Natural Speech Technology (NST)

- EPSRC (UK Government) Programme Grant: collaboration

  ○ significantly advance state-of-the-art in speech technology
  ○ more natural, approaching human levels of reliability, adaptability and conversational richness
  ○ ran from 2011 to 2016 - interesting times ...
What is Deep Learning?

From Wikipedia:

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-linear transformations.
What is Deep Learning?

From Wikipedia:

*Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using multiple processing layers, with complex structures or otherwise, composed of multiple non-linear transformations.*

For NST:

*Deep learning allows both speech synthesis and speech recognition to use the same underlying, highly flexible, building blocks and adaptation techniques (and improved performance).*
The Rise of Deep Learning (June 2016)
Overview

• Basic Building Blocks
  ○ neural network architectures
  ○ activation functions

• Sequence-to-Sequence Modelling
  ○ generative models
  ○ discriminative models
  ○ encoder-decoder models

• Speech Processing Applications
  ○ language modelling
  ○ speech recognition
  ○ speech synthesis

• Adaptation for Deep Learning
What I am Not Discussing

- History of development
- Optimisation
  - essential aspect of all systems (major issue)
- Only work from NST
  - though (of course) the talk is biased
  - indicates papers/contributions from NST in area
- Experimental Results
  - see NST publications and other papers (references at end)

- My personal opinion of DNNs ...
Basic Building Blocks
Deep Neural Networks \[19\]

- General mapping process from input \(x_t\) to output \(y_t\)

\[y_t = \mathcal{F}(x_t)\]

- deep refers to number of hidden layers

- Depending on nature of outputs (\(y_t\)) vary activation function
  - \texttt{softmax} often for classification
  - \texttt{tanh} or \texttt{sigmoid} common for hidden layers
Neural Network Layer/Node

\[ \phi(\mathbf{w}_i \mathbf{z}_i) \]
• Activation functions:
  ○ step function (green)
  ○ sigmoid function (red)
  ○ tanh function (blue)

• softmax, usual output layer (sum-to-one/positive) for classification

\[ \phi_i(x_t) = \frac{\exp(w^T_i x_t + b_i)}{\sum_{j=1}^{J} \exp(w^T_j x_t + b_j)} \]
Activation Functions - ReLUs [35, 54]

• Alternative activation function: **Rectified Linear Units**

\[ y_i = \max(0, z_i) \]

• Related activation function **noisy ReLU**:

\[ y_i = \max(0, z_i + \epsilon); \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \]

- efficient, no exponential/division, rapid convergence in training

• Possible to train the activation function parameters
  - e.g. gradient of slopes for ReLUs
• Modify layer to model the residual

\[ y_t = F(x_t) + x_t \]

○ allows deeper networks to be built
○ deep residual learning

• Links to highway connections
Pooling/Max-Out Functions \cite{27, 43}

- Possible to pool the output of a set of node
  - reduces the number of weights to connect layers together

- A range of functions have been examined
  - maxout $\phi(y_1, y_2, y_3) = \max(y_1, y_2, y_3)$
  - soft-maxout $\phi(y_1, y_2, y_3) = \log(\sum_{i=1}^{3} \exp(y_i))$
  - p-norm $\phi(y_1, y_2, y_3) = \left(\sum_{i=1}^{3} |y_i|\right)^{1/p}$

- Has also been applied for unsupervised adaptation
Convolutional Neural Networks [26, 1]
Mixture Density Neural Networks \[4, 52\]

- Predict a mixture of experts
  - multiple components \(M\)
  - \(\mathcal{F}_m^{(c)}(x_t)\): prior prediction
  - \(\mathcal{F}_m^{(\mu)}(x_t)\): mean prediction
  - \(\mathcal{F}_m^{(\sigma)}(x_t)\): variance prediction
- Optimise using maximum likelihood

\[
p(y_t|x_t) = \sum_{m=1}^{M} \mathcal{F}_m^{(c)}(x_t) \mathcal{N}(y_t; \mathcal{F}_m^{(\mu)}(x_t), \mathcal{F}_m^{(\sigma)}(x_t))
\]

- Form of output influences output activation function used
Recurrent Neural Networks [38, 37]

- Introduce recurrent units
  \[ h_t = f^h (W^f_h x_t + W^r_h h_{t-1} + b_h) \]
  \[ y_t = f^f (W^y_h h_t + W^x x_t + b_y) \]
  - \( h_t \) history vector at time \( t \)

