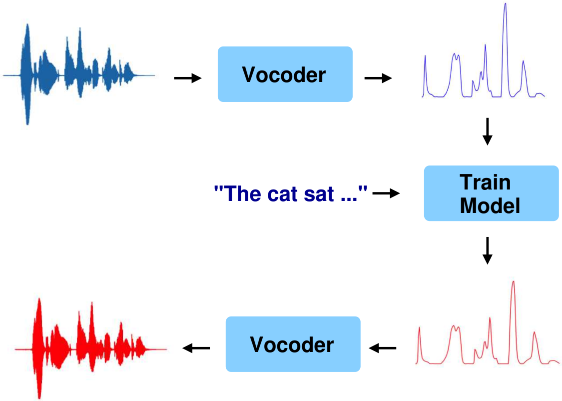
UNIVERSITY OF CAMBRIDGE

# MLMI14: Waveform Level Synthesis

Mark Gales

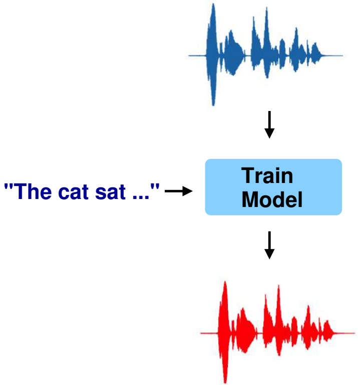
Lent 2021

# Statistical Parametric Synthesis [13]




- Quality of vocoder fundamentally limits performance

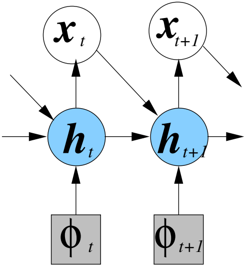
# Waveform Level Synthesis [19]




- Challenges:
  - 16KHz/24KHz predictions
  - long-span dependencies
  - 16KHz/24KHz labels

# Long-Span Dependencies

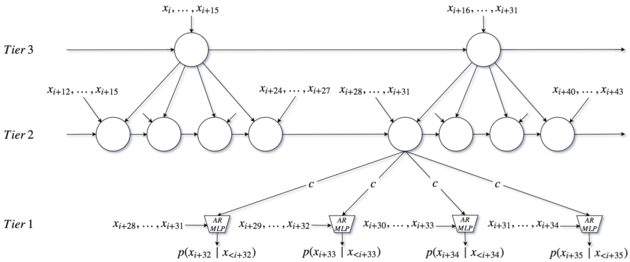
- Speech events act over about 250ms (roughly)
    - 5ms frame rate: this is 50 frames
    - 16Hz samples: this is 4000 samples




- Consider recurrent model:
    - history vector representation
    - rule-of-thumb < 100 samples
      6ms history!
    - insufficient history memory

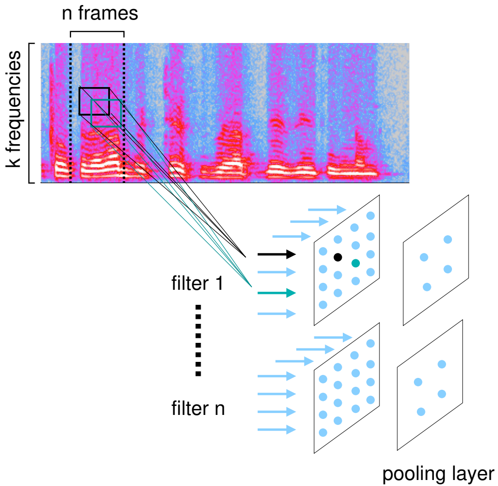
Alternative model (probably) required!

# Clockwork RNNs and SampleRNN [15, 10]




- Use a hierarchical RNN architecture - multiple tiers
  - each tier operates at a different timsescale: in diagram
    Tier1: 16KHz  Tier2: 4KHz  Tier3: 1KHz
  - like an (analogue) clock: hours, minutes, seconds
- Allows long-term dependencies in RNN architectures

# Convolutional Neural Networks [12]



## Convolutional Neural Networks [25]

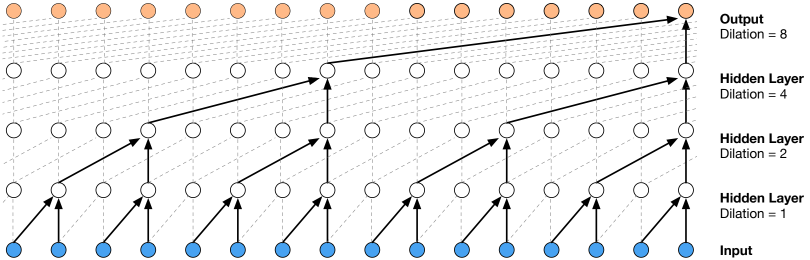
- Various parameters control the form of the CNN:
  - number (depth): how many filters to use
  - receptive field (filter size): height/width/depth ($h \times w \times d$)
  - stride: how far filter moves in the convolution
  - dilation: "gaps" between filter elements
  - zero-padding: do you pad the edges of the "image" with zeroes
- Filter output can be stacked to yield depth for next layer
- To illustrate the impact consider 1-dimensional case - default:
  - zero-padding, stride=1, dilation=0

sequence: $1, 2, 3, 4, 5, 6$ filter: $1, 1, 1$

default: $3, 6, 9, 12, 15, 11$ no padding: $6, 9, 12, 15$

dilation=1: $4, 6, 9, 12, 8, 10$ stride=2: $3, 9, 15$

# WaveNet Neural Vocoder [19]




- Use CNN acting over the one-dimensional waveform sequence
  - dilation used to control growth in number of parameters
  - different dilations at different levels
- A gated activation function is used (layer $k$, filter $f$)

$$\tanh\left(\mathbf{w}_f^{(k)} * \boldsymbol{x}_{1:T}\right) \odot \boldsymbol{\sigma}\left(\mathbf{w}_g^{(k)} * \boldsymbol{x}_{1:T}\right)$$

# Training Criterion

- Simplest training criterion - minimise

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{t=1}^{T} \log(p(x_t|\mathbf{x}_{1:t-1};\boldsymbol{\theta})) \propto \sum_{t=1}^{T} \frac{(\hat{x}_t - x_t)^2}{\sigma^2}$$

  - $\hat{x}_t = f(\mathbf{x}_{1:t-1};\boldsymbol{\theta})$ is the prediction
  - Gaussian noise added to systematic system - is this good?
- Adopt "ASR-style" estimation - run as classification
  - quantise the values to predict $x_t \rightarrow x_t^{\mathrm{q}}$ ($\mu$-law)
  - minimise the cross-entropy to the real target $\mathbf{x}_{1:T} \rightarrow \mathbf{x}^{\mathrm{q}}_{1:T}$

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{t=1}^{T}\sum_{k=1}^{K} \delta(\omega_k, x_t^{\mathrm{q}}) \log(P(\omega_k|\mathbf{x}_{1:t-1};\boldsymbol{\theta}))$$

