Neural Networks for Acoustic Modelling 4: LSTM acoustic models; Sequence discriminative training

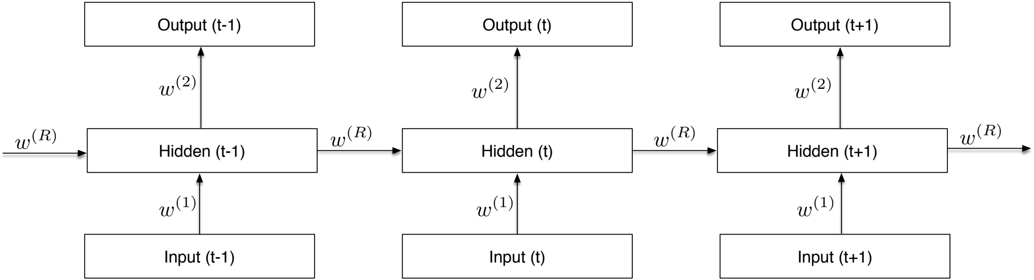
Steve Renals

Automatic Speech Recognition – ASR Lecture 10
14 February 2019

Modelling acoustic context

- DNNs allow the network to model acoustic context by including neighbouring frame in the input layer – the output is thus estimating the phone or state probability using that contextual information
- Richer NN models of acoustic context
  - **Time-delay neural networks (TDNNs)**
    - each layer processes a context window from the previous layer
    - higher hidden layers have a wider receptive field into the input
  - **Recurrent neural networks (RNNs)**
    - hidden units at time t take input from their value at time $t-1$
    - these recurrent connections allow the network to learn state
  - Both approaches try to learn invariances in time, and form representations based on compressing the history of observations

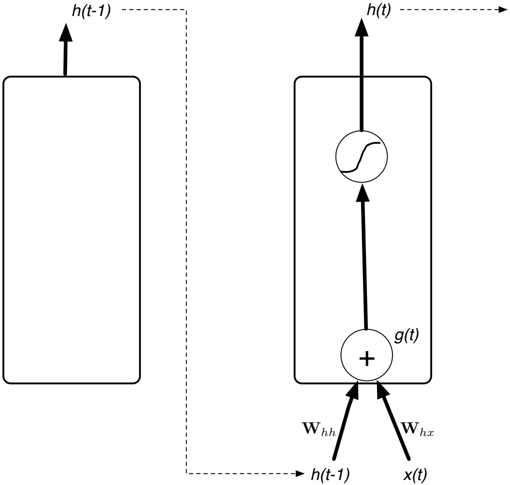
Recurrent network



- View an RNN for a sequence of T inputs as a T-layer network with shared weights
- Train by doing backprop through this unfolded network
- Recurrent hidden units are *state units*: can keep information through time
  - State units as memory – remember things for (potentially) an infinite time
  - State units as information compression – compress the history (sequence observed up until now) into a state representation