- Two history weight matrices
  - \( W^f_h \) forward
  - \( W^r_h \) recursion

- Uses (general) approximation (no optional i/o link)

\[
p(y_t|x_{1:t}) = p(y_t|x_t, x_{1:t-1}) \approx p(y_t|x_t, h_{t-1}) \approx p(y_t|h_t)
\]

- network has (causal) memory encoded in history vector \( (h_t) \)
RNN Variants [40, 10]

- **Bi-directional**: use complete observation sequence - non-causal
- **Variational**: introduce a latent variable sequence $z_{1:T}$

\[
p(y_t|x_{1:t}) \approx \int p(y_t|x_t, z_t, h_{t-1})p(z_t|h_{t-1})dz_t
\]
Network Gating

- A flexible extension to activation function is **gating**
  - standard form is \( \sigma() \) sigmoid activation function
  
  \[
  i = \sigma(W^f x_t + W^r h_{t-1} + b)
  \]
  - vector acts as a probabilistic gate on network values

- Gating can be applied at various levels
  - **features**: impact of input/output features on nodes
  - **time**: memory of the network
  - **layer**: influence of a layer’s activation function
Long-Short Term Memory Networks [20, 16]

\[ x_t, h_{t-1} \]
\[ \sigma \]
\[ f^m \]
\[ f^h \]
\[ c_t \]
\[ i_i \]
\[ i_o \]
\[ h_t \]

Time delay
The operations can be written as (peephole config):

- **Forget gate** ($i_f$), **Input gate** ($i_i$), **Output gate** ($i_o$)
  
  $$i_f = \sigma(W_f^fx_t + W_f^rh_{t-1} + W_m^fc_{t-1} + b_f)$$
  $$i_i = \sigma(W_i^fx_t + W_i^rh_{t-1} + W_i^mc_{t-1} + b_i)$$
  $$i_o = \sigma(W_o^fx_t + W_o^rh_{t-1} + W_o^mc_t + b_o)$$

- **Memory Cell**, **history vector** and gates are related by
  
  $$c_t = i_f \odot c_{t-1} + i_i \odot f^m(W_f^cx_t + W_r^rh_{t-1} + b_c)$$
  $$h_t = i_o \odot f^h(c_t)$$

- $\odot$ is **element-by-element** multiplication
- memory cell weight matrices ($W_f^m$, $W_i^m$, $W_o^m$) diagonal
- can allow explicit analysis of individual cell elements
Highway Connections [41]

- Gate the output of the node (example from LSTM)
  - combine with output from previous layer ($x_t$)
    
    $$i_h = \sigma(W^f h + W^r x + b)$$
    $$h_t = i_h \odot (i_o \odot f^n c_t) + (1 - i_h) \odot x_t$$
Sequence-to-Sequence Modelling
Sequence-to-Sequence Modelling

- Sequence-to-sequence modelling central to speech/language:
  - **speech synthesis**:
    - word sequence (discrete) $\rightarrow$ waveform (continuous)
  - **speech recognition**:
    - waveform (continuous) $\rightarrow$ word sequence (discrete)
  - **machine translation**:
    - word sequence (discrete) $\rightarrow$ word sequence (discrete)

- The sequence lengths on either side can differ
  - waveform sampled at 10ms/5ms frame-rate
  - word sequences (are words ...)

- Description focuses on ASR with RNNs (other models possible)
S2S: Generative Models [5, 6]

• Consider two sequences $L \leq T$:
  ◦ input: $x_{1:T} = \{x_1, x_2, \ldots, x_T\}$
  ◦ output: $y_{1:L} = \{y_1, y_2, \ldots, y_L\}$

• Consider generative model (ASR language)

\[
p(y_{1:L}, x_{1:T}) = p(y_{1:L}) p(x_{1:T} | y_{1:L}) \\
= p(y_{1:L}) \sum_{\phi_{1:T} \in \Phi} p(x_{1:T} | \phi_{1:T}) P(\phi_{1:T} | y_{1:L})
\]

  ◦ $p(y_{1:L})$: prior ("language") model
  ◦ $p(x_{1:T} | \phi_{1:T})$: (conditional) "acoustic" model
  ◦ $P(\phi_{1:T} | y_{1:L})$: alignment model - handles variable length
Prior Model Approximations [21, 3, 33]

