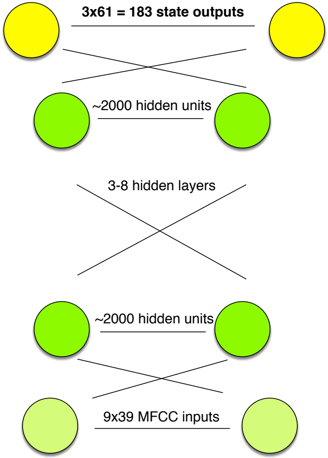
Neural Networks for Acoustic Modelling 3: Context-dependent DNNs, TDNNs and LSTMs

Peter Bell

Automatic Speech Recognition – ASR Lecture 12
25 February 2021

Modelling phonetic context

Recap: DNN for TIMIT



- **Deeper**: Deep neural network architecture – multiple hidden layers
- **Wider**: Use HMM state alignment as outputs rather than hand-labelled phones – 3-state HMMs, so 3×48 states
- Training many hidden layers is computationally expensive – use GPUs to provide the computational power

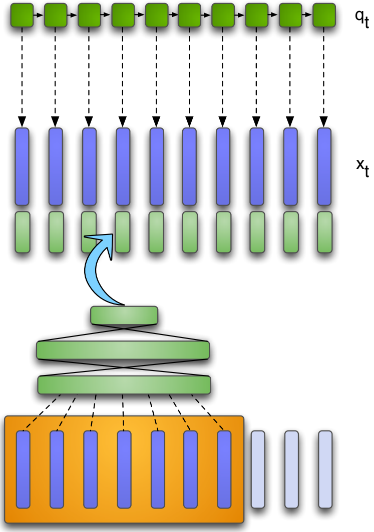
Recap: Cambridge GMM system

	CU-HTK 2000
Base model	HMM-GMM
Acoustic context	Δ, $\Delta\Delta$ features, HLDA projection
Phonetic context	Tied state triphones & quinphones
Speaker adaptation	Gender-dependent models, VTLN, MLLR
Training criterion	ML + MMI sequence training
System architecture	6-pass system
Other features	Multi-system combination
Hub 2000 WER	**19.3%**

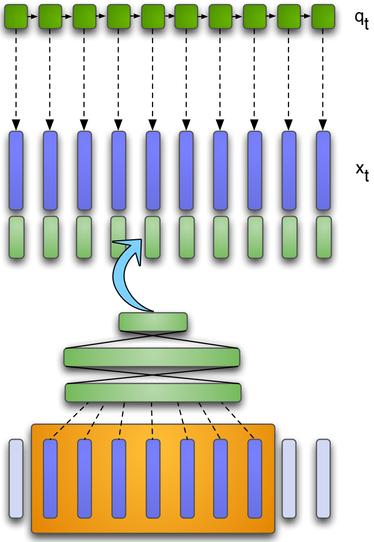
Tandem scheme

- Basic idea: use the output probabilities from the NN as input features to standard CD-HMM-GMM system
- Combines the benefits of both:
  - NNs good at modelling wide acoustic contexts, correlated input features
  - HMM-GMMs good for speaker adaptation, modelling phonetic context, sequence-training
- NN output probabilities are *Gaussianised* by taking logs and decorrelating with PCA
- Early variants used purely NN features; later variants augmented the feature vector with standard acoustic features
- Can also use "bottleneck features" (narrow, intermediate NN layers)

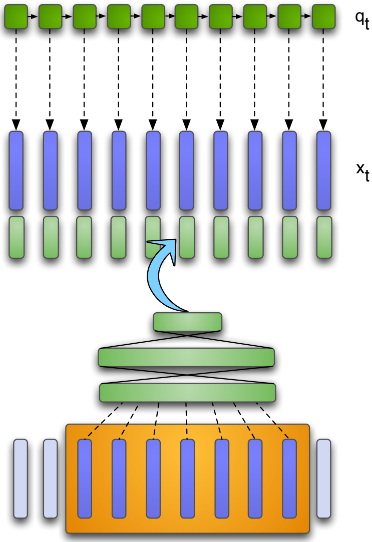
Tandem scheme



Tandem scheme



Tandem scheme



Modelling phonetic context with DNNs

- In the 1990s, this was considered hard (see Bourlard et al, 1992)
- But in 2011, a simple solution emerged: use state-tying from a GMM system