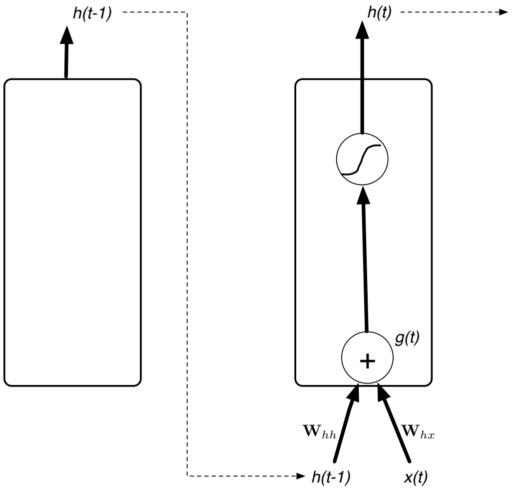
LSTM Recurrent Networks

Simple recurrent network unit

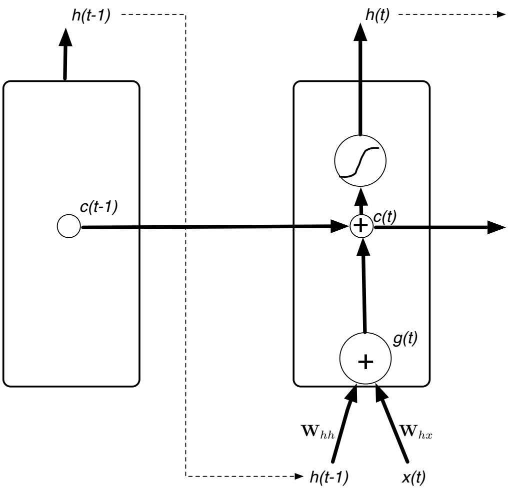


$$\boldsymbol{g}(t) = \boldsymbol{W}_{hx}\boldsymbol{x}(t) + \boldsymbol{W}_{hh}\boldsymbol{h}(t-1) + \boldsymbol{b}_h$$
$$\boldsymbol{h}(t) = \tanh\left(\boldsymbol{g}(t)\right)$$

LSTM

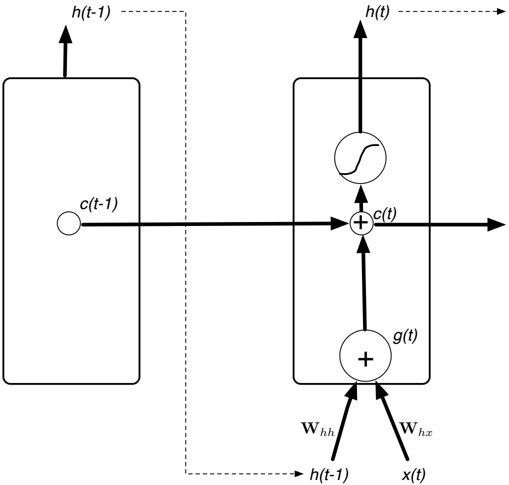


LSTM – Internal recurrent state



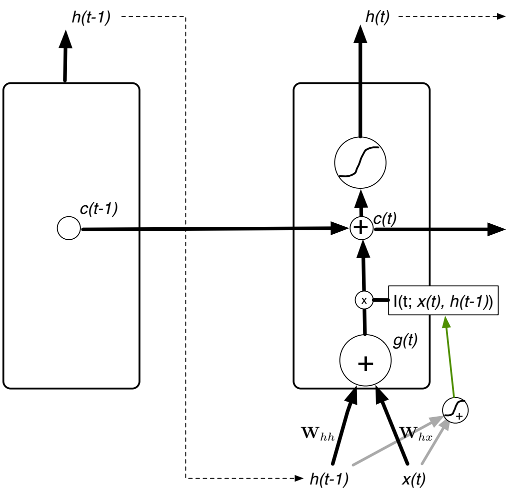
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$

LSTM – Internal recurrent state



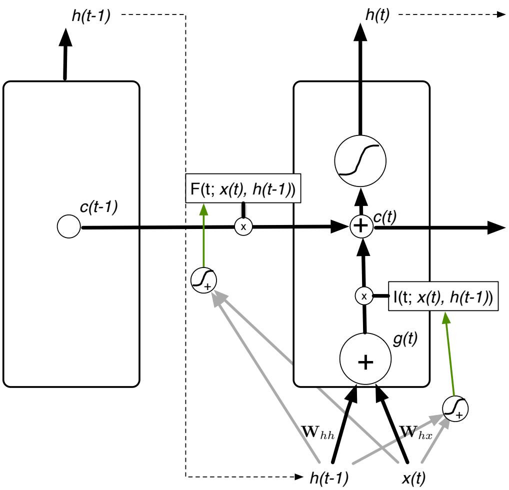
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state

