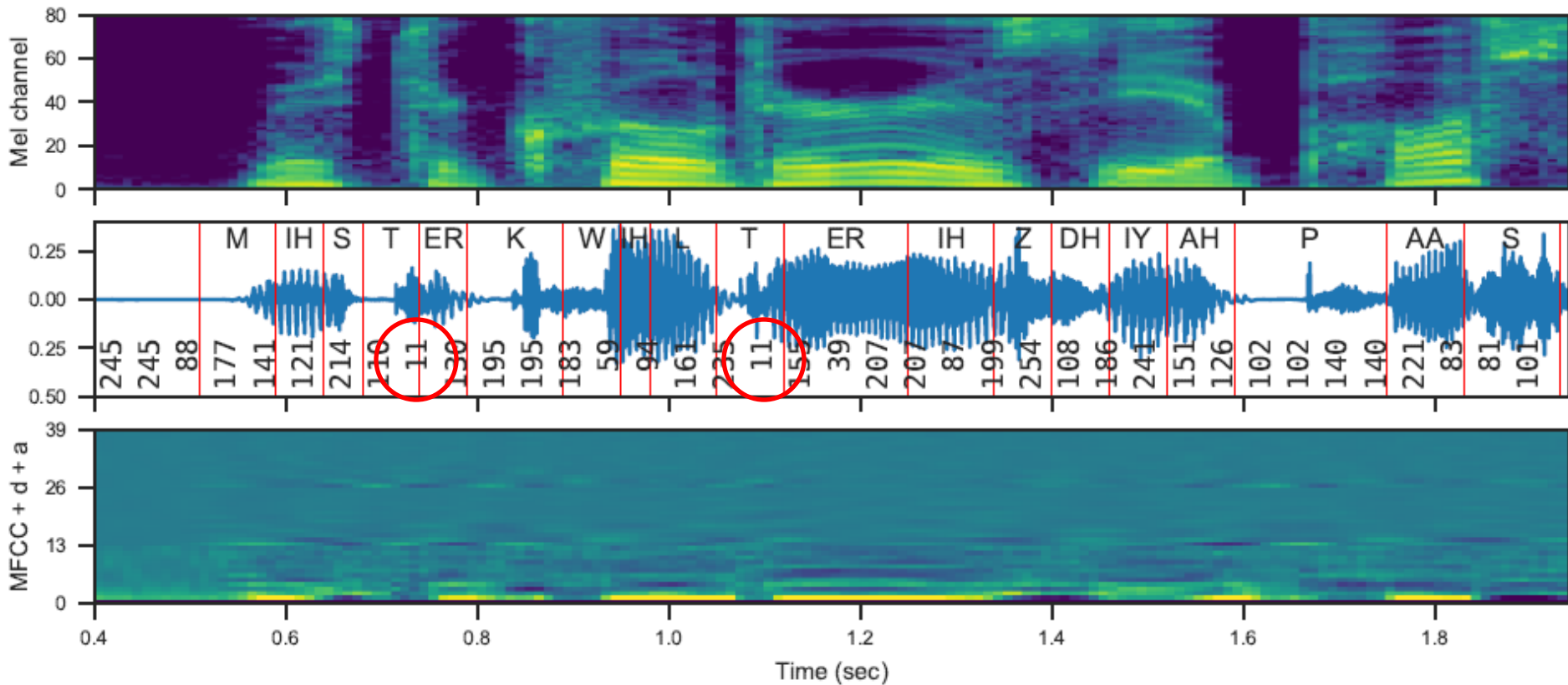
We have inserted probing classifiers at 4 points in the network:



Experimental Questions

- What information is captured in the latent codes/probing points?
- What is the role of the bottleneck layer?
- Can we regularize the latent representation?
- How to promote a segmentation?
- How good is the representation on downstream tasks?
- What design choices affect it?