# Quantising or Not?

- Quantising the waveform allows classification training
  - no assumption about Gaussian cost functions (important)
- Training criterion interesting mix of quantised and continuous

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{t=1}^{T}\sum_{k=1}^{K}\delta(\omega_k, x_t^{\text{q}})\log(P(\omega_k|\boldsymbol{x}_{1:t-1};\boldsymbol{\theta}))$$

  - predict quantised value $x_t^{\text{q}}$
  - history is continuous $\boldsymbol{x}_{1:t-1}$
- Sample generation follows simple process:
  - obtain discrete distribution $P(x_t^{\text{q}}|\hat{\boldsymbol{x}}_{1:t-1};\boldsymbol{\theta})$
  - sample from discrete distribution to yield $\hat{x}_t^{\text{q}}$
  - convert from discrete to continuous $\hat{x}_t^{\text{q}} \rightarrow \hat{x}_t$
  - append to history $\hat{\boldsymbol{x}}_{1:t} = \{\hat{\boldsymbol{x}}_{1:t-1}, \hat{x}_t\}$, $t = t+1$, repeat

# Conditional Synthesis

- The models described so far have no conditioning on the text
  - generates speech like sound, but not speech
- Additional information (beyond context labels) used:
  - duration: LSTM-RNN-based phone duration
  - log-F0: autoregressive CNN-based log-F0 prediction models
- Add conditioning vector $\boldsymbol{\lambda}$ yields information about
  - text context $\boldsymbol{\lambda}_{1:T}^{(q)}$: up-sampled phonetic information from text
  - speaker $\boldsymbol{\lambda}^{(s)}$: speaker representation
- Activation function then has the form (ignoring gating term)
  - for filter $f$ of layer $k$

$$\tanh\left(\mathbf{w}_f^{(k)} * \boldsymbol{x}_{1:T} + \boldsymbol{v}_f^{(k)} * \boldsymbol{\lambda}_{1:T}^{(q)} + \left[\boldsymbol{u}_f^{(k)\mathsf{T}}\boldsymbol{\lambda}^{(s)}\right]_{1:T}\right) \odot \boldsymbol{\sigma}\left(\ldots\right)$$

  - where $[\ldots]_{1:T}$ indicates repeating the output from 1 to $T$

# Parallel Wavenet

# Parallel WaveNet [20]

- WaveNet generates high quality samples
  - but orders of magnitude too much latency
  - limits sampling frequency to 16KHz
  - only able to support 8-bit resolution

- Rather than swapping neural vocoder
  - modified output distribution to support 16-bit resolution
  - modified synthesis to allow parallel generation of samples
  - different architectures for training and synthesis

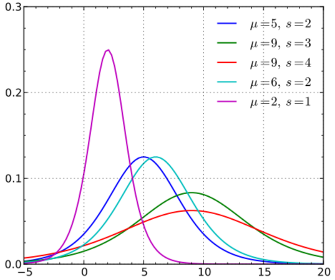
# Output Distribution: Logistic Distribution

- Interested in a distribution $\mathbb{L}(x;\mu,s)$ where

$$\int_{-\infty}^{a}\mathbb{L}(x;\mu,s)dx=\sigma\left(\frac{a-\mu}{s}\right);\quad \sigma(x)=\frac{1}{1+\exp(-x)}$$

  - $\sigma()$ is the logistic function - a valid CDF

- This is the Logistic distribution




- the form of the PDF is then

$$\mathbb{L}(x;\mu,s)=\frac{\exp((x-\mu)/s)}{s\left(1+\exp((x-\mu)/s)\right)^2}$$

- $\mu$ is the mean (location) of the distribution
- $s>0$ is the scale parameter

# Discretized Logistic Mixture Likelihood [16]

- Modelling using a softmax for the distribution limiting
  - doesn't scale well with the number of bin values
- Using the CDF of Logistic distributed variables

$$\int_a^b \mathbb{L}(x; \mu, s)dx = \sigma\left(\frac{b-\mu}{s}\right) - \sigma\left(\frac{a-\mu}{s}\right)$$

- Extending to a mixture model yields and examining a bin value

$$P(x^{\mathrm{q}}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{s}) = \sum_{m=1}^{M} \pi_m \left(\sigma\left(\frac{x^{\mathrm{q}} + 0.5 - \mu_m}{s_m}\right) - \sigma\left(\frac{x^{\mathrm{q}} - 0.5 - \mu_m}{s_m}\right)\right)$$

  - recognises that bins 127 and 129 are near 128
  - allows expansion of targets with increasing cost

# Parallel Generation: Inverse Autoregressive Flows (IAFs) [9]

- Generate $T$, independent Logistic distributed, samples $\boldsymbol{z}_{1:T}$
  - apply (causal) transformation from $\boldsymbol{z}_{1:T} \to \boldsymbol{x}_{1:T}$

$$\log(p(\boldsymbol{x}_{1:T})) = \log(p(\boldsymbol{z}_{1:T})) - \log\left(\left|\frac{d\boldsymbol{x}_{1:T}}{d\boldsymbol{z}_{1:T}}\right|\right); \quad x_t = f(\boldsymbol{z}_{1:t})$$

  - possible to show that (from the causal aspect of transform)

$$\log\left(\left|\frac{d\boldsymbol{x}_{1:T}}{d\boldsymbol{z}_{1:T}}\right|\right) = \sum_{t=1}^{T} \log\left(\frac{\partial f(\boldsymbol{z}_{1:t})}{\partial z_t}\right)$$

- The following transformation is applied

$$x_t = z_t f_{\mathrm{s}}(\boldsymbol{z}_{1:t-1};\boldsymbol{\theta}) + f_{\mu}(\boldsymbol{z}_{1:t-1};\boldsymbol{\theta})$$

  - $f_{\mathrm{s}}(\boldsymbol{z}_{1:t-1};\boldsymbol{\theta})$: scale parameter prediction from $\boldsymbol{z}_{1:t-1}$
  - $f_{\mu}(\boldsymbol{z}_{1:t-1};\boldsymbol{\theta})$: location parameter prediction from $\boldsymbol{z}_{1:t-1}$

## Inverse Autoregressive Flows (IAFs) (cont)

- As $z_t$ is Logistic ($\mathbb{L}(0,1)$) distributed this means that

$$p(x_t|\boldsymbol{z}_{1:t-1};\boldsymbol{\theta}) = \mathbb{L}(x_t; f_\mu(\boldsymbol{z}_{1:t-1};\boldsymbol{\theta}), f_{\rm s}(\boldsymbol{z}_{1:t-1};\boldsymbol{\theta}))$$