- Markovian: N-gram and Feed-Forward neural network

\[
p(y_{1:L}) = \prod_{i=1}^{L} p(y_i | y_{1:i-1}) \approx \prod_{i=1}^{L} p(y_i | y_{i-1}, \ldots, y_{i-N+1})
\]

- Non-Markovian: Recurrent neural network

\[
p(y_{1:L}) \approx \prod_{i=1}^{L} p(y_i | y_{i-1}, \tilde{h}_{i-2}) \approx \prod_{i=1}^{L} p(y_i | \tilde{h}_{i-1})
\]

- depends on complete unobserved word history
Hidden Markov Models  [2, 13]

- Important sequence model is the hidden Markov model (HMM)
  - an example of a dynamic Bayesian network (DBN)

- discrete latent variables
  - $\phi_t$ describes discrete state-space
  - conditional independence assumptions
    
    $$P(\phi_t|\phi_{1:t-1}, y_{1:L}) = P(\phi_t|\phi_{t-1})$$
    
    $$p(x_t|x_{1:t-1}, \phi_{1:t}) = p(x_t|\phi_t)$$

- The likelihood of the data is

    
    $$p(x_{1:T}|y_{1:L}) = \sum_{\phi_{1:T} \in \Phi_{y_{1:L}}} \left( \prod_{t=1}^{T} p(x_t|\phi_t) P(\phi_t|\phi_{t-1}) \right)$$


Acoustic Model Approximations

- **Fully Markovian:** HMM, simplest form of approximation

  \[ p(x_{1:T} | \phi_{1:T}) \approx \prod_{t=1}^{T} p(x_t | \phi_t) \]

- **State Markovian:**

  \[ p(x_{1:T} | \phi_{1:T}) \approx \prod_{t=1}^{T} p(x_t | \phi_t, x_{1:t-1}) \approx \prod_{t=1}^{T} p(x_t | \phi_t, h_{t-1}) \]

- **Feature Markovian:**

  \[ p(x_{1:T} | \phi_{1:T}) \approx \prod_{t=1}^{T} p(x_t | \phi_{1:t}) \approx \prod_{t=1}^{T} p(x_t | \tilde{h}_t) \]
**Markovian Approximations and Inference**

Markovian (HMM)

\[ \prod_{t=1}^{T} p(x_t | \phi_t) \]

State Markovian

\[ \prod_{t=1}^{T} p(x_t | \phi_t, h_{t-1}) \]

Feature Markovian

\[ \prod_{t=1}^{T} p(x_t | \tilde{h}_t) \]

- Inference costs significantly different:
  - state Markovian: all past history observed - deterministic
  - feature Markovian: past history unobserved - depends on path
Deep learning can be used to estimate distributions - MDNN
  - more often trained as a **discriminative** model
  - need to convert to a “likelihood”

Most common form (for RNN acoustic model):

\[
p(x_t|\phi_t, h_{t-1}) \propto \frac{P(\phi_t|x_t, h_{t-1})}{P(\phi_t)}
\]

- \(P(\phi_t|x_t, h_{t-1})\): modelled by a standard RNN
- \(P(\phi_t)\): state/phone prior probability

**Why use a generative sequence-to-sequence model?**
S2S: Discriminative Models [5]

- Directly compute posterior of sequence

\[
p(y_{1:L}|x_{1:T}) = \sum_{\phi_{1:T} \in \Phi} p(y_{1:L}|\phi_{1:T}) P(\phi_{1:T}|x_{1:T})
\]

- State Markovian RNNs used to model history/alignment

\[
P(\phi_{1:T}|x_{1:T}) \approx \prod_{t=1}^{T} P(\phi_t|x_{1:t}) \approx \prod_{t=1}^{T} P(\phi_t|x_t, h_{t-1}) \approx \prod_{t=1}^{T} P(\phi_t|h_t)
\]

- Expression does not have alignment/language models
Connectionist Temporal Classification [15]

- CTC: discriminative model, no explicit alignment model
  - introduces a blank output symbol (\(\epsilon\))

Consider word: CAT
- Pronunciation: /C/ /A/ /T/

Observe 7 frames
- possible state transitions
- example path: /C/ \(\epsilon\) /A/ /A/ \(\epsilon\) /T/ \(\epsilon\)
Extension to Non-Markovian