VQVAE Latent representation

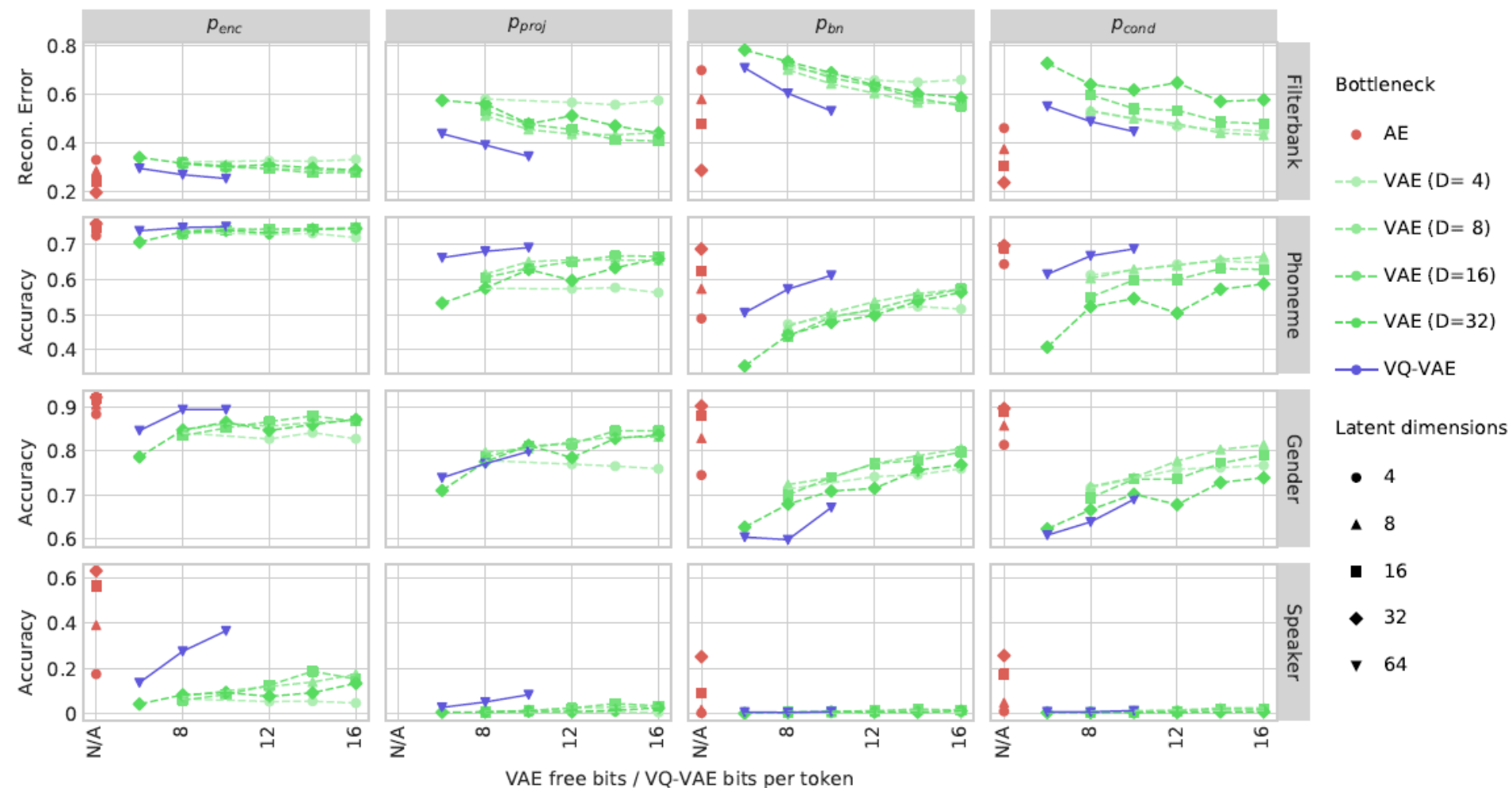


What information is captured in the latent codes?

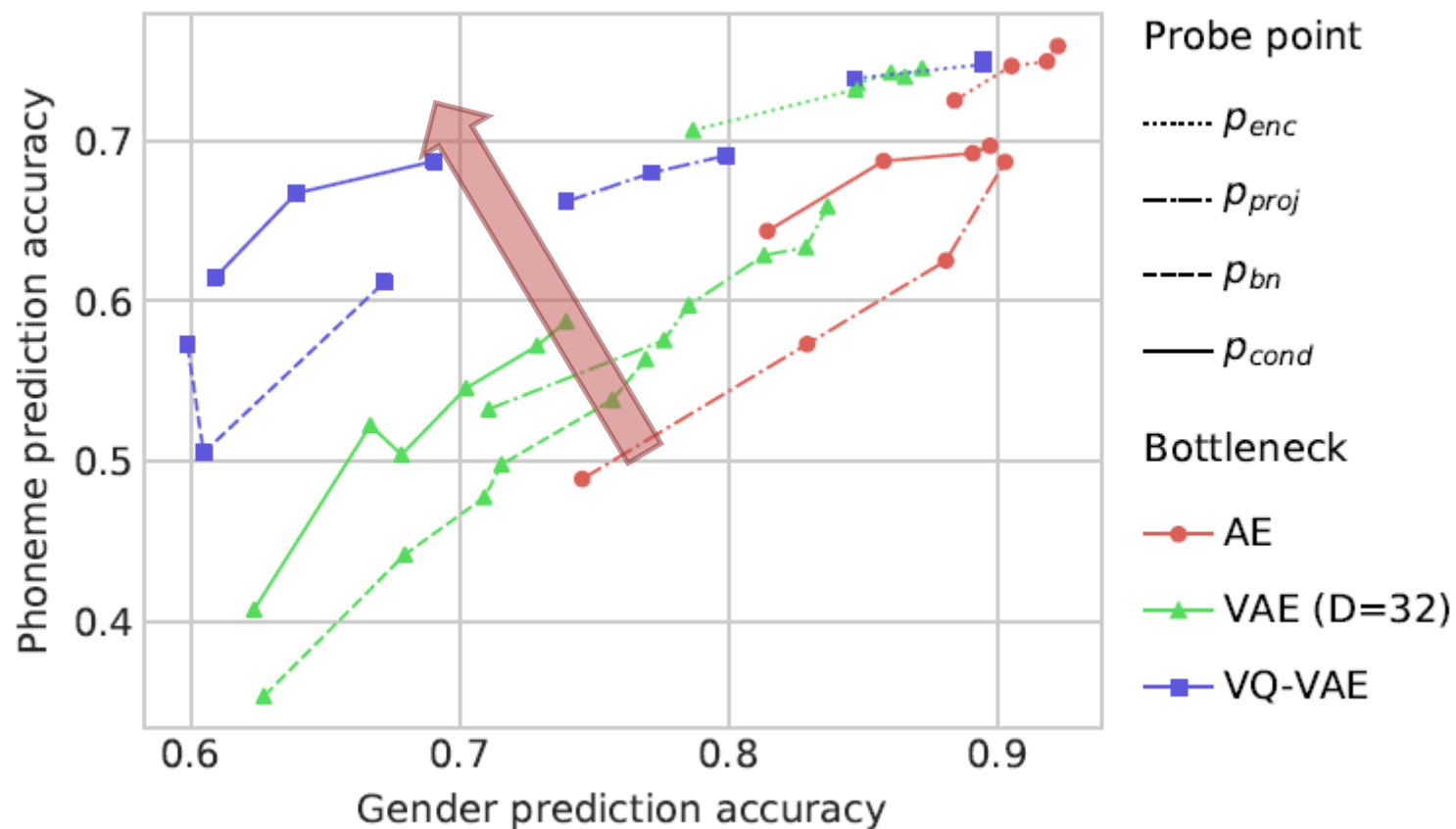
For each probing point, we have trained predictors for:

- Framewise phoneme prediction
- Speaker prediction
- Gender prediction
- Mel Filterbank reconstruction

Results



Phonemes vs Gender tradeoff



How to regularize the latent codes?