  - $f_\mu(\boldsymbol{z}_{1:t-1};\boldsymbol{\theta})$ and $f_{\rm s}(\boldsymbol{z}_{1:t-1};\boldsymbol{\theta})$ from WaveNet style network
- Allows all samples $\boldsymbol{x}_{1:T}$ to be generated in parallel
  - $x_t$ no explicit dependence on previous generated samples $\boldsymbol{x}_{1:t-1}$
  - $x_t$ only depends on the noise samples $\boldsymbol{z}_{1:t-1}$
  - can generate $\boldsymbol{x}_{1:T}$ in parallel given $\boldsymbol{z}_{1:T}$ - Parallel WaveNet
- Transforms random sequence $\boldsymbol{z}_{1:T}$ to structured $\boldsymbol{x}_{1:T}$
  - too challenging to achieve with a single "flow"
  - stack multiple flows together for parallel WaveNet

# Training: Probability Density Distillation [20]

- Training parallel WaveNet directly is slow
  - training log-likelihood calculation is sequential (see IAF)
  - WaveNet training efficient (prediction $x_t$ uses real data $\boldsymbol{x}_{1:t-1}$)
- Given a trained WaveNet $p_{\mathtt{w}}()$ train $p(\boldsymbol{x};\boldsymbol{\theta})$ to minimise

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{D}_{\mathtt{kl}}(p()||p_{\mathtt{w}}()) = \sum_{\boldsymbol{x}\in\mathcal{X}} p(\boldsymbol{x};\boldsymbol{\theta})\log\left(\frac{p(\boldsymbol{x};\boldsymbol{\theta})}{p_{\mathtt{w}}(\boldsymbol{x})}\right)$$

  - $\mathcal{X}$ is the set of all waveforms
  - contrast to teacher-student training

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}\in\mathcal{X}} p_{\mathtt{w}}(\boldsymbol{x})\log\left(\frac{p_{\mathtt{w}}(\boldsymbol{x})}{p(\boldsymbol{x};\boldsymbol{\theta})}\right)$$

- Two terms in criterion (both can be computed efficiently)
  - student entropy
  - cross-entropy

# Student Entropy

- Student entropy for a sequence of length $T$

$$\sum_{\boldsymbol{x}_{1:T} \in \mathcal{X}} p(\boldsymbol{x}_{1:T};\boldsymbol{\theta}) \log\left(p(\boldsymbol{x}_{1:T};\boldsymbol{\theta})\right) = -\mathbb{E}\left\{\sum_{t=1}^{T} \log\left(p(x_t|\boldsymbol{z}_{1:t-1};\boldsymbol{\theta})\right)\right\}_{p(\boldsymbol{z}_{1:T})}$$

$$= \mathbb{E}\left\{\sum_{t=1}^{T} \log\left(f_{\mathrm{s}}(\boldsymbol{z}_{1:t-1};\boldsymbol{\theta})\right)\right\}_{p(\boldsymbol{z}_{1:T})} + 2T$$

  - uses equality for entropy of a logistic distribution
  - can be computed without generating samples

# Cross-Entropy (to WaveNet model)

- The cross-entropy term has the following form

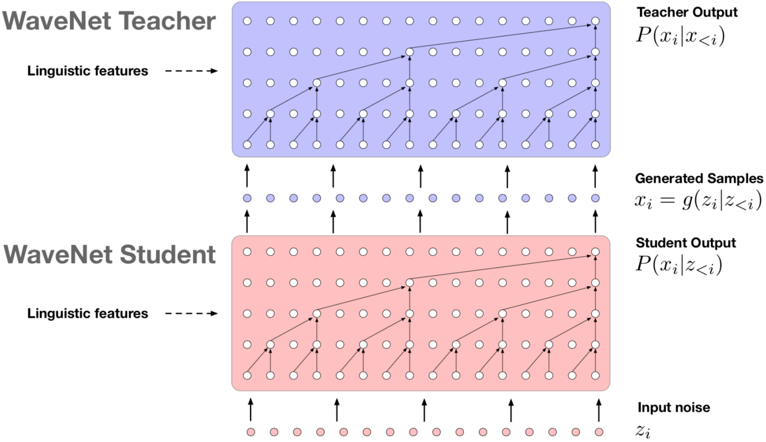
$$\sum_{\boldsymbol{x}_{1:T}\in\mathcal{X}} p(\boldsymbol{x}_{1:T};\boldsymbol{\theta})\log\left(p_{\mathtt{w}}(\boldsymbol{x}_{1:T})\right)$$

  - this requires drawing sample sequences from $p(\boldsymbol{x};\boldsymbol{\theta})$
- For a $T$ length sequence this can be written as (see paper)

$$\sum_{t=1}^{T}\mathbb{E}\left\{p(x_t|\boldsymbol{x}_{1:t-1};\boldsymbol{\theta})\log\left(p_{\mathtt{w}}(x_t|\boldsymbol{x}_{1:t-1})\right)\right\}_{p(\boldsymbol{x}_{1:t-1};\boldsymbol{\theta})}$$

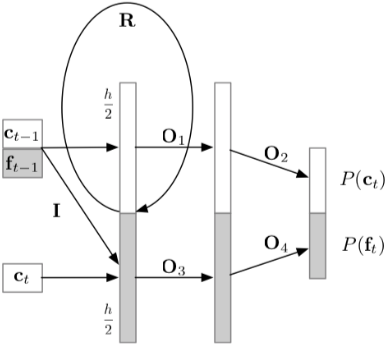
  - this is efficient (see paper!)

# Probability Density Distillation Overview



# WaveRNN

# WaveRNN [6]




- Modification to original WaveNet configuration
  - simplified (recurrent) architecture
  - split representation coarse ($\boldsymbol{c}_t$) and fine ($\boldsymbol{f}_t$)
  - efficient, improved, resolution

# WaveRNN Diagram Notation

- Split the 16-bit output into:
  - coarse, $\boldsymbol{c}_t$ 8 most significant bits
  - fine, $\boldsymbol{f}_t$ 8 least significant bits (conditioned on coarse)
- Diagram (from paper) uses notation
  - $\boldsymbol{O}_1$ and $\boldsymbol{O}_2$: "coarse" weight matrices
  - $\boldsymbol{O}_3$ and $\boldsymbol{O}_4$: "fine" weight matrices
  - $\mathbf{I}$ weight matrix (not identity!) from paper
- For the network layers
  - first layer recurrent (variant on GRU)
  - second layer ReLU
  - third layer soft-max

(Additional speed-ups descibed in paper: sparse WaveRNN)

# WaveRNN Architecture

- Split the prediction 16-bit prediction into 2 parts

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{c}_t \\ \boldsymbol{f}_t \end{bmatrix}$$

  - coarse $\boldsymbol{c}_t$ (8 bits - 256 levels): top 8 bits of waveform

$$\boldsymbol{c}_t = \mathcal{F}_{\text{c}}(\boldsymbol{x}_{t-1}, \boldsymbol{h}_{t-1}) = \mathcal{F}_{\text{c}}(\boldsymbol{c}_{t-1}, \boldsymbol{f}_{t-1}, \boldsymbol{h}_{t-1})$$