LSTM – Input Gate



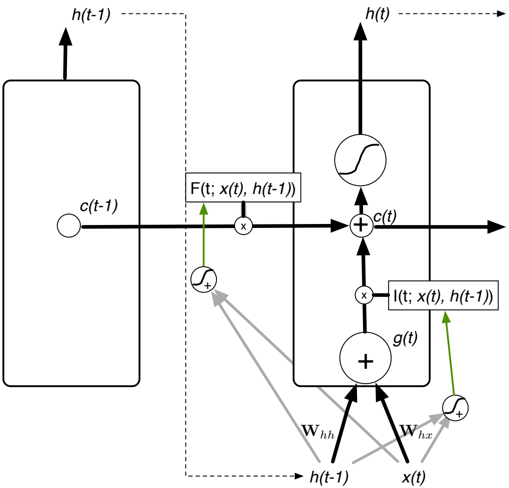
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state
- **Input gate**: controls how much input to the unit $g(t)$ is written to the internal state $c(t)$

LSTM – Forget Gate

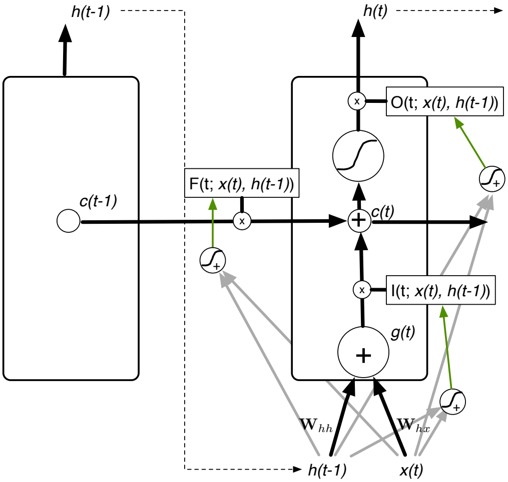


- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state
- **Input gate**: controls how much input to the unit $g(t)$ is written to the internal state $c(t)$
- **Forget gate**: controls how much of the previous internal state $c(t-1)$ is written to the internal state $c(t)$
  - Input and forget gates together allow the network to control what information is stored and overwritten at each step

LSTM – Input and Forget Gates

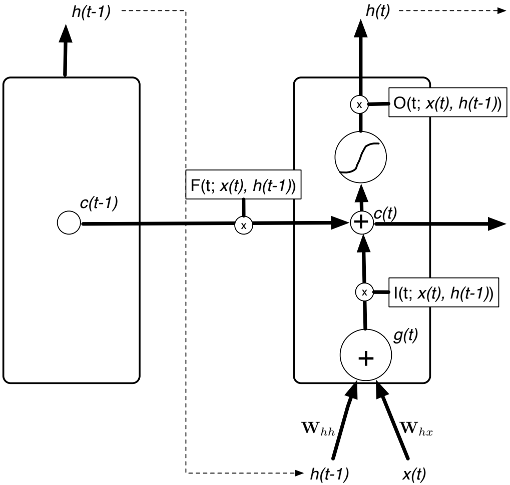


LSTM – Output Gate



- **Output gate**: controls how much of each unit's activation is output by the hidden state – it allows the LSTM cell to keep information that is not relevant at the current time, but may be relevant later