We want the codes to capture phonetic information.

Phones vary in duration – from about 30ms to 1s (long silences).

Thus we need to extract the latent codes frequently enough to capture the short phones, but when the phone doesn't change, the latents should be stable too.

This is similar to *slow features* analysis.

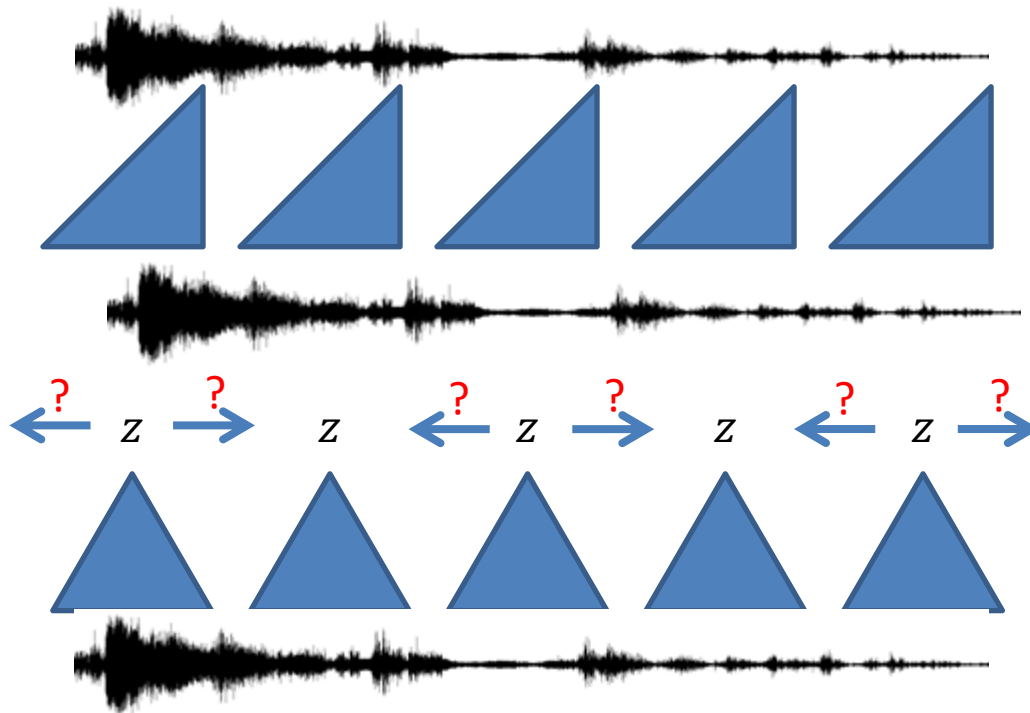
Problem with enforcing slowness

Enforcing slow features (small changes to the latents), has a trivial optimum: constant latents.

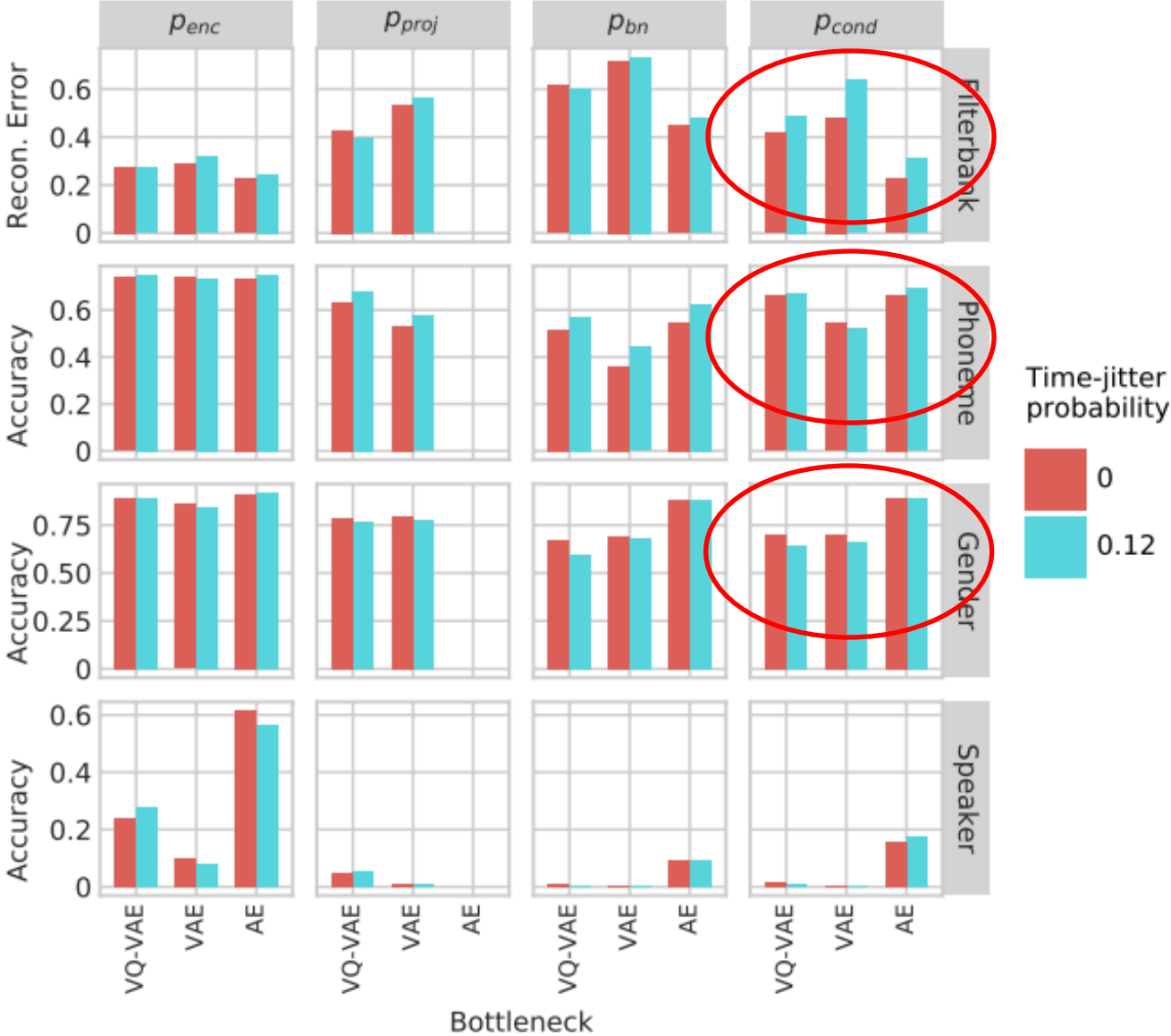
Then WaveNet can just disregard the encoder, and latent space collapses.