  - fine $\boldsymbol{f}_t$ (8 bits - 256 levels): bottom 8 bits of waveform

$$\boldsymbol{f}_t = \mathcal{F}_{\text{c}}(\boldsymbol{c}_{t-1}, \boldsymbol{f}_{t-1}, \boldsymbol{h}_{t-1}, \boldsymbol{c}_t)$$

# WaveRNN Coarse Generation

- Equations can be split into distinct parts (ignoring biases)

$$\begin{aligned} P(\boldsymbol{c}_t) &= \mathtt{softmax}(\mathbf{O}_2\mathtt{relu}(\mathbf{O}_1\mathbf{z}_t^{\mathrm{c}})) \\ \boldsymbol{z}_t^{\mathrm{c}} &= \boldsymbol{u}_t^{\mathrm{c}} \odot \boldsymbol{h}_{t-1}^{\mathrm{c}} + (\mathbf{1} - \boldsymbol{u}_t^{\mathrm{c}}) \odot \boldsymbol{e}_t^{\mathrm{c}} \\ \boldsymbol{e}_t^{\mathrm{c}} &= \mathtt{tanh}(\boldsymbol{r}_t^{\mathrm{c}} \odot (\mathbf{W}_{\mathrm{e}}^{\mathrm{c}}\boldsymbol{h}_{t-1}) + \mathbf{W}_{\mathtt{x}}^{\mathrm{c}}\boldsymbol{x}_{t-1}) \end{aligned}$$

  - where the gates have the form

$$\begin{aligned} \boldsymbol{r}_t^{\mathrm{c}} &= \mathtt{sigmoid}(\mathbf{W}_{\mathtt{rh}}^{\mathrm{c}}\boldsymbol{h}_{t-1} + \mathbf{W}_{\mathtt{rx}}^{\mathrm{c}}\boldsymbol{x}_{t-1}) \\ \boldsymbol{u}_t^{\mathrm{c}} &= \mathtt{sigmoid}(\mathbf{W}_{\mathtt{uh}}^{\mathrm{c}}\boldsymbol{h}_{t-1} + \mathbf{W}_{\mathtt{ux}}^{\mathrm{c}}\boldsymbol{x}_{t-1}) \end{aligned}$$

  - and the input and the history have the form

$$\boldsymbol{x}_{t-1} = \begin{bmatrix} \boldsymbol{c}_{t-1} \\ \boldsymbol{f}_{t-1} \end{bmatrix}; \quad \boldsymbol{h}_{t-1} = \begin{bmatrix} \boldsymbol{z}_{t-1}^{\mathrm{c}} \\ \boldsymbol{z}_{t-1}^{\mathtt{f}} \end{bmatrix}$$

# WaveRNN Fine Generation

- Initially the coarse elements are generated

$$\boldsymbol{c}_t \sim P(\boldsymbol{c}_t)$$

- Equations can be split into distinct parts

$$\begin{aligned}
P(\boldsymbol{c}_t) &= \texttt{softmax}(\mathbf{O}_4\texttt{relu}(\mathbf{O}_3\mathbf{z}^{\text{f}}_t)) \\
\boldsymbol{z}^{\text{f}}_t &= \boldsymbol{u}^{\text{f}}_t \odot \boldsymbol{h}^{\text{f}}_{t-1} + (\mathbf{1} - \boldsymbol{u}^{\text{f}}_t) \odot \boldsymbol{e}^{\text{f}}_t \\
\boldsymbol{e}^{\text{f}}_t &= \texttt{tanh}(\boldsymbol{r}^{\text{f}}_t \odot (\mathbf{W}^{\text{f}}_{\texttt{e}}\boldsymbol{h}_{t-1}) + \mathbf{W}^{\text{f}}_{\texttt{x}}\boldsymbol{x}_{t-1} + \mathbf{W}_{\texttt{uc}}\boldsymbol{c}_t)
\end{aligned}$$

  - where the "fine" gates have the form

$$\begin{aligned}
\boldsymbol{r}^{\text{f}}_t &= \texttt{sigmoid}(\mathbf{W}^{\text{f}}_{\texttt{rh}}\boldsymbol{h}_{t-1} + \mathbf{W}^{\text{f}}_{\texttt{rx}}\boldsymbol{x}_{t-1} + \mathbf{W}_{\texttt{rc}}\boldsymbol{c}_t) \\
\boldsymbol{u}^{\text{f}}_t &= \texttt{sigmoid}(\mathbf{W}^{\text{f}}_{\texttt{uh}}\boldsymbol{h}_{t-1} + \mathbf{W}^{\text{f}}_{\texttt{ux}}\boldsymbol{x}_{t-1} + \mathbf{W}_{\texttt{uc}}\boldsymbol{c}_t)
\end{aligned}$$

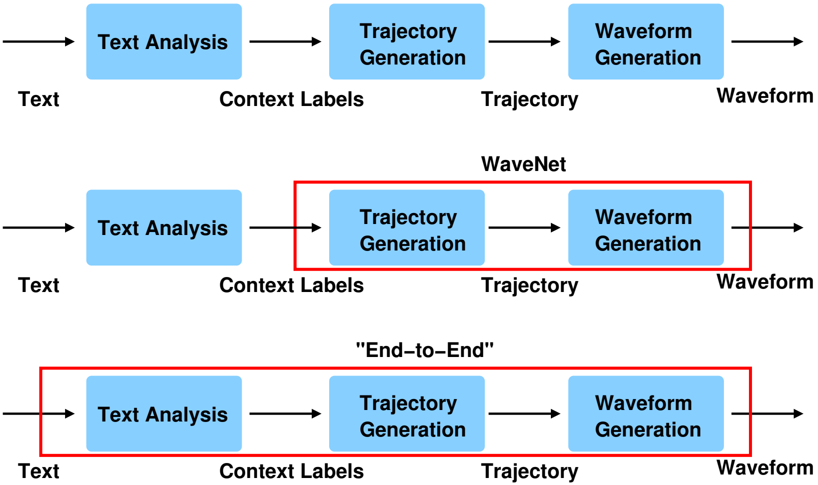
  - input, $\boldsymbol{x}_{t-1}$, and history, $\boldsymbol{h}_{t-1}$, same as coarse

# Conditioning Vectors

# Conditional Synthesis

- To generate speech require conditioning on text:
  - convert text to context features
  - generate durations for each of model states
  - append "position" features to context
- Yields context features at 200Hz (5ms frame rate)
  - up-sample to yield required (16KHz) context features $\boldsymbol{\lambda}_{1:T}^{(q)}$
- Requires linguistic information:
  - text processing and POS tags
  - phonetic lexicon (may introduce errors)
  - phone/state duration models