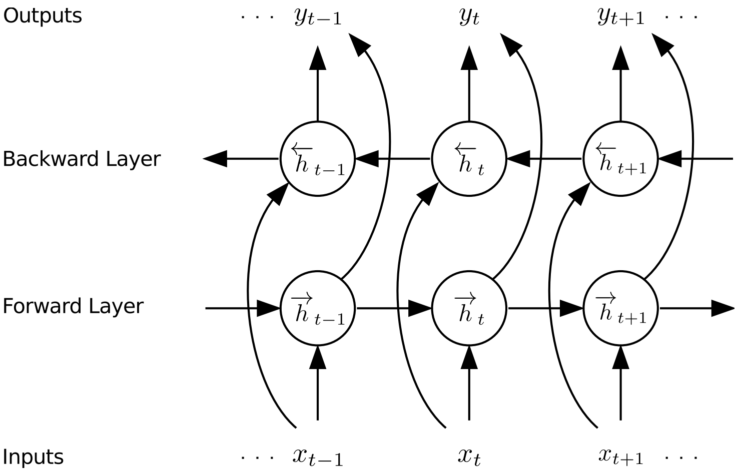
LSTM



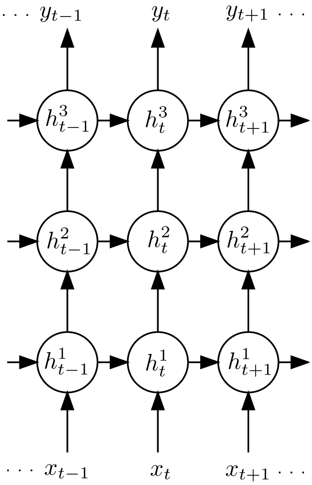
$$\boldsymbol{I}(t) = \sigma\left(\boldsymbol{W}_{ix}\boldsymbol{x}(t) + \boldsymbol{W}_{ih}\boldsymbol{h}(t-1) + \boldsymbol{b}_i\right)$$
$$\boldsymbol{F}(t) = \sigma\left(\boldsymbol{W}_{fx}\boldsymbol{x}(t) + \boldsymbol{W}_{fh}\boldsymbol{h}t-1) + \boldsymbol{b}_f\right)$$
$$\boldsymbol{O}(t) = \sigma\left(\boldsymbol{W}_{ox}\boldsymbol{x}(t) + \boldsymbol{W}_{oh}\boldsymbol{h}(t-1) + \boldsymbol{b}_o\right)$$
$$\boldsymbol{g}(t) = \boldsymbol{W}_{hx}\boldsymbol{x}(t) + \boldsymbol{W}_{hh}\boldsymbol{h}(t-1) + \boldsymbol{b}_h$$
$$\boldsymbol{c}(t) = \boldsymbol{F}(t) \circ \boldsymbol{c}(t-1) + \boldsymbol{I}(t) \circ \boldsymbol{g}(t)$$
$$\boldsymbol{h}(t) = \boldsymbol{O}(t) \circ \tanh\left(\boldsymbol{c}(t)\right)$$

C Olah (2015), Understanding LSTMs, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Bidirectional RNN

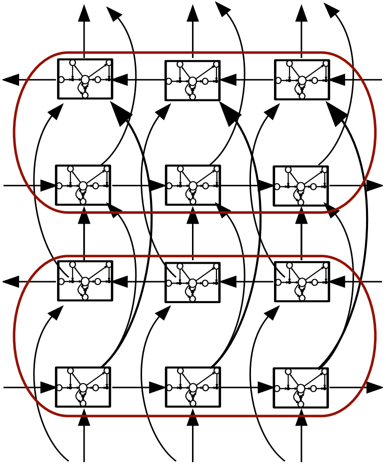


Deep RNN



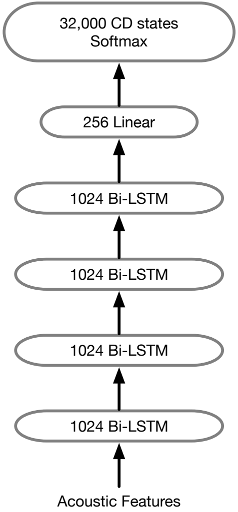
$\cdots\ y_{t-1}$
y_t
$y_{t+1}\ \cdots$
h^3_{t-1}
h^3_t
h^3_{t+1}
h^2_{t-1}
h^2_t
h^2_{t+1}
h^1_{t-1}
h^1_t
h^1_{t+1}
$\cdots\ x_{t-1}$
x_t
$x_{t+1}\ \cdots$

Deep Bidirectional LSTM



Example: Deep Bidirectional LSTM Acoustic Model (Switchboard)