Randomized time jitter

Rather than putting a penalty on changes of the latent z vectors, add time jitter to them. This forces the model to have a more stable representation over time.



Randomized time jitter results



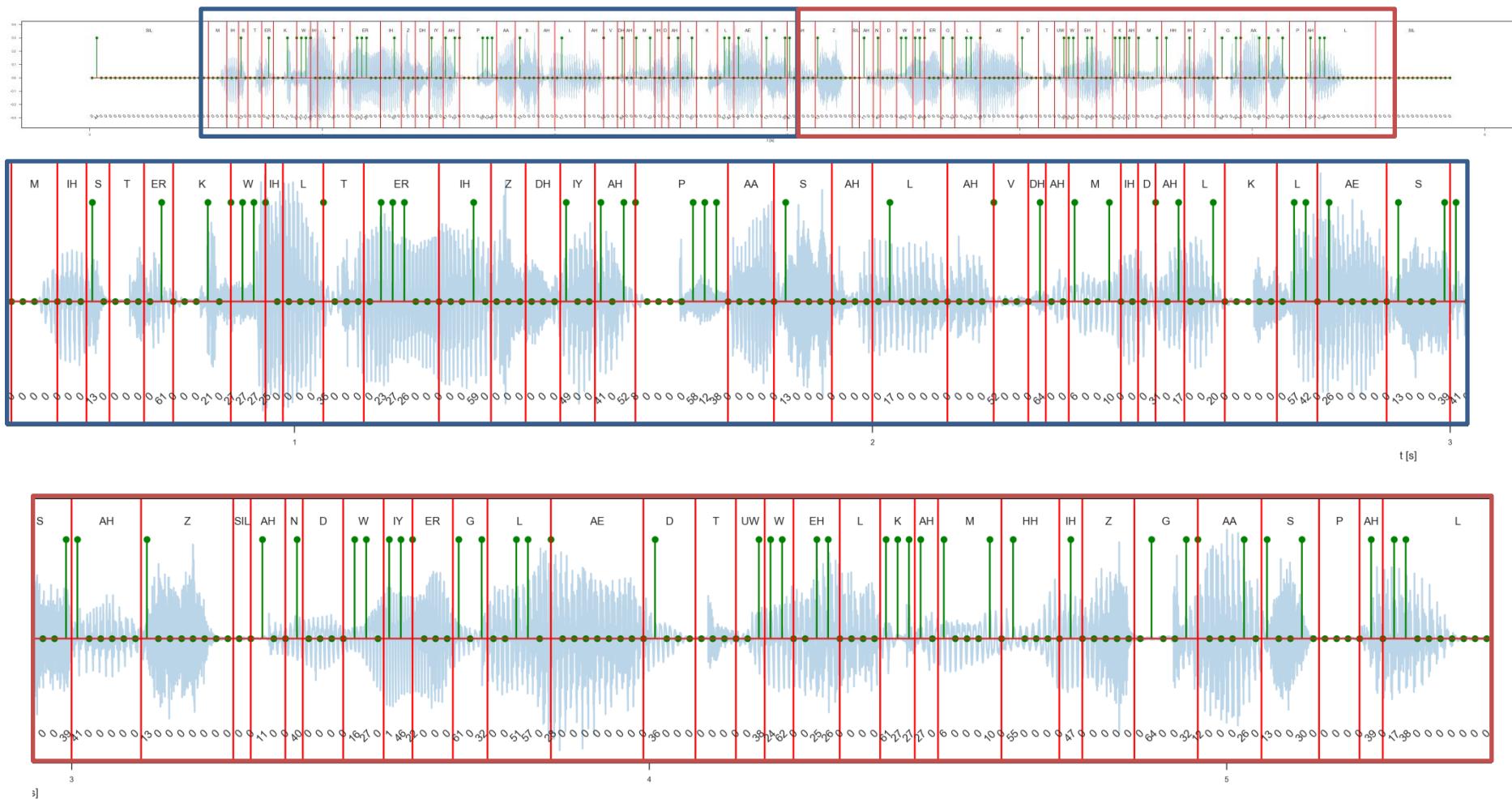
How to learn a segmentation?