Modelling phonetic context with DNNs

- In the 1990s, this was considered hard (see Bourlard et al, 1992)
- But in 2011, a simple solution emerged: use state-tying from a GMM system

Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition

George E. Dahl, Dong Yu, *Senior Member, IEEE*, Li Deng, *Fellow, IEEE*, and Alex Acero, *Fellow, IEEE*

***Abstract*—We propose a novel context-dependent (CD) model for large-vocabulary speech recognition (LVSR) that leverages recent advances in using deep belief networks for phone recognition. We describe a pre-trained deep neural network hidden Markov model (DNN-HMM) hybrid architecture that trains the DNN to produce a distribution over senones (tied triphone states) as its output. The deep belief network pre-training algorithm is a robust and often helpful way to initialize deep neural networks generatively that**

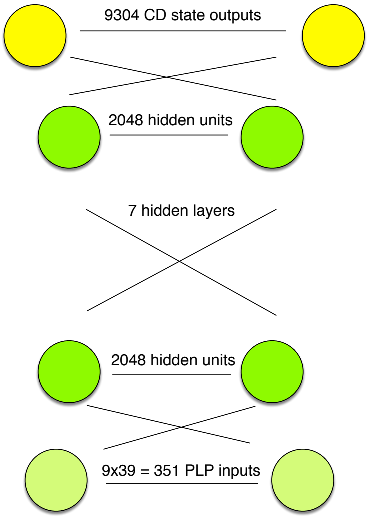
fields (CRFs) [18]–[20], hidden CRFs [21], [22], and segmental CRFs [23]). Despite these advances, the elusive goal of human level accuracy in real-world conditions requires continued, vibrant research.

Recently, a major advance has been made in training densely connected, directed belief nets with many hidden layers. The resulting deep belief nets learn a hierarchy of nonlinear feature

Context-dependent hybrid HMM/DNN

- First train a context-dependent HMM/GMM system on the same data, using a phonetic decision tree to determine the HMM tied states
- Perform Viterbi alignment using the trained HMM/GMM and the training data
- Train a neural network to map the input speech features to a label representing a context-dependent tied HMM state
  - So the size of the label set is thousands (number of context-dependent tied states) rather than tens (number of context-independent phones) Each frame is labelled with the Viterbi aligned tied state
- Train the neural network using gradient descent as usual
- Use the context-dependent scaled likelihoods obtained from the neural network when decoding

Example: HMM/DNN acoustic model for Switchboard



(Siede et al (2011))

Example: HMM/DNN acoustic model for Switchboard

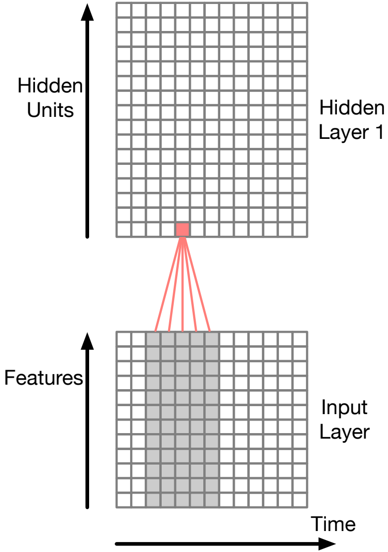
- Alignments generated from context-dependent HMM/GMM system
- Hybrid HMM/DNN system
  - Context-dependent — 9304 output units obtained from Viterbi alignment of HMM/GMM system
  - 7 hidden layers, 2048 units per layer
  - 11 frames of acoustic context
- DNN-based system results in significant word error rate reduction compared with GMM-based system
- Note: still no speaker adaptation or sequence-level training

