

Neural Networks for Acoustic Modelling 2: DNN architectures

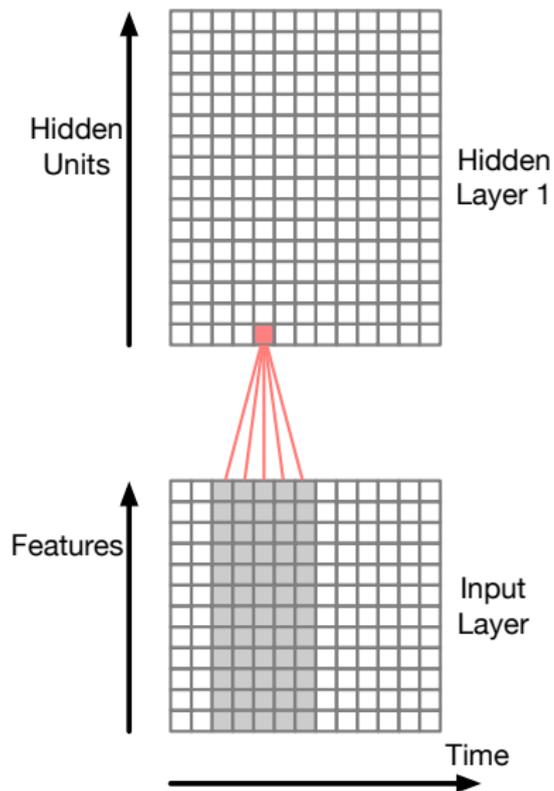
Peter Bell

Automatic Speech Recognition – ASR Lecture 11
23 February 2026

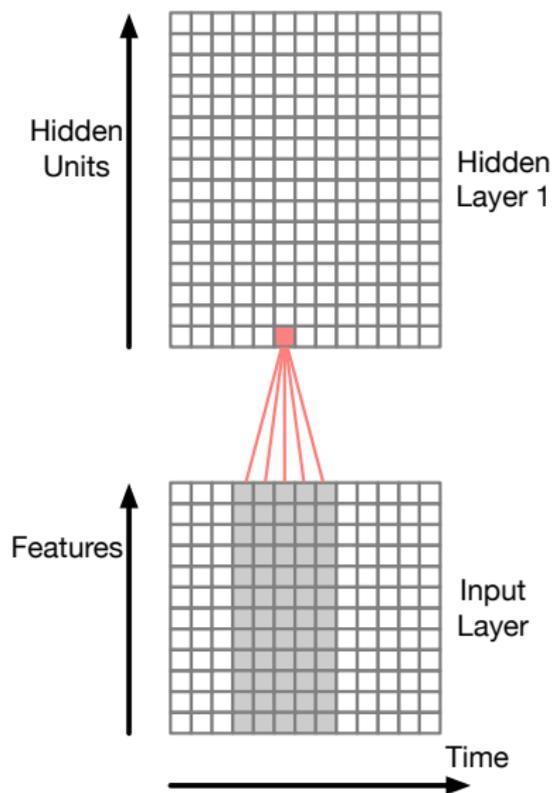
Modelling acoustic context

- 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
- We'll also briefly mention CNNs and Transformers

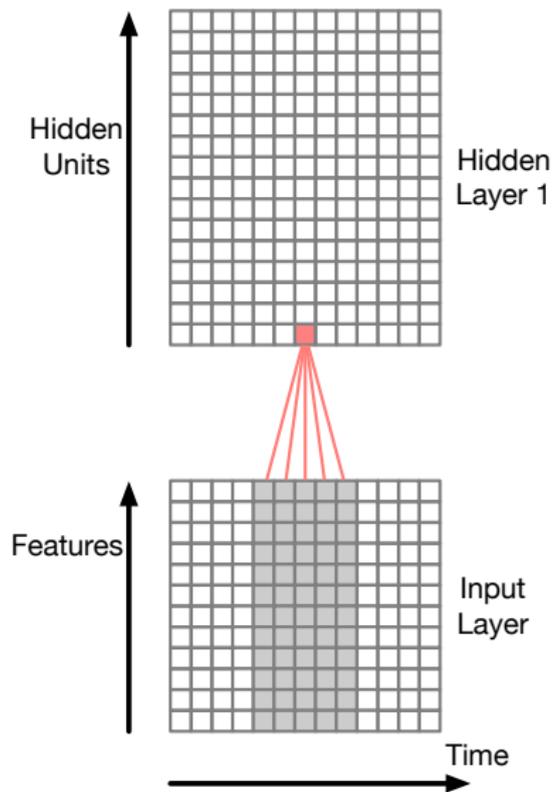
TDNNs – first hidden layer receptive field