The representation should be constant within a phoneme, then change abruptly

Enforcing slowness leads to collapse, jitter prevents the model from using pairs of tokens as codepoints

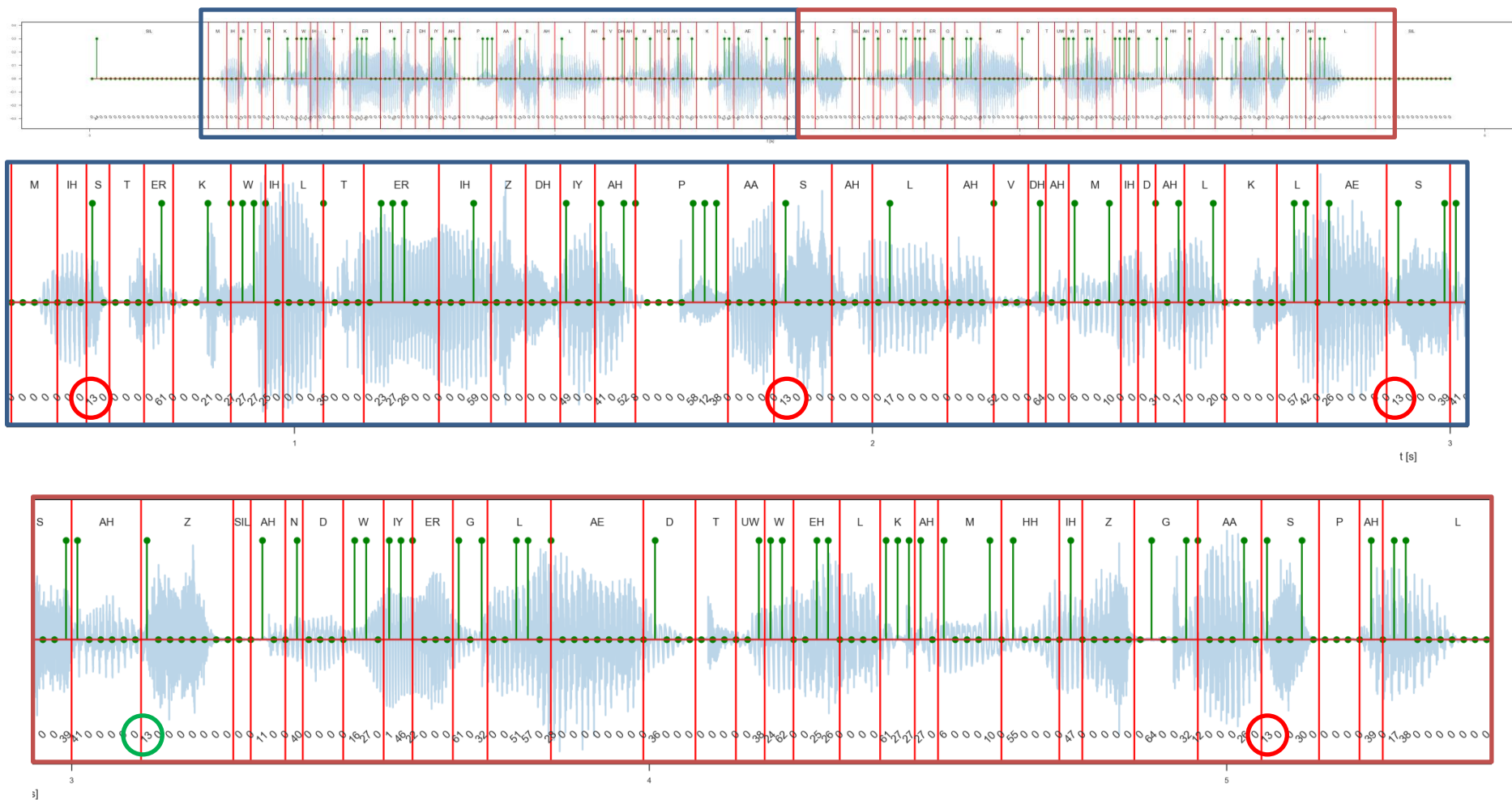
Idea: allow the model to infrequently emit a non-trivial representation