# Synthesis Pipelines



# "End-to-End" Synthesis

- Use deep-learning to derive $\boldsymbol{\lambda}_{1:T}^{(q)}$
  - map word sequence $\boldsymbol{\omega}_{1:L}$ to character sequence $\mathbf{c}_{1:P}$
  - map character sequence $\mathbf{c}_{1:P}$ to context features $\boldsymbol{\lambda}_{1:T}^{(q)}$
  - optimise mapping for synthesis performance
  - note increase in sample rate $\approx$ 10-20 Hz to 16Hz
- To handle difference in required sample rate use attention
  - use decode state, $\boldsymbol{s}_{\tau-1}$ as key for attention at time $\tau$

$$
\begin{aligned}
\boldsymbol{c}_{1:P} &\rightarrow \boldsymbol{h}_{1:P}(\texttt{character encoding}) \\
\boldsymbol{\alpha}_{\tau} &= \mathcal{F}_{\texttt{att}}(\boldsymbol{s}_{\tau-1}, \boldsymbol{\alpha}_{\tau-1}, \boldsymbol{h}_{1:P}); \quad \boldsymbol{g}_{\tau} = \sum_{i=1}^{P} \alpha_{\tau i} \boldsymbol{h}_i \\
\boldsymbol{y}_{\tau} &= \texttt{generate}(\boldsymbol{s}_{\tau-1}, \boldsymbol{g}_{\tau}); \quad \boldsymbol{s}_{\tau} = \mathcal{F}_{\texttt{rec}}(\boldsymbol{s}_{\tau-1}, \boldsymbol{g}_{\tau}, \boldsymbol{y}_{\tau})
\end{aligned}
$$

  - form of $\mathcal{F}_{\texttt{att}}(.)$ very general

# Content and Location Based Attention [2]

- Content-based attention - seen before
  - operates between the key and elements of the encoding

$$\boldsymbol{\alpha}_\tau = \mathcal{F}_{\texttt{att}}(\boldsymbol{s}_{\tau-1}, \boldsymbol{h}_{1:P}); \quad \alpha_{\tau,i} = \frac{\exp(\mathbf{f}(\boldsymbol{s}_{\tau-1}, \boldsymbol{h}_i))}{\sum_{j=1}^{P} \exp(\mathbf{f}(\boldsymbol{s}_{\tau-1}, \boldsymbol{h}_j))}$$

$$\mathbf{f}(\boldsymbol{s}_{\tau-1}, \boldsymbol{h}_i) = \mathbf{w}^{\mathsf{T}} \texttt{tanh}(\mathbf{W}_{\texttt{s}} \boldsymbol{s}_{\tau-1} + \mathbf{W}_{\texttt{h}} \boldsymbol{h}_i)$$

- Location-based attention
  - operates between key and previous attention vector

$$\boldsymbol{\alpha}_\tau = \mathcal{F}_{\texttt{att}}(\boldsymbol{s}_{\tau-1}, \boldsymbol{\alpha}_{\tau-1});$$
$$\mathbf{f}(\boldsymbol{s}_{\tau-1}, \boldsymbol{\alpha}_{\tau-1}, i) = \mathbf{w}^{\mathsf{T}} \texttt{tanh}(\mathbf{W}_{\texttt{s}} \boldsymbol{s}_{\tau-1} + \mathbf{W}_{\alpha} [\mathbf{F} \star \boldsymbol{\alpha}_{\tau-1})]_i)$$

  - **F** is the matrix for progression (location) of attention
- Hybrid attention combines attributes of both

# Mixture of Gaussian Functions Attention [5]

- For sequence data left-to-right emphasis useful
  - reduces chance of inappropriate attention for long sequences
- Use modified version of location attention mechanism
  - convolve with an $M$-component mixture of Gaussian functions

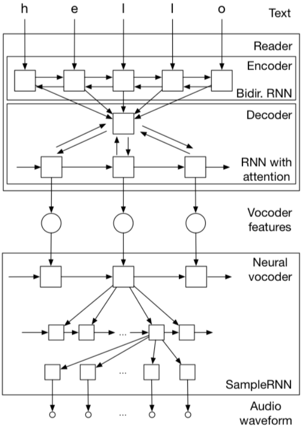
$$\alpha_{\tau i} = \sum_{k=1}^{M} a_t^{(k)} \exp\left(-b_t^{(k)}(\kappa_t^{(k)} - i)^2\right); \quad \boldsymbol{g}_\tau = \sum_{i=1}^{P} \alpha_{\tau i} \boldsymbol{h}_i$$

  - where $\hat{a}_t^{(k)}, \hat{b}_t^{(k)}$ and $\hat{\kappa}_t^{(k)}$ are predicted from the network

$$\begin{aligned} a_t^{(k)} &= \exp(\hat{a}_t^{(k)}); \quad b_t^{(k)} = \exp(\hat{b}_t^{(k)}) \\ \kappa_t^{(k)} &= \kappa_{t-1}^{(k)} + \exp(\hat{\kappa}_t^{(k)}) \end{aligned}$$

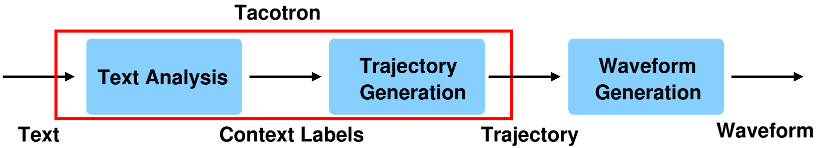
  - has been used for end-to-end synthesis and recognition

# Char2Wav [18]




- Early end-to-end architecture
- Uses location attention for context
  - predicts vocoder features
  - neural vocoder for synthesis
- SampleRNN neural vocoder used

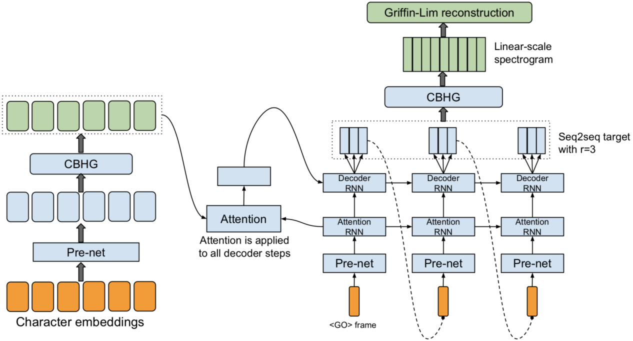
# "End-to-End" Spectrogram Generation




- "End-to-End" systems attractive - just deep-learning
  - requires large quantities of accurately transcribed data
- Alternative approach - predict spectrograms
  - replace text-processing/trajectory models with single block
  - standard sequence-to-sequence mode

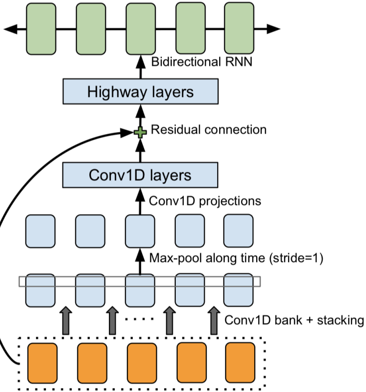
Still needs a vocoder!