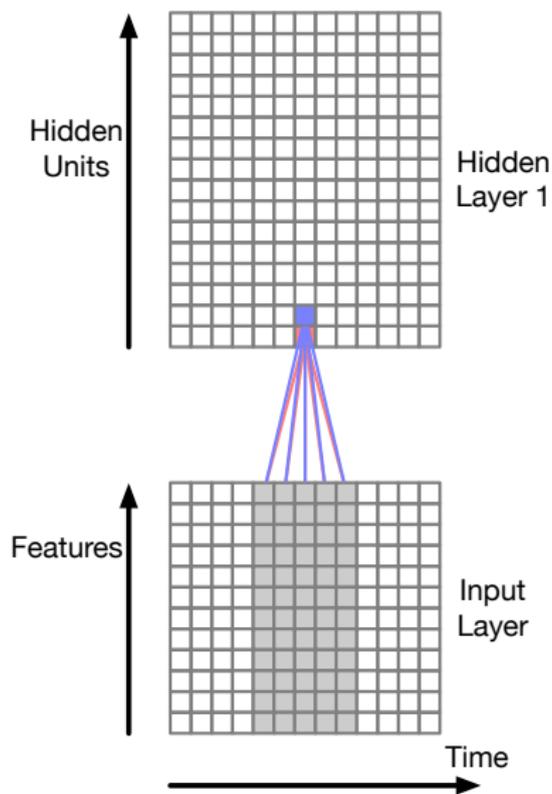
TDNNs – first hidden layer receptive field



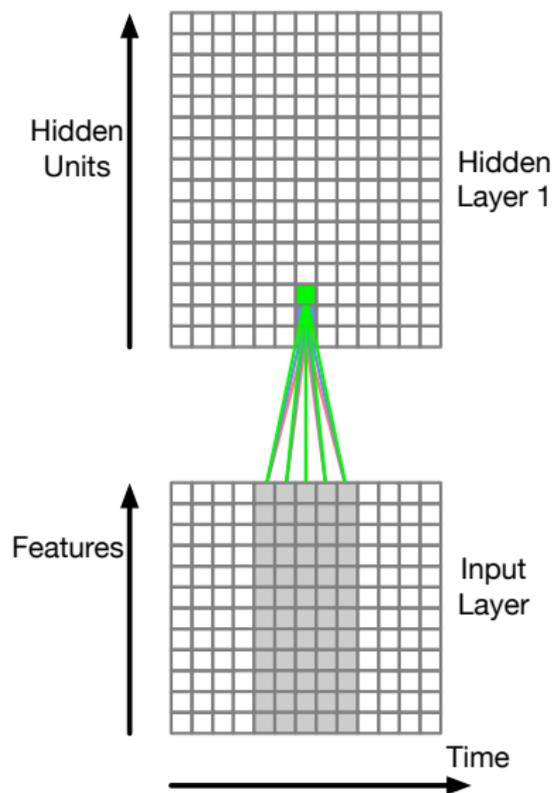
TDNNs – first hidden layer receptive field



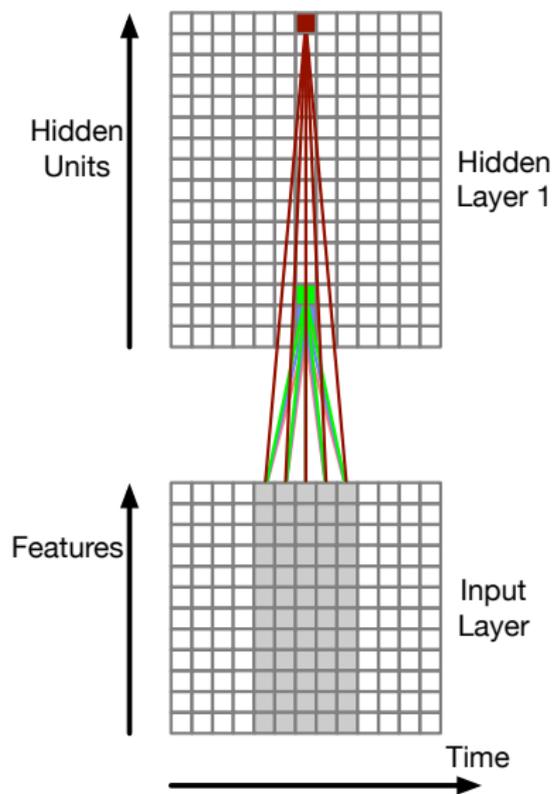
TDNNs – first hidden layer receptive field



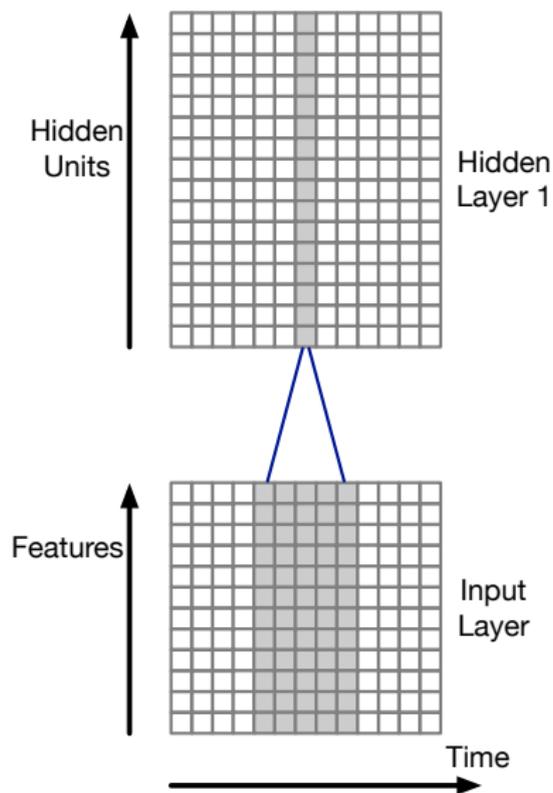
TDNNs – first hidden layer receptive field