Non-max suppression – choosing where to emit



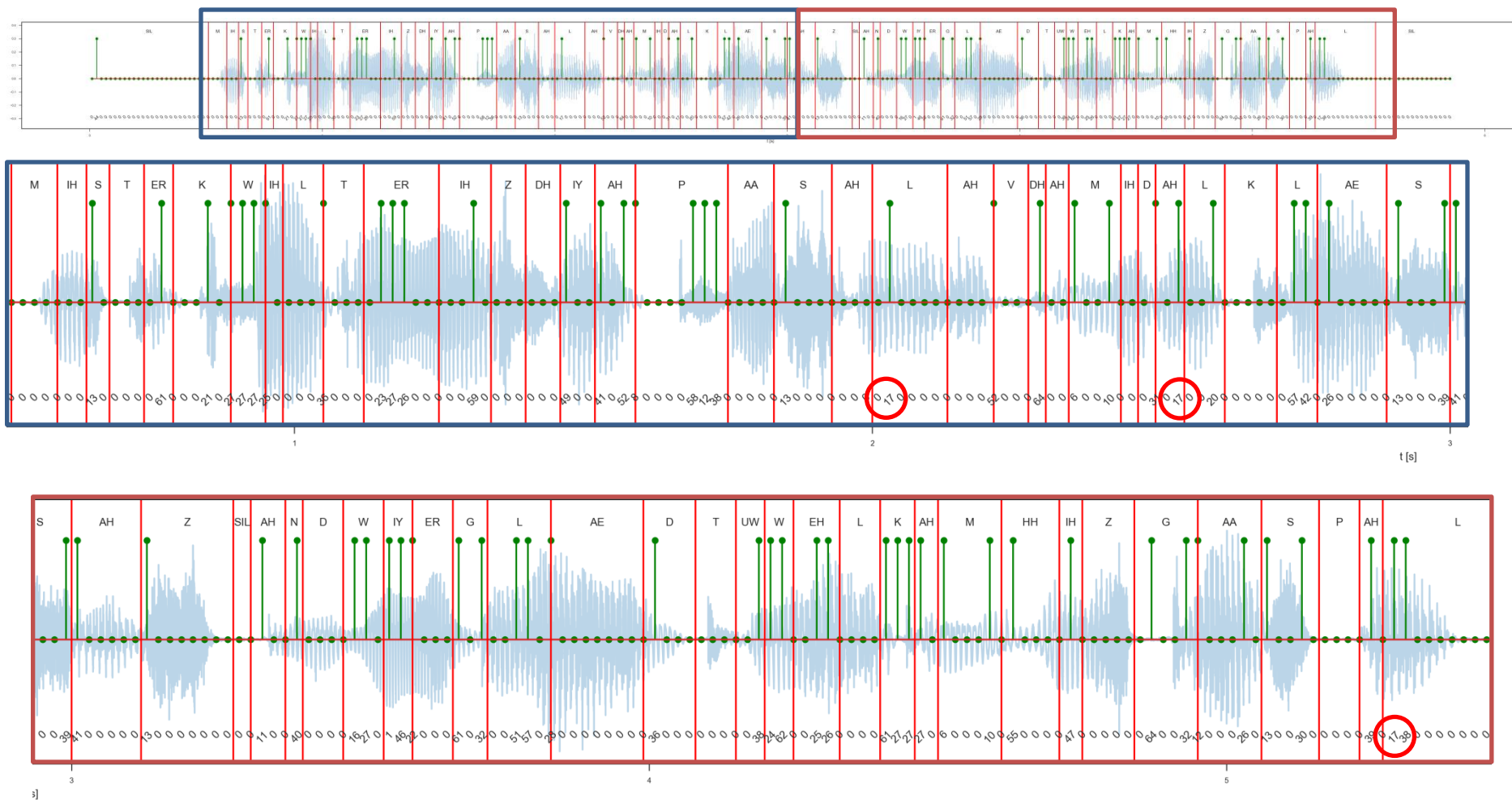
Latents computed at 25Hz, but allow only $\frac{1}{4}$ nonzero

Non-max suppression – choosing where to emit



Token 13 is near emissions of „S” and some „Z”

Non-max suppression – choosing where to emit



Token 17 is near emissions of some „L”

Performance on ZeroSpeech unit discovery

Model	Within-speaker									Across-speaker								
	English (45h)			French (24h)			Mandarin (2.4h)			English (45h)			French (24h)			Mandarin (2.4h)		
	1s	10s	2m	1s	10s	2m	1s	10s	2m	1s	10s	2m	1s	10s	2m	1s	10s	2m
Unsupervised baseline	12.0	12.1	12.1	12.5	12.6	12.6	11.5	11.5	11.5	23.4	23.4	23.4	25.2	25.5	25.2	21.3	21.3	21.3
Supervised topline	6.5	5.3	5.1	8.0	6.8	6.8	9.5	4.2	4.0	8.6	6.9	6.7	10.6	9.1	8.9	12.0	5.7	5.1
VQ-VAE (per lang, p_{cond})	5.6	5.5	5.5	7.3	7.5	7.5	11.2	10.7	10.8	8.1	8.0	8.0	11.0	10.8	11.1	12.2	11.7	11.9
Heck et al. [57]	6.9	6.2	6.0	9.7	8.7	8.4	8.8	7.9	7.8	10.1	8.7	8.5	13.6	11.7	11.3	8.8	7.4	7.3
Chen et al. [58]	8.5	7.3	7.2	11.2	9.4	9.4	10.5	8.7	8.5	12.7	11.0	10.8	17.0	14.5	14.1	11.9	10.3	10.1
Ansari et al. [59]	7.7	6.8	N/A	10.4	N/A	8.8	10.4	9.3	9.1	13.2	12.0	N/A	17.2	N/A	15.4	13.0	12.2	12.3
Yuan et al. [60]	9.0	7.1	7.0	11.9	9.5	9.5	11.1	8.5	8.2	14.0	11.9	11.7	18.6	15.5	14.9	12.7	10.8	10.7

SOTA results in unsupervised phoneme discrimination Fr and EN ZeroSpeech challenge.

Mandarin shows limitation of the method:

- Too little training data (only 2.4h unsup. speech)
- Tonal information is discarded.

English: VQVAE bottleneck adds speaker invariance

English	Within spkr.	Across spkr.
VQ-VAE (per lang, MFCC, p_{cond})	5.6	8.0
VQ-VAE (per lang, MFCC, p_{bn})	6.2	8.8
VQ-VAE (per lang, MFCC, p_{proj})	5.9	9.0
VQ-VAE (all lang, MFCC, p_{cond})	5.8	8.6
VQ-VAE (all lang, MFCC, p_{bn})	6.3	9.2
VQ-VAE (all lang, MFCC, p_{proj})	5.8	9.3
VQ-VAE (all lang, fbank, p_{proj})	6.0	10.1

The quantization discards speaker info, improving across-speaker results
MFCCs slightly better than FBanks