- Interesting to consider state dependencies (right)

\[
P(\phi_{1:T}|x_{1:T}) \approx \prod_{t=1}^{T} P(\phi_t|x_{1:t}, \phi_{1:t-1}) \approx \prod_{t=1}^{T} P(\phi_t|\tilde{h}_t)
\]
Link of “Pain” - Non-Markovian

- Exact inference intractable
  - complete history dependence
  - see RNNLM ASR decoding
Discriminative Models and “Priors” [18]

- No language models in (this form of) discriminative model
  - in CTC the word history “captured” in state (frame) history
  - no explicit dependence on state (word) history
- Treat as a product of experts (log-linear model): for CTC

\[
p(y_{1:L} | x_{1:T}) = \frac{1}{Z(x_{1:T})} \exp \left( \alpha^T \left[ \log \left( \sum_{\phi_{1:T} \in \Phi} P(\phi_{1:T} | x_{1:T}) \right) \right] \right)
\]

- \(\alpha\) trainable parameter (related to LM scale)
- \(p(y_{1:L})\) standard “prior” (language) model
- Normalisation term not required in decoding
  - \(\alpha\) often empirically tuned
S2S: Encoder-Decoder Style Models

- Directly model relationship

\[
p(y_{1:L}|x_{1:T}) = \prod_{i=1}^{L} p(y_i|y_{1:i-1}, x_{1:T})
\]

\[
\approx \prod_{i=1}^{L} p(y_i|y_{i-1}, \tilde{h}_{i-2}, c)
\]

- Looks like an RNN LM with additional dependence on \( c \)

\[
c = \phi(x_{1:T})
\]

- \( c \) is a fixed length vector - like a sequence kernel
RNN Encoder-Decoder Model

- Simplest form is to use hidden unit from acoustic RNN/LSTM

\[ c = \phi(x_{1:T}) = h_T \]

- dependence on context is global via \( c \) - possibly limiting
Attention-Based Models [9, 7, 32]
Attention-Based Models

- Introduce attention layer to system
  - introduce dependence on locality $i$

\[
p(y_{1:L}|x_{1:T}) \approx \prod_{i=1}^{L} p(y_i|y_{i-1}, \tilde{h}_{i-2}, c_i) \approx \prod_{i=1}^{L} p(y_i|\tilde{h}_{i-1})
\]

\[
c_i = \sum_{\tau=1}^{T} \alpha_{i\tau} h_{\tau}; \quad \alpha_{i\tau} = \frac{\exp(e_{i\tau})}{\sum_{k=1}^{T} \exp(e_{ik})}, \quad e_{i\tau} = f^e(\tilde{h}_{i-2}, h_\tau)
\]

- $e_{i\tau}$ how well position $i-1$ in input matches position $\tau$ in output
- $h_\tau$ is representation (RNN) for the input at position $\tau$

- Attention can “wander” with large input size ($T$)
  - use a pyramidal network to reduce frame-rate for attention
S2S: Structured Discriminative Models [12, 31, 50]

- General form of structured discriminative model

\[
p(y_{1:L}|x_{1:T}) = \frac{1}{Z(x_{1:T})} \sum_{\phi_{1:T} \in \Phi \ y_{1:L}} \exp(\alpha^T f(x_{1:T}, \phi_{1:T}, y_{1:L}))
\]

- \( f(x_{1:T}, \phi_{1:T}, y_{1:L}) \) extracts features from observations/states/words
- need to map variable length sequence to a fixed length (again)
- latent variables, state sequence \( \phi_{1:T} \), “aid” attention

- Integrate with deep learning to model segment features:
  - RNN to map segments to a fixed length vector
  - segment posterior outputs from multiple systems (joint decoding)
Speech Processing
Applications
• Neural networks extensively used for language modelling
  ○ recurrent neural networks - complete word history

\[
P(\omega_{1:L}) = \prod_{i=1}^{L} P(\omega_i | \omega_{1:i-1}) \approx \prod_{i=1}^{L} P(\omega_i | \omega_{i-1}, \tilde{h}_{i-2}) \approx \prod_{i=1}^{L} P(\omega_i | \tilde{h}_{i-1})
\]