Modelling acoustic context

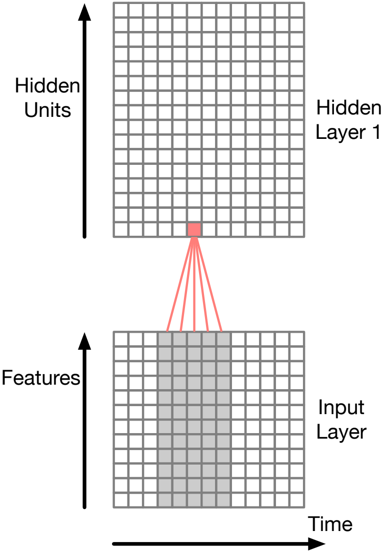
Modelling acoustic context

- 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

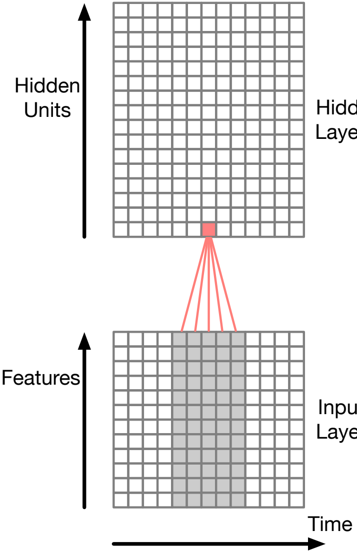
TDNNs – first hidden layer receptive field



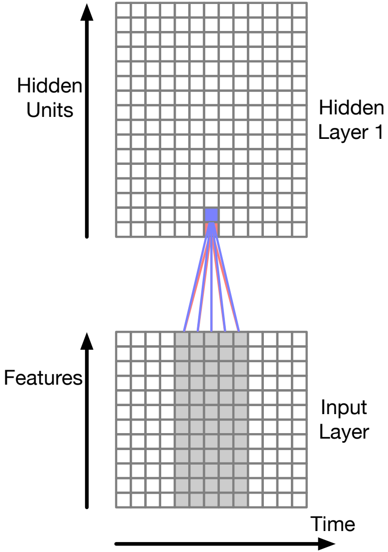
TDNNs – first hidden layer receptive field



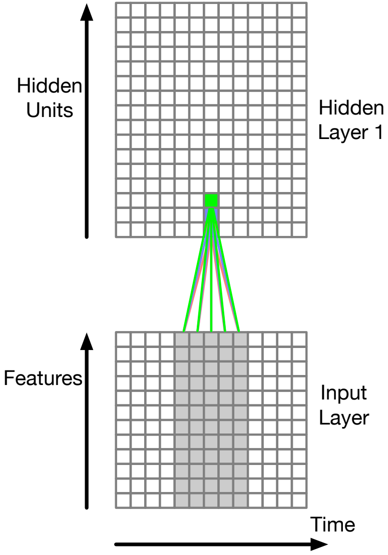
TDNNs – first hidden layer receptive field



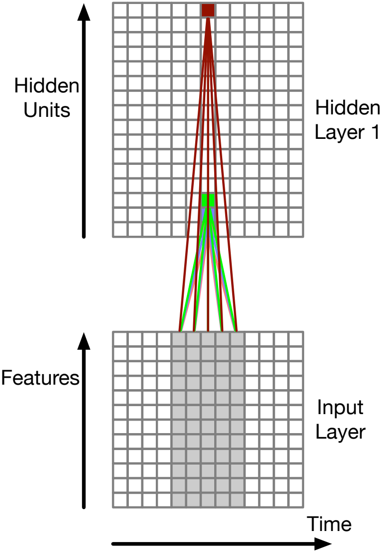
TDNNs – first hidden layer receptive field



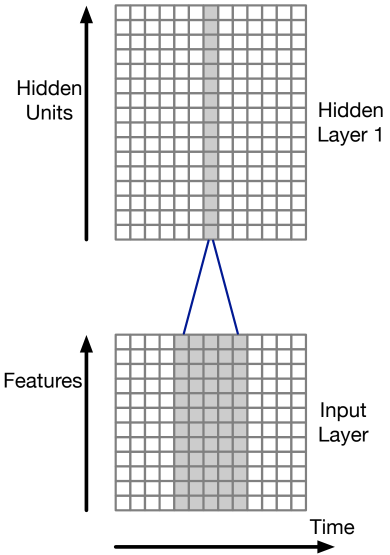
TDNNs – first hidden layer receptive field