Mandarin: VQVAE bottleneck discards phone information

Mandarin	Within spkr.	Across spkr.
VQ-VAE (per lang, MFCC, p_{cond})	11.2	12.2
VQ-VAE (per lang, MFCC, p_{bn})	10.8	11.9
VQ-VAE (per lang, MFCC, p_{proj})	9.9	11.0
VQ-VAE (all lang, MFCC, p_{cond})	9.2	10.3
VQ-VAE (all lang, MFCC, p_{bn})	9.0	9.9
VQ-VAE (all lang, MFCC, p_{proj})	7.4	8.6
VQ-VAE (all lang, fbank, p_{proj})	6.8	7.8

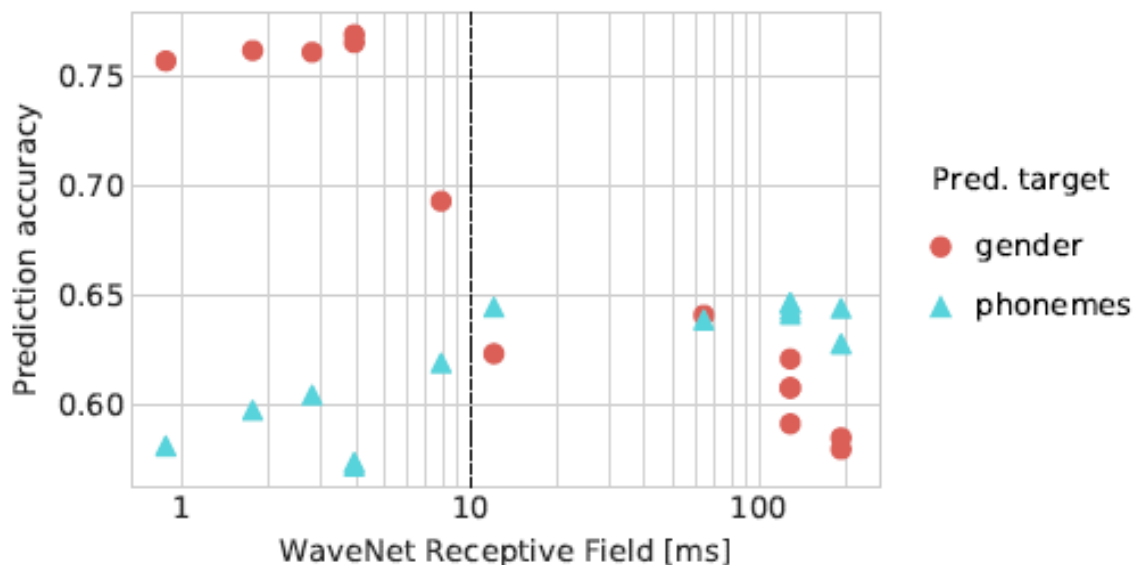
The quantization discards too much (tone insensitivity?)

MFCCs worse than FBanks

What impacts the representation?

Implicit time constant of the model:

- Input field of view of the encoder – optimum close to 0.3s
- WaveNet field of view - needs at minimum 10ms

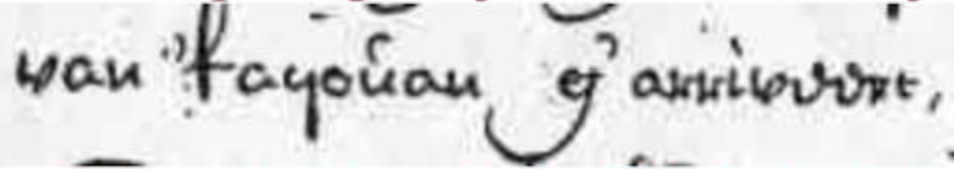


Failed attempts

- I found no benefits from building a hierarchical representation (extract latents at different timescales), even when the slower latents had no bottleneck
- Filterbank reconstruction works worse than waveform
 - Too easy for the autoregressive model?
 - Too little detail?

The future

ende voorspoedige reijse den 16=en Julij da

Handwritten Dutch text in a cursive script, showing the words 'van 'fayōian g'arriveert,'.

van taijoan g'arriveert, sijn E[delen] aldaar

We will explore similar ideas during JSALT2019 topic “Distant supervision for representation learning”.

The workshop will:

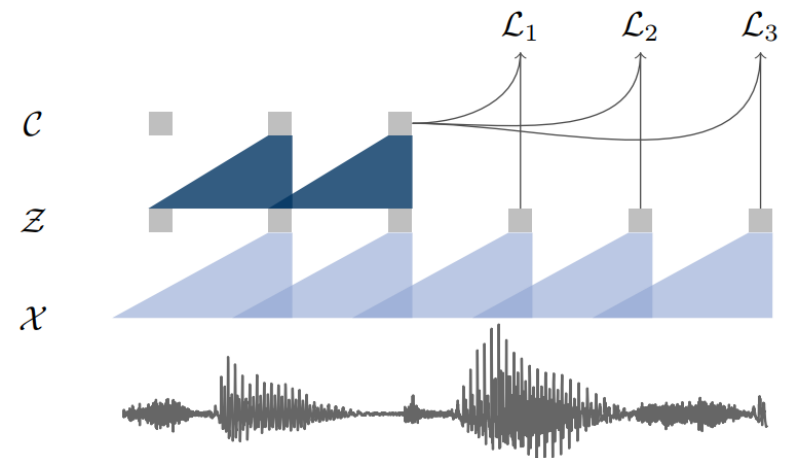
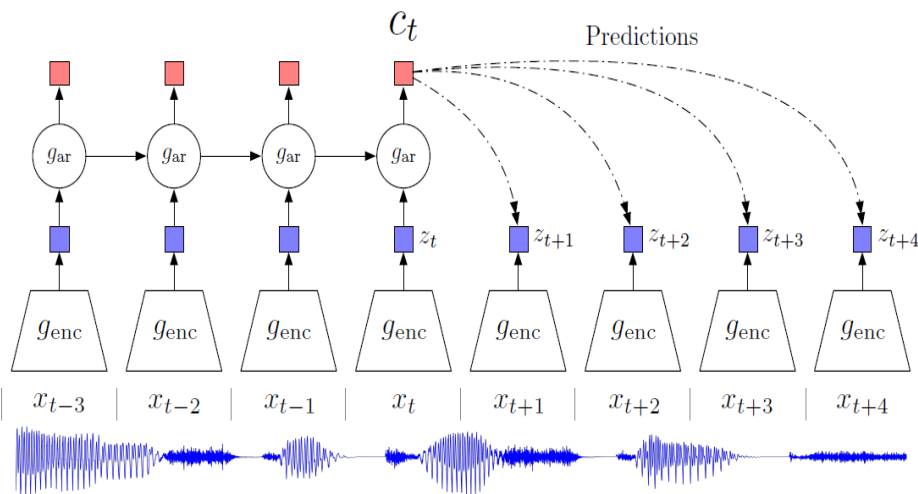
- Work on speech and handwriting
- Explore ways of integrating metadata and unlabeled data to control latent representations
- Focus on downstream supervised OCR and ASR tasks under low data conditions