# Tacotron [23] (Not Waveform-Level)



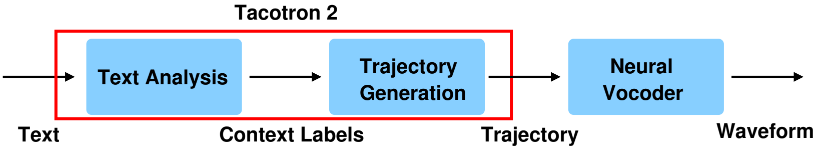

- End-to-end configurations - characters as input
  - content-based attention mechanism
  - CBHG layer added for post-processing
  - predicts linear spectral magnitude (not vocoder parameters)
  - Griffin Lim is used as the synthesiser

# CBHG Layer



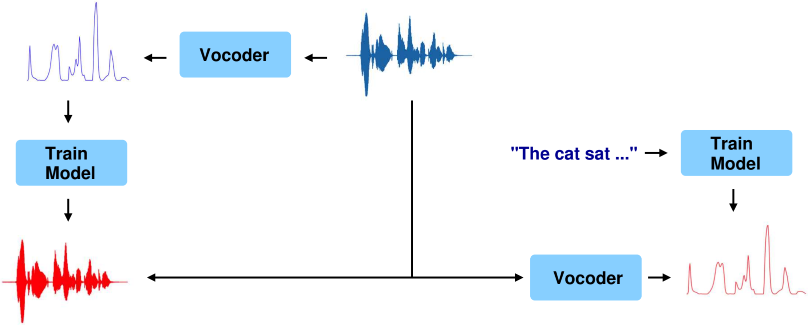
Bidirectional RNN
Highway layers
Residual connection
Conv1D layers
Conv1D projections
Max-pool along time (stride=1)
Conv1D bank + stacking

# "Neural Vocoder"



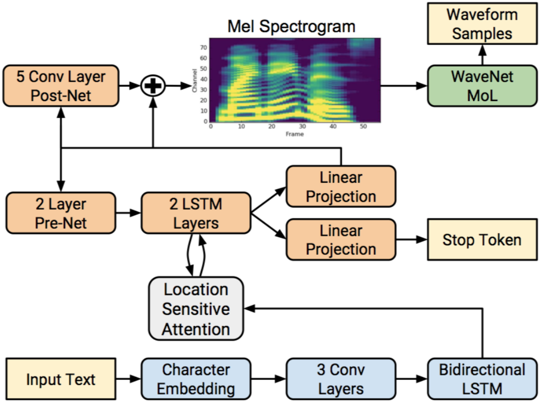

- Split Process into two deep-learning parts
  - text to spectrograms (mel-spectrograms can be used)
  - spectrogram to waveform generation
- (1) has previously been described (e.g. Tacotron)
- (2) can be trained on large amounts of data
  - just needs audio data (waveform)
  - learn a mapping from real spectrogram to real waveform

# Training Neural Vocoders




- Neural vocoder training does not need transcriptions
  - does require high quality trajectories at synthesis time

# Tacotron 2 [17, 23]




- Integrates (modified) Tacotron with (modified) WaveNet

# Tacotron 2 Modifications

- Changes to the original Tacotron paper:
  - predict Mel-spectrogram not linear
  - simplified network configuration for encoder/decoder
  - location-based attention mechanism

- Changes to the original WaveNet paper:
  - condition on Mel-spectrogram (not linguistic features)
  - uses parallel WaveNet for speed

# Speaker Representations

# Variable Length Mapping

- Range of applications make use of speaker representations
  - speaker clustering
  - speaker recognition/verification
  - speaker adaptation
- Simplest form is multi-speaker (1-of-K coding)
- More generally require a fixed-length representation
  - variable length sequence $\boldsymbol{x}_{1:T}^{(s)} \rightarrow$ speaker representation $\boldsymbol{\lambda}^{(s)}$

$$\boldsymbol{\lambda}^{(s)} = \phi(\boldsymbol{x}_{1:T}^{(s)})$$

  - in 4F10 already seen application using SVMs
  - makes use of Fisher Kernel
- Can make use of a UBM ($\boldsymbol{\theta}_{\texttt{ubm}}$)
  - large GMM used to represent all speakers
  - MAP adapt model to target speaker $\boldsymbol{\theta}_{\texttt{ubm}}^{(s)}$

# Fisher Kernel [14]

- From 4F10 lectures

$$\phi(\boldsymbol{x}_{1:T}) = \left[ \begin{array}{c} \log(p(\boldsymbol{x}_{1:T}|\boldsymbol{\theta}^{(s)}_{\texttt{ubm}})) - \log(p(\boldsymbol{x}_{1:T}|\boldsymbol{\theta}_{\texttt{ubm}})) \\ \nabla_{\boldsymbol{\theta}} \log\left(p\left(\boldsymbol{x}_{1:T}|\boldsymbol{\theta}\right)\right)\big|_{\boldsymbol{\theta}^{(s)}_{\texttt{ubm}}} \end{array} \right]$$

  - the first term is the standard GMM-based score
  - the second term is Fisher score for the speaker model
  - only derivatives wrt the mean parameters used
- If only the derivative part is used then

$$\phi(\boldsymbol{x}_{1:T}) = \left[ \begin{array}{c} \sum_{t=1}^{T} P(1|\boldsymbol{x}_t, \boldsymbol{\theta}^{(s)}_{\texttt{ubm}}) \boldsymbol{\Sigma}^{(s1)-1}_{\texttt{ubm}} (\boldsymbol{x}_t - \boldsymbol{\mu}^{(s1)}_{\texttt{ubm}}) \\ \vdots \\ \sum_{t=1}^{T} P(\texttt{M}|\boldsymbol{x}_t, \boldsymbol{\theta}^{(s)}_{\texttt{ubm}}) \boldsymbol{\Sigma}^{(sM)-1}_{\texttt{ubm}} (\boldsymbol{x}_t - \boldsymbol{\mu}^{(sM)}_{\texttt{ubm}}) \end{array} \right]$$

- Fisher Information Matrix is sometimes used as a metric

# Mean Super-Vector Kernel [1]

- Rather than taking derivative, it is possible to use parameters
  - consider the means of the speaker adapted UBM

$$\boldsymbol{\lambda}^{(s)} = \phi(\boldsymbol{x}_{1:T}^{(s)}) = \begin{bmatrix} \boldsymbol{\mu}_{\texttt{ubm}}^{(s1)} \\ \vdots \\ \boldsymbol{\mu}_{\texttt{ubm}}^{(sm)} \\ \vdots \\ \boldsymbol{\mu}_{\texttt{ubm}}^{(sM)} \end{bmatrix}$$

- Both this form and Fisher Kernel yield large spaces
  - if only means used $M \times d$ elements
  - originally used for SVM-based systems (see 4F10)
- Can we make the speaker information more compact?