- LSTM has 4-6 bidirectional layers with 1024 cells/layer (512 each direction)
- 256 unit linear bottleneck layer
- 32k context-dependent state outputs
- Input features
  - 40-dimension linearly transformed MFCCs (plus ivector)
  - 64-dimension log mel filter bank features (plus first and second derivatives)
  - concatenation of MFCC and FBANK features
- Training: 14 passes frame-level cross-entropy training, 1 pass sequence training (2 weeks on a K80 GPU)



Switchboard Results

Network Architecture	Test Set WER/% Switchboard	CallHome
GMM (ML)	21.2	36.4
GMM (BMMI)	18.6	33.0
DNN (7x2048) / CE	14.2	25.7
DNN (7x2048) / MMI	12.9	24.6
TDNN (6x1024) / CE	12.5	
TDNN (6x576) / LF-MMI	9.2	17.3
LSTM (4x1024)	8.0	14.3
LSTM (6x1024)	7.7	14.0
LSTM-6 + feat fusion	7.2	12.7

GMM and DNN results - Vesely et al (2013); TDNN-CE results - Peddinti et al (2015); TDNN/LF-MMI results - Povey et al (2016); LSTM results - Saon et al (2017)

Combining models, and with multiple RNN language models, WER reduced to 5.5/10.3% (Saon et al, 2017)

Sequence Discriminative Training

Recall: Maximum likelihood estimation of HMMs

- Maximum likelihood estimation (MLE) sets the parameters so as to maximize an objective function F_{MLE}:

$$F_{\text{MLE}} = \sum_{u=1}^{U} \log P_\lambda(\mathbf{X}_u \mid M(W_u))$$

for training utterances $\mathbf{X}_1 \ldots \mathbf{X}_U$ where W_u is the word sequence given by the transcription of the uth utterance, $M(W_u)$ is the corresponding HMM, and λ is the set of HMM parameters

Maximum mutual information estimation

- Maximum mutual information estimation (MMIE) aims to directly maximise the posterior probability (sometimes called conditional maximum likelihood). Using the same notation as before, with $P(w)$ representing the language model probability of word sequence w:

$$F_{\text{MMIE}} = \sum_{u=1}^{U} \log P_\lambda(M(W_u) \mid \mathbf{X}_u)$$
$$= \sum_{u=1}^{U} \log \frac{P_\lambda(\mathbf{X}_u \mid M(W_u))P(W_u)}{\sum_{w'} P_\lambda(\mathbf{X}_u \mid M(w'))P(w')}$$

Maximum mutual information estimation

- Maximum mutual information estimation (MMIE) aims to directly maximise the posterior probability (sometimes called conditional maximum likelihood). Using the same notation as before, with $P(w)$ representing the language model probability of word sequence w:

$$F_{\text{MMIE}} = \sum_{u=1}^{U} \log P_\lambda(M(W_u) \mid \mathbf{X}_u)$$

$$F_{\text{MLE}} = \sum_{u=1}^{U} \log \frac{P_\lambda(\mathbf{X}_u \mid M(W_u))P(W_u)}{\sum_{w'} P_\lambda(\mathbf{X}_u \mid M(w'))P(w')}$$

Maximum mutual information estimation

$$F_{\text{MMIE}} = \sum_{u=1}^{U} \log \frac{P_\lambda(\mathbf{X}_u \mid M(W_u))P(W_u)}{\sum_{w'} P_\lambda(\mathbf{X}_u \mid M(w'))P(w')}$$

Maximum mutual information estimation

$$F_{\text{MMIE}} = \sum_{u=1}^{U} \log \frac{P_\lambda(\mathbf{X}_u \mid M(W_u))P(W_u)}{\sum_{w'} P_\lambda(\mathbf{X}_u \mid M(w'))P(w')}$$

- **Numerator**: likelihood of data given correct word sequence ("clamped" to reference alignment)

Maximum mutual information estimation

$$F_{\text{MMIE}} = \sum_{u=1}^{U} \log \frac{P_\lambda(\mathbf{X}_u \mid M(W_u))P(W_u)}{\sum_{w'} P_\lambda(\mathbf{X}_u \mid M(w'))P(w')}$$