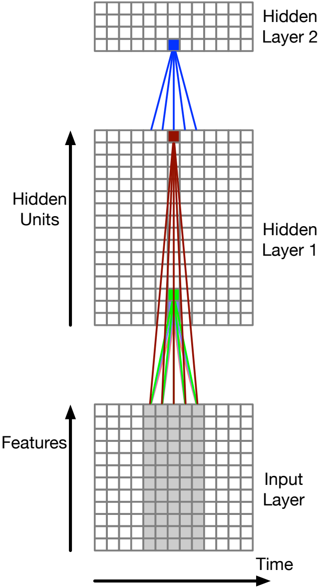
TDNNs – first hidden layer receptive field



TDNNs – first hidden layer receptive field

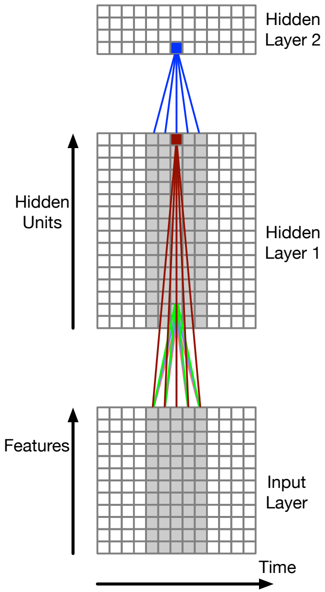


TDNNs – second hidden layer receptive field



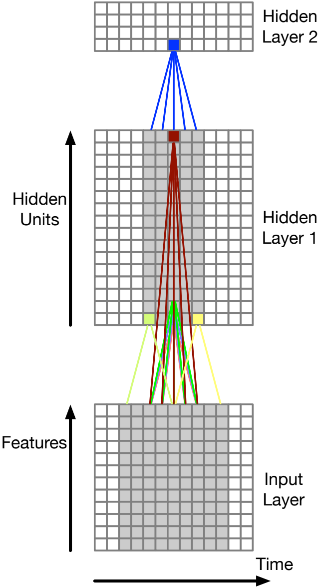
- Higher hidden layers take input from a time window over the previous hidden layer
- Lower hidden layers learn from narrower contexts, higher hidden layers from wider acoustic contexts
- Receptive field increases for higher hidden layers

TDNNs – second hidden layer receptive field



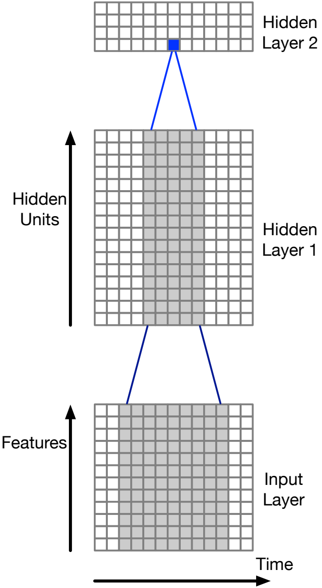
- Higher hidden layers take input from a time window over the previous hidden layer
- Lower hidden layers learn from narrower contexts, higher hidden layers from wider acoustic contexts
- Receptive field increases for higher hidden layers

TDNNs – second hidden layer receptive field



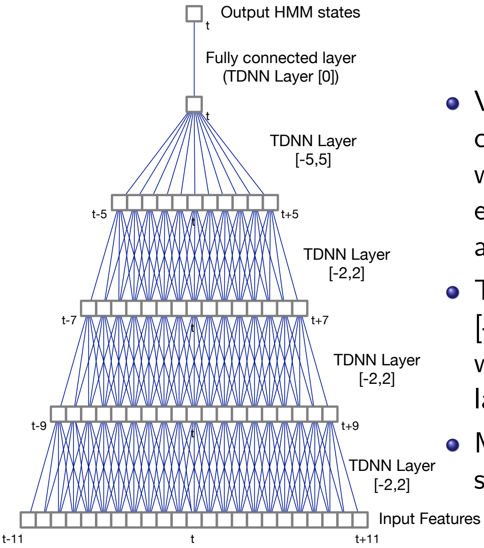
- Higher hidden layers take input from a time window over the previous hidden layer
- Lower hidden layers learn from narrower contexts, higher hidden layers from wider acoustic contexts
- Receptive field increases for higher hidden layers