# Joint Factor Analysis (JFA) [8]

- The actual observed data is impacted by multiple factors
  - speaker (desired variability to model)
  - channel/session attributes (not desired)
- Decomposing the mean supervector yields

$$\boldsymbol{\lambda}^{(s)} = \boldsymbol{\mu}_{\mathtt{si}} + \mathbf{V}\boldsymbol{\lambda}^{(s)}_{\mathtt{sp}} + \mathbf{U}\boldsymbol{\lambda}^{(s)}_{\mathtt{ch}} + \mathbf{D}\boldsymbol{z}$$

  - $\mathbf{V}$ and $\mathbf{U}$ and loading matrices
  - $\boldsymbol{\mu}_{\mathtt{si}}$ is the speaker-independent mean
  - $\boldsymbol{\lambda}^{(s)}_{\mathtt{sp}}$ point in speaker-space (prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$)
  - $\boldsymbol{\lambda}^{(s)}_{\mathtt{ch}}$ point in channel/session-space (prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$)
  - $\mathbf{D}$ the noise matrix, $\boldsymbol{z}$ noise term (prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$)
- Effectively a large Gaussian distribution: typical dimensions
  - $\boldsymbol{\lambda}^{(s)}$: 20000; $\boldsymbol{\lambda}^{(s)}_{\mathtt{sp}}$: 300; $\boldsymbol{\lambda}^{(s)}_{\mathtt{ch}}$: 100
  - iterative training process - see paper

# iVector Model Training [3]

- Identity Vector (iVector): simplify JFA merge speaker/channel

$$\boldsymbol{\lambda}^{(s)} = \boldsymbol{\mu}_{\texttt{si}} + \mathbf{T}\boldsymbol{\lambda}_{\texttt{sp}}^{(s)}$$

  - $\mathbf{T}$ is the total variability matrix
  - $\boldsymbol{\lambda}_{\texttt{sp}}^{(s)}$ point in speaker-space (prior $\mathcal{N}(\mathbf{0},\mathbf{I})$)
- This is similar to Factor Analysis: use EM
  - unobserved: speaker $\boldsymbol{\lambda}_{\texttt{sp}}$, component at $t$ $P(m|\boldsymbol{\lambda}_{\texttt{sp}}, \boldsymbol{x}_t^{(s)}\boldsymbol{\theta})$

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\theta},\hat{\boldsymbol{\theta}}) &= \sum_{s=1}^{S}\int p(\boldsymbol{\lambda}_{\texttt{sp}}|\boldsymbol{\theta},\boldsymbol{x}_{1:T}^{(s)})\sum_{t=1}^{T}\sum_{m=1}^{M}P(m|\boldsymbol{\lambda}_{\texttt{sp}},\boldsymbol{x}_t^{(s)},\boldsymbol{\theta}) \\ &\qquad \log\left(\mathcal{N}(\boldsymbol{x}_t^{(s)};\hat{\boldsymbol{\mu}}_{\texttt{si}}^{(m)}+\hat{\mathbf{T}}^{(m)}\boldsymbol{\lambda}_{\texttt{sp}},\hat{\boldsymbol{\Sigma}}^{(m)})\right)d\boldsymbol{\lambda}_{\texttt{sp}}\end{aligned}$$

  - new model parameters $\hat{\boldsymbol{\theta}} = \left\{\ldots,\hat{\mathbf{T}}^{(m)},\hat{\boldsymbol{\mu}}_{\texttt{si}}^{(m)},\hat{\boldsymbol{\Sigma}}^{(m)},\ldots\right\}$
  - for simplicity $P(m|\boldsymbol{\lambda}_{\texttt{sp}},\boldsymbol{x}_t^{(s)},\boldsymbol{\theta})$ often fixed for training

# iVector Extraction [7, 11, 4]

- At test-time iVector extracted using

$$\hat{\boldsymbol{\lambda}}_{\text{sp}}^{(s)} = \arg\max_{\boldsymbol{\lambda}_{\text{sp}}} \left\{ p(\boldsymbol{\lambda}_{\text{sp}} | \boldsymbol{x}_{1:T}^{(s)}, \boldsymbol{\theta}) \right\}$$

  - again EM is used to find iVector
- Model related to CAT and EigenVoices
  - point estimate of $\boldsymbol{\lambda}_{\text{sp}}^{(s)}$ used, rather than distribution
  - treated as part of the parameter estimation stage

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) &= \sum_{s=1}^{S} \sum_{m=1}^{M} \sum_{t=1}^{T} P(m | \boldsymbol{\lambda}_{\text{sp}}^{(s)}, \boldsymbol{x}_t^{(s)} \boldsymbol{\theta}) \Big[ \log(P(\hat{\boldsymbol{\lambda}}_{\text{sp}}^{(s)})) \\ &\quad + \log\left( \mathcal{N}(\boldsymbol{x}_t^{(s)}; \hat{\boldsymbol{\mu}}_{\text{si}}^{(m)} + \hat{\mathbf{T}}^{(m)} \hat{\boldsymbol{\lambda}}_{\text{sp}}^{(s)}, \hat{\boldsymbol{\Sigma}}^{(m)}) \right) \Big] \end{aligned}$$

  - possible to factorise $\boldsymbol{\lambda}_{\text{sp}}^{(s)}$ (JFA) include orthogonality constraint

# iVectors for Speaker Recognition

- Extract iVectors for all enrolled speakers, $\boldsymbol{\lambda}_{\text{sp}}^{(1)}, \ldots, \boldsymbol{\lambda}_{\text{sp}}^{(S)}$
  - extract for test speaker $\boldsymbol{\lambda}_{\text{sp}}$
  - need to select "closest" enrolled speaker
- For speed look at distances between iVectors

$$\hat{s} = \arg\min_{s} \left\{ d(\boldsymbol{\lambda}_{\text{sp}}, \boldsymbol{\lambda}_{\text{sp}}^{(s)}) \right\}$$

  - euclidean distance:

$$d(\boldsymbol{\lambda}_{\text{sp}}, \boldsymbol{\lambda}_{\text{sp}}^{(s)}) = ||\boldsymbol{\lambda}_{\text{sp}} - \boldsymbol{\lambda}_{\text{sp}}^{(s)}||^2$$

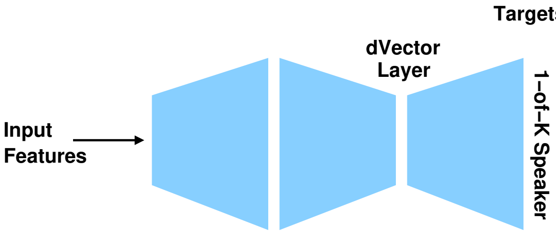
  - (-) cosine distance:

$$d(\boldsymbol{\lambda}_{\text{sp}}, \boldsymbol{\lambda}_{\text{sp}}^{(s)}) = -\frac{\boldsymbol{\lambda}_{\text{sp}}^{\mathsf{T}} \boldsymbol{\lambda}_{\text{sp}}^{(s)}}{\sqrt{\boldsymbol{\lambda}_{\text{sp}}^{\mathsf{T}} \boldsymbol{\lambda}_{\text{sp}} \boldsymbol{\lambda}_{\text{sp}}^{(s)\mathsf{T}} \boldsymbol{\lambda}_{\text{sp}}^{(s)}}}$$

  popular choice (empirically good!)