• 1-of-K ("one-hot") coding for \(i^{th}\) word, \(\omega_i, y_i\)
  ○ additional out-of-shortlist symbol may be added
  ○ softmax activation function on output layer

• Issues that need to be addressed
  1. \textbf{training}: how to efficiently train on billions of words?
  2. \textbf{decoding}: how to handle dependence on complete history?
LM: Cross-Entropy Training Criteria

- Standard training criterion for word sequence $\omega_{1:L} = \omega_1, \ldots, \omega_L$

$$\mathcal{F}_{ce} = -\frac{1}{L} \sum_{i=1}^{L} \log \left( P(\omega_i|\tilde{h}_{i-1}) \right)$$

- GPU training makes this reasonable BUT

- Compute cost for softmax normalisation term $Z(\tilde{h}_{i-1})$

$$P(\omega_i|\tilde{h}_{i-1}) = \frac{1}{Z(\tilde{h}_{i-1})} \exp \left( w^T_{f(\omega_i)} \tilde{h}_{i-1} \right)$$

- required as unobserved sequence (contrast acoustic model)
LM: Alternative Training Criteria [8]

- **Variance Regularisation**: eliminate normalisation from decoding

\[
\mathcal{F}_{vr} = \mathcal{F}_{ce} + \frac{\gamma}{2} \frac{1}{L} \sum_{i=1}^{L} \left( \log(Z(\tilde{h}_{i-1})) - \overline{\log(Z)} \right)^2
\]

- $\overline{\log(Z)}$ average (log) history normalisation
- all normalisation terms tend to be the same

- **Noise Contrastive Estimation**: efficient decoding and training

\[
\mathcal{F}_{nce} = -\frac{1}{L} \sum_{i=1}^{L} \left( \log(P(y_i = T|\omega_i, \tilde{h}_{i-1}) + \sum_{j=1}^{k} \log(P(y_i = F|\hat{\omega}_{ij}, \tilde{h}_{i-1})) \right)
\]

- $\hat{\omega}_{ij}$ are competing samples for $\omega_i$ - often sample from uni-gram LM
ASR decoding with RNNLMs has a link of “pain”
- History vector depends on “unobserved” word sequence
LM: ASR Decoding with RNNLMs

Consider word-lattice on the left
- expands to prefix tree (right) if complete history taken into account
- significant increase in number of paths
Prefix Tree

N-Gram Approximation

- Use exact RNN LM value but
  - merge paths based on N-gram history
  - can also use history vector distance merging
ASR: Sequence Training [24]

- Cross-Entropy using fixed alignment standard criterion (RNN)

\[ F_{ce} = - \sum_{t=1}^{T} \log \left( P(\hat{\phi}_t | x_t, h_{t-1}) \right) \]

  ○ criterion based on frame-by-frame classification

- Sequence training integrates sequence modelling into training

\[ F_{mbr} = \sum_{r=1}^{R} \sum_{\tilde{\omega}} P(\tilde{\omega}|x_{1:T}^{(r)}) \mathcal{L}(\tilde{\omega}, \omega^{(r)}) \]

  ○ MBR described (various loss functions) - also CML used
  ○ may be applied to generative and discriminative models
ASR: Sequence Training and CTC [36]

- Sequence-training is discriminative, so is CTC ...
  - let’s ignore the blank symbol
  - consider MMI as the training criterion
    \[ F_{\text{mmi}} = \sum_{r=1}^{R} \log \left( P(\omega^{(r)}|x^{(r)}_{1:T}) \right) \]
    - lattice-free training uses some of the CTC-style approaches

- CTC has local (every frame) normalisation
  - discriminatively-trained generative models use global normalisation

- CTC has no “language-model”
  - use phone level language model \( P(\text{ph}_i|\text{ph}_j) \) (a 4-gram used)
DNN acoustic models and Tandem systems
- uses bottleneck features and stacking
- fusion: based on log-linear models - structured SVM
TTS: Deep Learning for Synthesis  [53, 28]

For statistical speech synthesis model

\[
p(x_{1:T} | y_{1:L}) = \sum_{\phi_{1:T} \in \Phi_{y_{1:L}}} p(x_{1:T} | \phi_{1:T}) P(\phi_{1:T} | y_{1:L}) \approx p(x_{1:T} | \hat{\phi}_{1:T})
\]
TTS: Bottleneck Features