TDNNs – second hidden layer receptive field



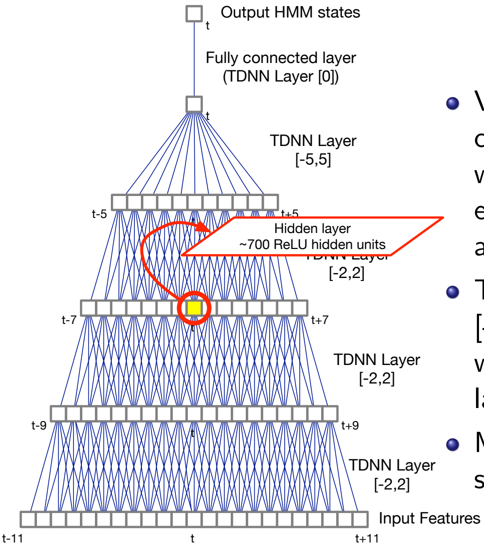
- Higher hidden layers take input from a time window over the previous hidden layer
- Lower hidden layers learn from narrower contexts, higher hidden layers from wider acoustic contexts
- Receptive field increases for higher hidden layers

Example TDNN Architecture



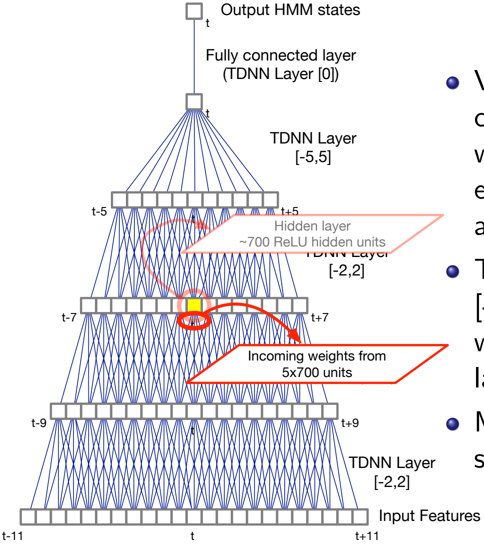
- View a TDNN as a 1D convolutional network with the transforms for each hidden unit tied across time
- TDNN layer with context [-2,2] has 5x as many weights as a regular DNN layer
- More computation, more storage required!

Example TDNN Architecture



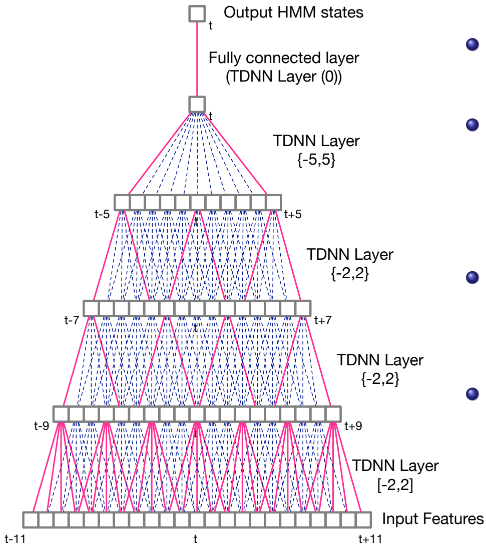
- View a TDNN as a 1D convolutional network with the transforms for each hidden unit tied across time
- TDNN layer with context [-2,2] has 5x as many weights as a regular DNN layer
- More computation, more storage required!