# dVector Representation [21]

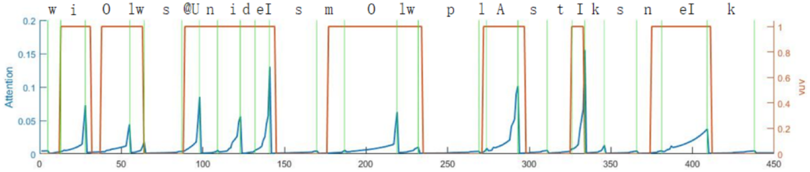
- Train vector to discriminate between speakers
  - related to bottleneck features for ASR




- Targets are a 1-of-K coding of speaker
  - wide window of features to yield good performance
- Simple approach used to handle temporal aspect of signal
  - $\boldsymbol{\lambda}_t^{(s)}$ is the vector for frames centered at time $t$

$$\boldsymbol{\lambda}^{(s)} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{\lambda}_t^{(s)}$$

# Attention-based dVector Representation [22]

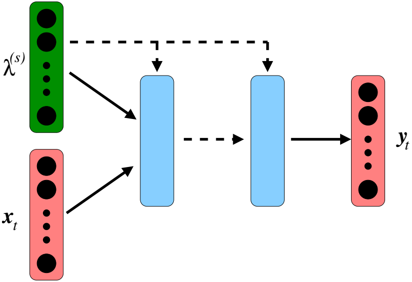



- Rather than using simple averaging of d-Vectors, use attention

$$\boldsymbol{\lambda}^{(s)} = \sum_{t=1}^{T} \alpha_t \boldsymbol{\lambda}_t^{(s)}$$

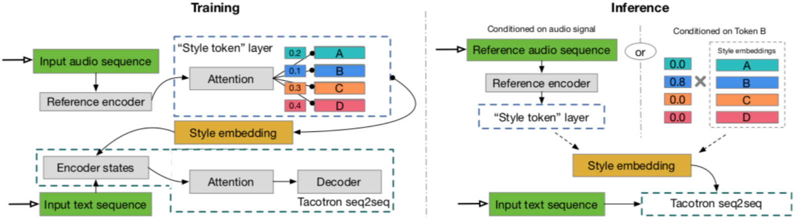
  - self-attention mechanism used based on context features
  - model trained in an integrated fashion (including attention)
- Attention focuses on voiced regions

# Speaker Representations for Adaptation




- Speaker representation can be used as auxiliary information
  - simple for of speaker adaptation
  - no initial hypothesis required
  - can be optionally be applied to other layers of network

# General Style Tokens [24]




- In addition to speaker, also interested in style
  - speakers prosody varies with context
  - narrative speech, emotions, audiobooks
- It is hard to consistently label "style"
  - unsupervised approaches increasingly popular
  - general style tokens for Tacotron one example

[1] W. Campbell, D.Sturim, D. Reynolds, and A. Solomonoff, "SVM based speaker verification using a GMM supervector kernel and NAP variability compensation," in *Proc. ICASSP*, 2006.

[2] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *CoRR*, vol. abs/1506.07503, 2015. [Online]. Available: http://arxiv.org/abs/1506.07503

[3] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Trans. Audio, Speech & Language Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[4] M. J. F. Gales, "Cluster adaptive training of hidden Markov models," *IEEE Transactions Speech and Audio Processing*, vol. 8, pp. 417–428, 2000.

[5] A. Graves, "Generating sequences with recurrent neural networks," *CoRR*, vol. abs/1308.0850, 2013. [Online]. Available: http://arxiv.org/abs/1308.0850

[6] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," *CoRR*, vol. abs/1802.08435, 2018. [Online]. Available: http://arxiv.org/abs/1802.08435

[7] P. Karanasou, Y. Wang, M. Gales, and P. Woodland, "Adaptation of deep neural network acoustic models using factorised i-vectors," in *Proceedings of Interspeech'14*, 2014.

[8] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition," *IEEE Trans. Audio, Speech & Language Processing*, vol. 15, no. 4, pp. 1435–1447, 2007.

[9] D. P. Kingma, T. Salimans, and M. Welling, "Improving variational inference with inverse autoregressive flow," *CoRR*, vol. abs/1606.04934, 2016. [Online]. Available: http://arxiv.org/abs/1606.04934

[10] J. Koutník, K. Greff, F. J. Gomez, and J. Schmidhuber, "A clockwork RNN," *CoRR*, vol. abs/1402.3511, 2014. [Online]. Available: http://arxiv.org/abs/1402.3511

[11] R. Kuhn, P. Nguyen, J.-C. Junqua, L. Goldwasser, N. Niedzielski, S. Fincke, K. Field, and M. Contolini, "Eigenvoices for speaker adaptation," in *Proceedings ICSLP*, 1998, pp. 1771–1774.

[12] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[13] Z.-H. Ling, S.-Y. Kang, H. Zen, A. Senior, M. Schuster, X.-J. Qian, H. M. Meng, and L. Deng, "Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 35–52, 2015.

[14] C. Longworth and M. Gales, "Derivative and parametric kernels for speaker verification," in *Proceedings InterSpeech*, September 2007.

[15] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," *CoRR*, vol. abs/1612.07837, 2016. [Online]. Available: http://arxiv.org/abs/1612.07837

[16] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications," *CoRR*, vol. abs/1701.05517, 2017. [Online]. Available: http://arxiv.org/abs/1701.05517

[17] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions," *CoRR*, vol. abs/1712.05884, 2017. [Online]. Available: http://arxiv.org/abs/1712.05884

[18] J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio, "Char2wav: End-to-end speech synthesis," 2017.

[19] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw auaio," *CoRR*, 2016. [Online]. Available: https://arxiv.org/pdf/1609.03499.pdf

[20] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, "Parallel wavenet: Fast high-fidelity speech synthesis," *CoRR*, vol. abs/1711.10433, 2017. [Online]. Available: http://arxiv.org/abs/1711.10433

[21] E. Variani, X. Lei, E. McDermott, I. Lopez-Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4052–4056, 2014.

[22] M. Wan, G. Degottex, and M. J. F. Gales, "Integrated speaker-adaptive speech synthesis," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec 2017, pp. 705–711.

[23] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. V. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: A fully end-to-end text-to-speech synthesis model," *CoRR*, vol. abs/1703.10135, 2017. [Online]. Available: http://arxiv.org/abs/1703.10135

[24] Y. Wang, D. Stanton, Y. Zhang, R. J. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, F. Ren, Y. Jia, and R. A. Saurous, "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," *CoRR*, vol. abs/1803.09017, 2018. [Online]. Available: http://arxiv.org/abs/1803.09017

[25] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015. [Online]. Available: http://arxiv.org/abs/1511.07122