Some approaches to try:

- Contrastive predictive coding
- Masked reconstruction

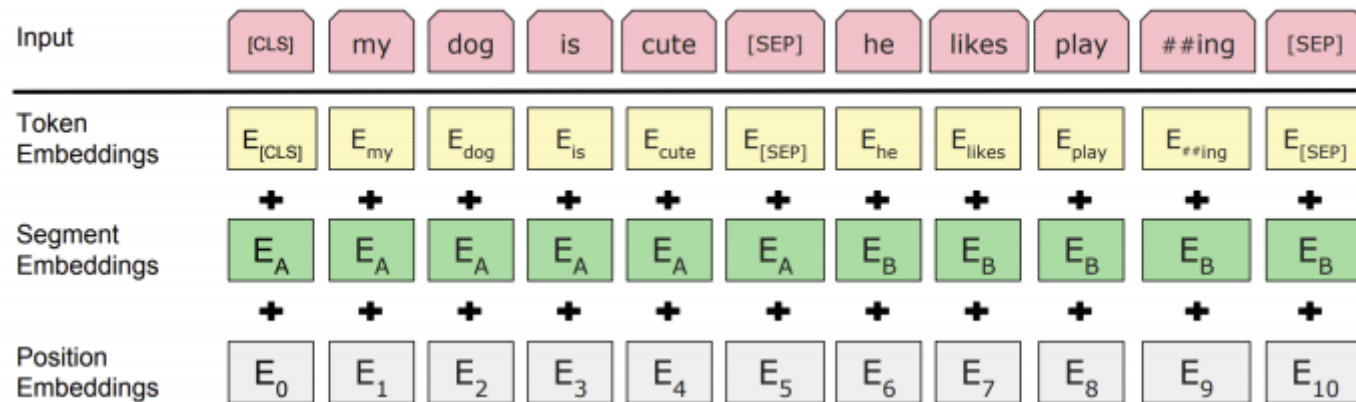
The future: CPC

- Contrastive coding learns representations that can tell a frame from other ones



The future: masked reconstruction

- BERT is a recent, SOTA model for sentence representation learning



- Mask the inputs:

Input: The man went to the [MASK]₁ . He bought a [MASK]₂ of milk .

Labels: [MASK]₁ = store; [MASK]₂ = gallon

Thank you!

- Questions?

Backup

ELBO Derivation pt. 1

$$\begin{aligned} KL(q_\phi(z|x)||p_\theta(z|x)) &= \mathbb{E}_{z \sim q_\phi(z|x)} \left[-\log \frac{p_\theta(z|x)}{q_\phi(z|x)} \right] = \\ &= \mathbb{E}_{z \sim q_\phi(z|x)} \left[-\log \frac{p_\theta(z|x)p_\theta(x)}{q_\phi(z|x)p_\theta(x)} \right] = \\ &= \mathbb{E}_{z \sim q_\phi(z|x)} \left[-\log \frac{p_\theta(z|x)p_\theta(x)}{q_\phi(z|x)} \right] + \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x)] = \\ &= \mathbb{E}_{z \sim q_\phi(z|x)} \left[-\log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] + \log p_\theta(x) \end{aligned}$$

$$\log p_\theta(x) = KL(q_\phi(z|x)||p_\theta(z|x)) + \mathbb{E}_{z \sim q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\phi(z|x)} \right]$$

ELBO derivation pt. 2

$$\begin{aligned}\log p_{\theta}(\mathbf{x}) &\geq \mathbb{E}_{z \sim q_{\phi}(z|\mathbf{x})} \left[\log \frac{p_{\theta}(\mathbf{x}, z)}{q_{\phi}(z|\mathbf{x})} \right] = \\ &= \mathbb{E}_{z \sim q_{\phi}(z|\mathbf{x})} \left[\log \frac{p_{\theta}(\mathbf{x}|z)p_{\theta}(z)}{q_{\phi}(z|\mathbf{x})} \right] = \\ &= \mathbb{E}_{z \sim q_{\phi}(z|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|z)] - \mathbb{E}_{z \sim q_{\phi}(z|\mathbf{x})} \left[-\log \frac{p_{\theta}(z)}{q_{\phi}(z|\mathbf{x})} \right] \\ &= \mathbb{E}_{z \sim q_{\phi}(z|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|z)] - KL(q_{\phi}(z|\mathbf{x}) || p_{\theta}(z))\end{aligned}$$