Example TDNN Architecture



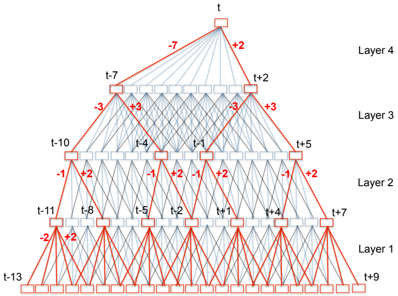
- View a TDNN as a 1D convolutional network with the transforms for each hidden unit tied across time
- TDNN layer with context [-2,2] has 5x as many weights as a regular DNN layer
- More computation, more storage required!

Sub-sampled TDNN



- Sub sample window of hidden unit activations
- Large overlaps between input contexts at adjacent time steps – likely to be correlated
- Allow gaps between frames in a window (cf. dilated convolutions)
- Sub-sampling saves computation and reduces number of model size (number of weights)

Example sub-sampled TDNN



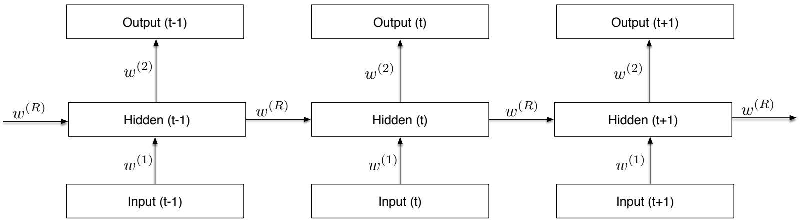
Peddinti (2015)

Layer	Context	Sub-sampled Context
1	[-2,2]	[-2,2]
2	[-1,2]	{-1,2}
3	[-3,3]	{-3,3}
4	[-7,2]	{-7,2}
5	{0}	{0}

- Increase the context for higher layers of the network
- Subsampled so that difference between sampled hidden units is multiple of 3 to enable "clean" sub-sampling
- Asymmetric contexts
- MFCC features in this case

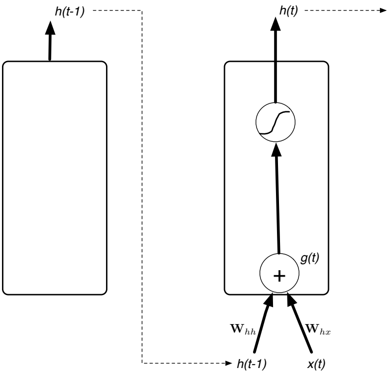
Recurrent Networks

Recurrent network



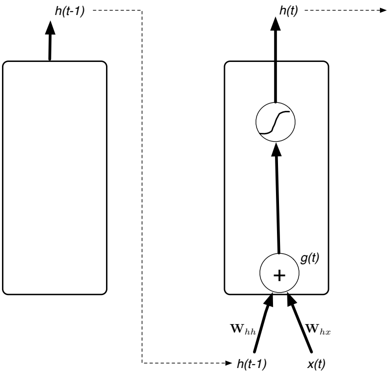
- View an RNN for a sequence of T inputs as a T-layer network with shared weights
- Train by doing backpropagation through this unfolded network
- Recurrent hidden units are *state units*: can keep information through time
  - State units as memory – remember things for (potentially) an infinite time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

Simple recurrent network unit

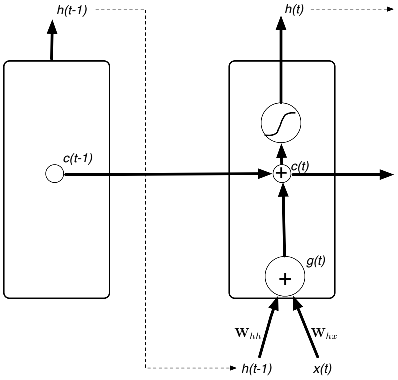


$$\boldsymbol{g}(t) = \boldsymbol{W}_{hx}\boldsymbol{x}(t) + \boldsymbol{W}_{hh}\boldsymbol{h}(t-1) + \boldsymbol{b}_h$$
$$\boldsymbol{h}(t) = \tanh(\boldsymbol{g}(t))$$