- **Numerator**: likelihood of data given correct word sequence ("clamped" to reference alignment)
- **Denominator**: total likelihood of the data given all possible word sequences – equivalent to summing over all possible word sequences estimated by the full acoustic and language models in recognition. ("free")

Maximum mutual information estimation

$$F_{\text{MMIE}} = \sum_{u=1}^{U} \log \frac{P_\lambda(\mathbf{X}_u \mid M(W_u))P(W_u)}{\sum_{w'} P_\lambda(\mathbf{X}_u \mid M(w'))P(w')}$$

- **Numerator**: likelihood of data given correct word sequence ("clamped" to reference alignment)
- **Denominator**: total likelihood of the data given all possible word sequences – equivalent to summing over all possible word sequences estimated by the full acoustic and language models in recognition. ("free")
- The objective function F_{MMIE} is optimised by making the correct word sequence likely (maximise the numerator), and all other word sequences unlikely (minimise the denominator)

Sequence training and lattices

- Computing the denominator involves summing over all possible word sequences – estimate by generating lattices, and summing over all words in the lattice
- In practice also compute numerator statistics using lattices (useful for summing multiple pronunciations)
- Generate numerator and denominator lattices for every training utterance
- Denominator lattice uses recognition setup (with a weaker language model)
- Each word in the lattice is decoded to give a phone segmentation, and forward-backward is then used to compute the state occupation probabilities
- Lattices not usually re-computed during training

MMIE is sequence discriminative training

- **Sequence:** like forward-backward (MLE) training, the overall objective function is at the sequence level – maximise the posterior probability of the word sequence given the acoustics $P_\lambda(M(W_u) \mid \mathbf{X}_u)$
- **Discriminative:** unlike forward-backward (MLE) training the overall objective function for MMIE is discriminative – to maximise MMI:
  - Maximise the numerator by increasing the likelihood of data given the correct word sequence
  - Minimise the denominator by decreasing the total likelihood of the data given all possible word sequences

  This results in "pushing up" the correct word sequence, while "pulling down" the rest

MPE: Minimum phone error

- Basic idea adjust the optimization criterion so it is directly related to word error rate
- Minimum phone error (MPE) criterion

MPE: Minimum phone error

- Basic idea adjust the optimization criterion so it is directly related to word error rate
- Minimum phone error (MPE) criterion

$$F_{\mathrm{MPE}} = \sum_{u=1}^{U} \log \frac{\sum_{W} P_{\lambda}(\mathbf{X}_u \mid M(W))P(W)A(W, W_u)}{\sum_{W'} P_{\lambda}(\mathbf{X}_u \mid M(W'))P(W')}$$

- $A(W, W_u)$ is the phone transcription accuracy of the sentence W given the reference W_u

MPE: Minimum phone error

- Basic idea adjust the optimization criterion so it is directly related to word error rate
- Minimum phone error (MPE) criterion

$$F_{\text{MMIE}} = \sum_{u=1}^{U} \log \frac{\sum_W P_\lambda(\mathbf{X}_u \mid M(W_{\underline{u}}))P(W_{\underline{u}})A(W, W_u)}{\sum_{W'} P_\lambda(\mathbf{X}_u \mid M(W'))P(W')}$$

- $A(W, W_u)$ is the phone transcription accuracy of the sentence W given the reference W_u

MPE: Minimum phone error

- Basic idea adjust the optimization criterion so it is directly related to word error rate
- Minimum phone error (MPE) criterion

$$F_{\mathrm{MPE}} = \sum_{u=1}^{U} \log \frac{\sum_{W} P_\lambda(\mathbf{X}_u \mid M(W)) P(W) A(W, W_u)}{\sum_{W'} P_\lambda(\mathbf{X}_u \mid M(W')) P(W')}$$

- $A(W, W_u)$ is the phone transcription accuracy of the sentence W given the reference W_u
- F_{MPE} is a weighted average over all possible sentences w of the raw phone accuracy
- Although MPE optimizes a phone accuracy level, it does so in the context of a word-level system: it is optimized by finding probable sentences with low phone error rates

HMM/DNN systems

- DNN-based systems are discriminative – the cross-entropy (CE) training criterion with softmax output layer "pushes up" the correct label, and "pulls down" competing labels
- CE is a frame-based criterion – we would like a sequence level training criterion for DNNs, operating at the word sequence level
- Can we train DNN systems with an MMI-type objective function?