- Can also include bottleneck features (similar to ASR)
  - FE-DNN: standard feed-forward NN
  - MTE-BN-DNN: feed-forward NN with MTE and BN features

- Subjectively better as well
TTS: Minimum Trajectory Error [47]

• Smoothing is an important issue for generating trajectories in TTS
  ○ predict experts for static and dynamics parameters (MLPG)
  ○ use recursion on the output layer

• When using experts can minimise trajectory error (MTE)

\[ \mathcal{F}_{\text{mge}} = \sum_{r=1}^{R} (\hat{x}_{1:T}^{(r)} - x_{1:T}^{(r)})^T (\hat{x}_{1:T}^{(r)} - x_{1:T}^{(r)}) \]

\[ = \sum_{r=1}^{R} (R\hat{o}_{1:T}^{(r)} - x_{1:T}^{(r)})^T (R\hat{o}_{1:T}^{(r)} - x_{1:T}^{(r)}) \]

○ \( \hat{o}_{1:T}^{(r)} \) is the sequence of static/delta (means)
○ \( R \) is the mapping matrix from static/deltas to trajectory
• Make the system non-Markovian
• 1-of-K coding for samples
  ◦ sample-level synthesis
  ◦ 8-bit = 256 output
  ◦ softmax output activation
• Recurrent units (shown):
  ◦ 240ms=3840 samples
  ◦ insufficient history memory

• Replace recurrent units by a sparse Markovian history
  ◦ extensive use of dilation to limit model parameters

• Search not an issue - simply sampling!
• Markovian history modelling limited by parameter growth
  ○ network parameters grows linearly with the length of history
• Non-Markovian (recurrent) approaches address parameter issue
  ○ but hard to train, and long-term representation (often) poor

Because models with causal convolutions do not have recurrent connections, they are typically faster to train than RNNs, especially when applied to very long sequences. One of the problems of causal convolutions is that they require many layers, or large filters to increase the receptive field. For example, in Fig. 2 the receptive field is only 5 (= #layers + filter length - 1). In this paper we use dilated convolutions to increase the receptive field by orders of magnitude, without greatly increasing computational cost.

A dilated convolution (also called "a trous", or convolution with holes) is a convolution where the filter is applied over an area larger than its length by skipping input values with a certain step. It is equivalent to a convolution with a larger filter derived from the original filter by dilating it with zeros, but is significantly more efficient. A dilated convolution effectively allows the network to operate on a coarser scale than with a normal convolution. This is similar to pooling or strided convolutions, but here the output has the same size as the input. As a special case, dilated convolution with dilation 1 yields the standard convolution. Fig. 3 depicts dilated causal convolutions for dilations 1, 2, 4, and 8. Dilated convolutions have previously been used in various contexts, e.g. signal processing (Holschneider et al., 1989; Dutilleux, 1989), and image segmentation (Chen et al., 2015; Yu & Koltun, 2016).

Stacked dilated convolutions enable networks to have very large receptive fields with just a few layers, while preserving the input resolution throughout the network as well as computational efficiency. In this paper, the dilation is doubled for every layer up to a limit and then repeated: e.g. 1, 2, 4, ..., 512, 1, 2, 4, ..., 512.

The intuition behind this configuration is two-fold. First, exponentially increasing the dilation factor results in exponential receptive field growth with depth (Yu & Koltun, 2016). For example each 1, 2, 4, ..., 512 block has receptive field of size 1024, and can be seen as a more efficient and discriminative (non-linear) counterpart of a 1⇥1024 convolution. Second, stacking these blocks further increases the model capacity and the receptive field size.

2.2 SOFTMAX DISTRIBUTIONS

One approach to modeling the conditional distributions \( p(x_t | x_1, ..., x_{t-1}) \) over the individual audio samples would be to use a mixture model such as a mixture density network (Bishop, 1994) or mixture of conditional Gaussian scale mixtures (MCGSM) (Theis & Bethge, 2015). However, van den Oord et al. (2016a) showed that a softmax distribution tends to work better, even when the data is implicitly continuous (as is the case for image pixel intensities or audio sample values). One of the reasons is that a categorical distribution is more flexible and can more easily model arbitrary distributions because it makes no assumptions about their shape.