LSTM

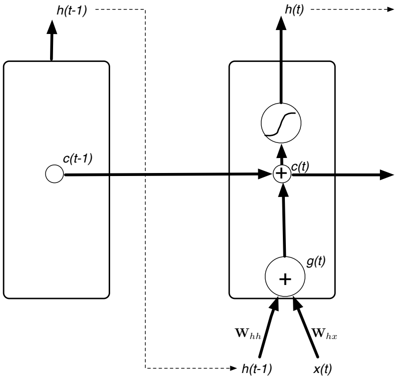


LSTM – Internal recurrent state



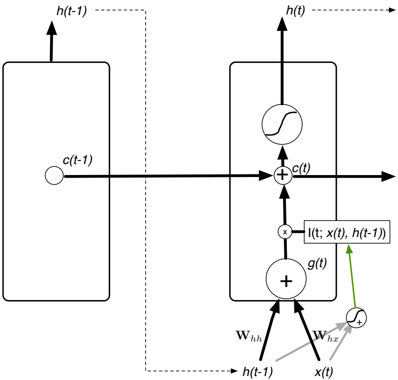
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$

LSTM – Internal recurrent state

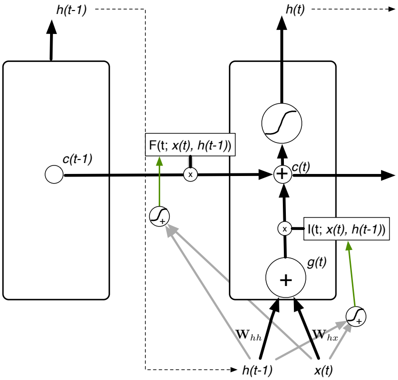


- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state

LSTM – Input Gate

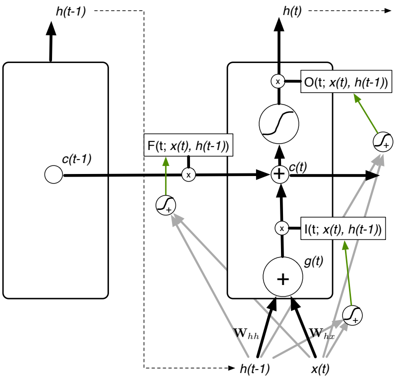


- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state
- **Input gate**: controls how much input to the unit $g(t)$ is written to the internal state $c(t)$



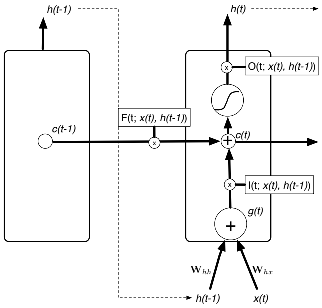
- **Internal recurrent state** ("cell") $c(t)$ combines previous state $c(t-1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state
- **Input gate**: controls how much input to the unit $g(t)$ is written to the internal state $c(t)$
- **Forget gate**: controls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