HMM/DNN systems

- DNN-based systems are discriminative – the cross-entropy (CE) training criterion with softmax output layer "pushes up" the correct label, and "pulls down" competing labels
- CE is a frame-based criterion – we would like a sequence level training criterion for DNNs, operating at the word sequence level
- Can we train DNN systems with an MMI-type objective function? – **Yes**

Sequence training of hybrid HMM/DNN systems

- Forward- and back-propagation equations are structurally similar to forward and backward recursions in HMM training
- Initially train DNN framewise using cross-entropy (CE) error function
  - Use CE-trained model to generate alignments and lattices for sequence training
  - Use CE-trained weights to initialise weights for sequence training
- Train using back-propagation with sequence training objective function (e.g. MMI)

Sequence training results on Switchboard (Kaldi)

Results on Switchboard "Hub 5 '00" test set, trained on 300h training set, comparing maximum likelihood (ML) and discriminative (BMMI) trained GMMs with framewise cross-entropy (CE) and sequence trained (MMI) DNNs. GMM systems use speaker adaptive training (SAT).
All systems had 8859 tied triphone states.
GMMs – 200k Gaussians
DNNs – 6 hidden layers each with 2048 hidden units

	SWB	CHE	Total
GMM ML (+SAT)	21.2	36.4	28.8
GMM BMMI (+SAT)	18.6	33.0	25.8
DNN CE	14.2	25.7	20.0
DNN MMI	12.9	24.6	18.8

Veseley et al, 2013.

Summary

- LSTM recurrent networks and TDNNs offer different ways to model temporal context
- LSTM and/or TDNN systems are currently state-of-the-art
- Sequence training: discriminatively optimise GMM or DNN to a sentence (sequence) level criterion rather than a frame level criterion
  - ML training of HMM/GMM – sequence-level, not discriminative
  - CE training of HMM/NN – discriminative at the frame level
  - MMI training of HMM/GMM or HMM/NN – discriminative at the sequence level
- Usually initialise sequence discriminative training
  - HMM/GMM – first train using ML, followed by MMI
  - HMM/NN – first train at frame level (CE), followed by MMI

Reading

- Bidirectional LSTM acoustic model: Graves et al (2013), "Hybrid speech recognition with deep bidirectional LSTM", ASRU-2013. http://www.cs.toronto.edu/~graves/asru_2013.pdf
- IBM Switchboard system: Saon et al (2017), "English Conversational Telephone Speech Recognition by Humans and Machines", Interspeech-2107. https://arxiv.org/abs/1703.02136
- HMM discriminative training: Sec 27.3.1 of: S Young (2008), "HMMs and Related Speech Recognition Technologies", in *Springer Handbook of Speech Processing*, Benesty, Sondhi and Huang (eds), chapter 27, 539–557. http://www.inf.ed.ac.uk/teaching/courses/asr/2010-11/restrict/Young.pdf
- NN sequence training: K Vesely et al (2013), "Sequence-discriminative training of deep neural networks", Interspeech-2013, http://homepages.inf.ed.ac.uk/aghoshal/pubs/is13-dnn_seq.pdf
- Lattice-free MMI: D Povey et al (2016), "Purely sequence-trained neural networks for ASR based on lattice-free MMI", Interspeech-2016. http://www.danielpovey.com/files/2016_interspeech_mmi.pdf; slides – http://www.danielpovey.com/files/2016_interspeech_mmi_presentation.pptx (covered in lecture 12)