Because raw audio is typically stored as a sequence of 16-bit integer values (one per timestep), a softmax layer would need to output 65,536 probabilities per timestep to model all possible values. To make this more tractable, we first apply a \( \mu \)-law companding transformation (ITU-T, 1988) to the data, and then quantize it to 256 possible values:

\[
  f(x_t) = \text{sign}(x_t) \ln (1 + \mu |x_t|) - 3 \ln (1 + \mu),
\]

where \( \mu \) is the companding parameter.
Neural Network Adaptation
• Similar approaches used for TTS/ASR/LMs - broad classes
  ○ **auxiliary information**: i-vectors, gender information, emotion ]
  ○ **network adaptation**: weight adaptation, activation function adaptation
  ○ **feature transformation**: linear transformation of the features

• Possible to combine multiple approaches together
Auxiliary Information [34, 39, 22, 23, 8]

- Include information about speaker/topic/environment to network
  - often vector representation used - iVectors, LDA topic spaces
  - possible to adapt to speaker without hypothesis (e.g. iVectors)
- Applied to language modelling, speaker and noise adaptation
Seen a range of network adaptation approaches for ASR
- use “well-trained” network parameters as a prior
- updates all the parameters, weights, of the system

Is it possible to reduce number of parameters to be adapted?
• Structure network as a series of bases
  ◦ interpolation layer is speaker dependent

\[ h^{(l_s)} = \sum_{k=1}^{K} \lambda_k^{(s)} h_k^{(l)} \]

  ◦ few parameters per speaker - very rapid adaptation
  ◦ interpolation estimation convex optimisation problem
• Consider a layer of a network with $1000 \times 1000$ connections
  ○ weights: 1,000,000 parameters to adjust
  ○ activation functions: 2,000 functions (output and input)

• Take the example of a sigmoid activation function

$$\phi_i(\alpha_i^{(s)}, \alpha_o^{(s)}, \alpha_b^{(s)}) = \frac{\alpha_o^{(s)}}{1 + \exp\left(\alpha_i^{(s)} \mathbf{w}_i^T \mathbf{x}_t + \alpha_b^{(s)}\right)}$$

  ○ $\alpha_i^{(s)}$: scaling of the input
  ○ $\alpha_o^{(s)}$: scaling of the output
  ○ $\alpha_b^{(s)}$: offset on the activation
  ○ train these (or subset) parameters to be speaker specific

• Exact form of parameter adaptation depends on activation function
• Consider a speaker-specific linear transform of the weight matrix

\[ W^{(s)} = A^{(s)}W \]

○ act on a low-dimensional compression layer

• Compact transform central to aspects like covariance modelling:

\[ W^{(s)} = W + \sum_{i=1}^{P} \lambda_i^{(s)} A^{(i)}W \]

○ number of parameters is \( P \) - independent of weight matrix size
○ \( A^{(i)} \) low-rank, limits number of parameters
○ can make a function of auxiliary information (e.g. iVectors)
Feature Transformation (TTS) [11, 48]

- Train the network to predict experts for normalised features, $\tilde{y}_t$,
  - transform normalised experts to target speaker experts
- ASR constrained to use global transform
  - TTS can make use of regression class trees (CSMAPLR)
NST and Deep Learning

- Many other applications of deep learning under NST

  - **Speech Recognition**
    - audio data segmentation
    - beamforming and channel combination
    - network initialisation and regularisation
    - acoustic feature dependency modelling
    - paraphrastic language models

  - **Speech Synthesis**
    - post-processing
    - influence/limitations of LSTM parameters on synthesis
    - robust duration modelling
    - unit selection
Is Deep Learning the Solution?

• Most research still uses a two-stage approach to training:
  1. feature extraction: convert waveform to parametric form
  2. modelling: given parameters train model
• Limitations in the feature extraction stage cannot be overcome …
  ○ integrate feature extraction into process
  ○ attempt to directly model/synthesise waveform (WaveNet)
• Both are interesting, active, areas of research
  ○ links with “integrated end-to-end” systems: waveform-in words-out
  ○ feasible as quantity of data increases

• BUT
  ○ networks are difficult to optimise - tuning required
  ○ hard to interpret networks to get insights
  ○ sometimes difficult to learn from previous tasks …
Network Interpretation

- Deep learning usually highly distributed - hard to interpret
  - awkward to adapt/understand/regularise
  - modify training - add stimulation regularisation (improves ASR!)
Thank-you!