LSTM – Input, Forget and Output Gates



- **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

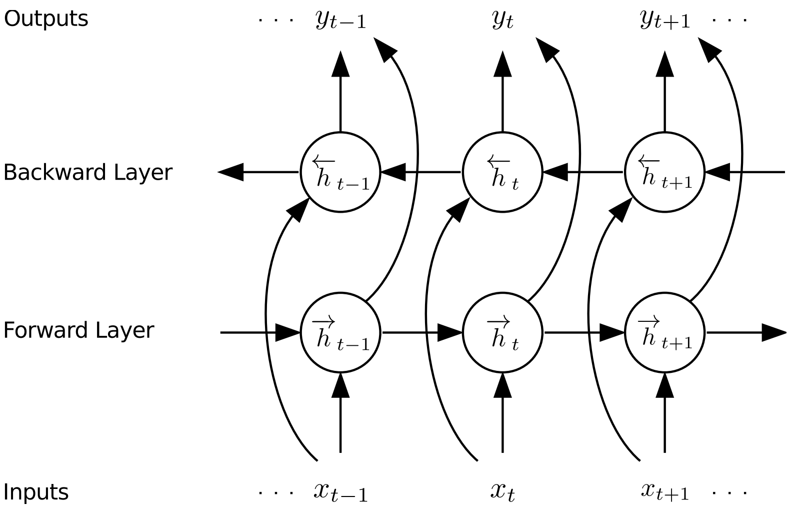


$$\boldsymbol{I}(t) = \sigma\left(\boldsymbol{W}_{ix}\boldsymbol{x}(t) + \boldsymbol{W}_{ih}\boldsymbol{h}(t-1) + \boldsymbol{b}_i\right)$$
$$\boldsymbol{F}(t) = \sigma\left(\boldsymbol{W}_{fx}\boldsymbol{x}(t) + \boldsymbol{W}_{fh}\boldsymbol{h}t-1) + \boldsymbol{b}_f\right)$$
$$\boldsymbol{O}(t) =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

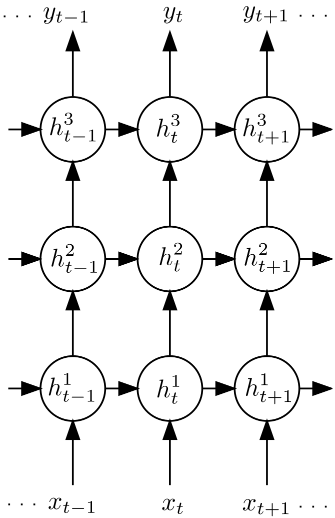
Aovids the vanishing gradient problem of conventional RNNs

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

Bidirectional RNN

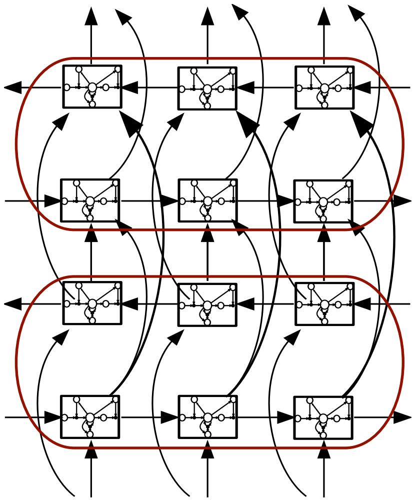


Deep RNN



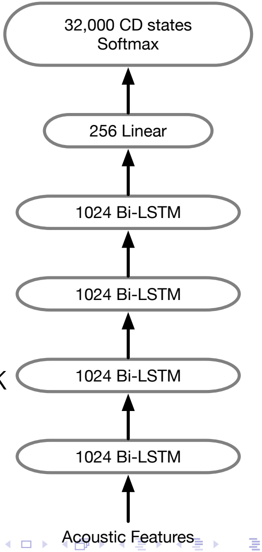
$\cdots\ y_{t-1}$
y_t
$y_{t+1}\ \cdots$
h^3_{t-1}
h^3_t
h^3_{t+1}
h^2_{t-1}
h^2_t
h^2_{t+1}
h^1_{t-1}
h^1_t
h^1_{t+1}
$\cdots\ x_{t-1}$
x_t
$x_{t+1}\ \cdots$

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 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)

Summary and Conclusions

- Scaling DNNs for large vocabulary speech recognition
- Context-dependent DNNs – use state clusters from CD HMM/GMM as output labels – results in significant improvements in accuracy for DNNs over GMMs
- LSTM recurrent networks and TDNNs offer different ways to model temporal context
- TDNN and/or LSTM systems are currently state-of-the-art

Reading

- A Maas et al (2017). "Building DNN acoustic models for large vocabulary speech recognition", *Computer Speech and Language*, **41**:195–213.
  https://arxiv.org/abs/1406.7806
- V Peddinti et al (2015). "A time delay neural network architecture for efficient modeling of long temporal contexts", *Interspeech*.
  https://www.isca-speech.org/archive/interspeech_2015/i15_3214.html

Background Reading:

- G Hinton et al (Nov 2012). "Deep neural networks for acoustic modeling in speech recognition", *IEEE Signal Processing Magazine*, **29**(6), 82–97.
  http://ieeexplore.ieee.org/document/6296526
- Hervé Bourlard (1992). "CDNN: A context-dependent neural network for speech recognition", *Proc. ICASSP*