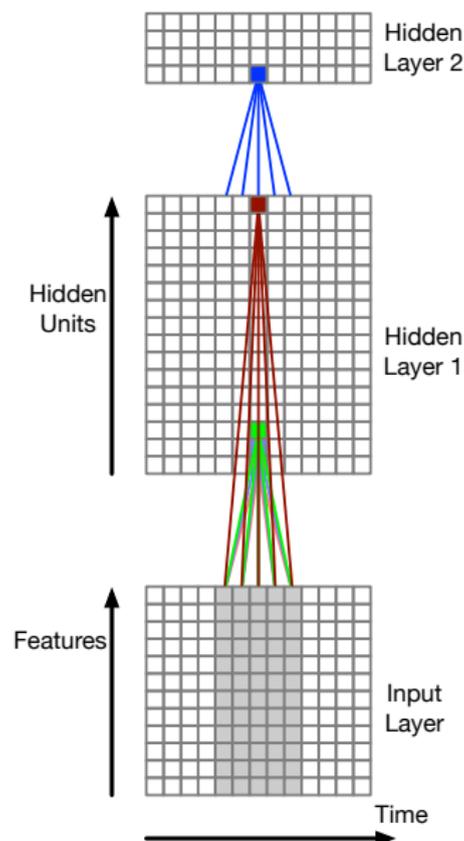
TDNNs – first hidden layer receptive field



TDNNs – first hidden layer receptive field

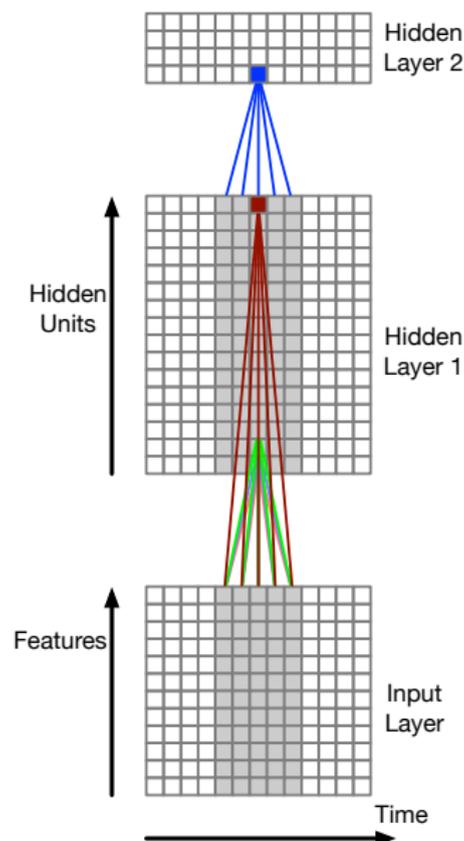


TDNNs – second hidden layer receptive field



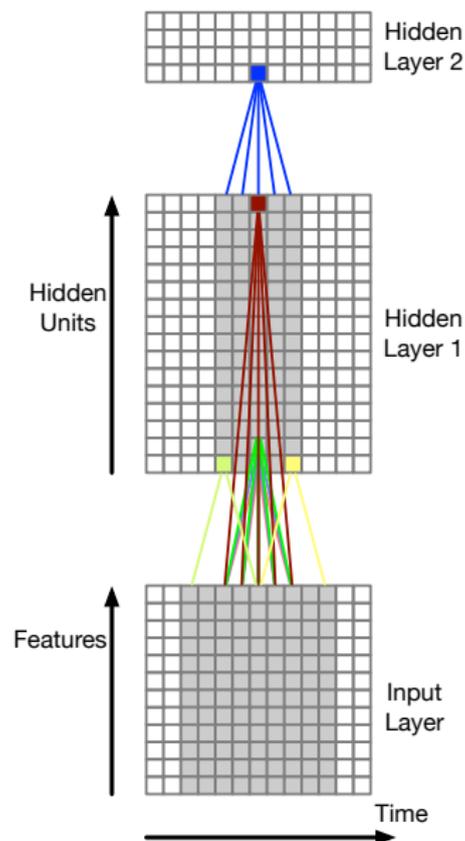
- Higher hidden layers take input from a time window over the previous hidden layer
- Lower hidden layers learn from narrower contexts, higher hidden layers from wider acoustic contexts
- Receptive field increases for higher hidden layers

TDNNs – second hidden layer receptive field



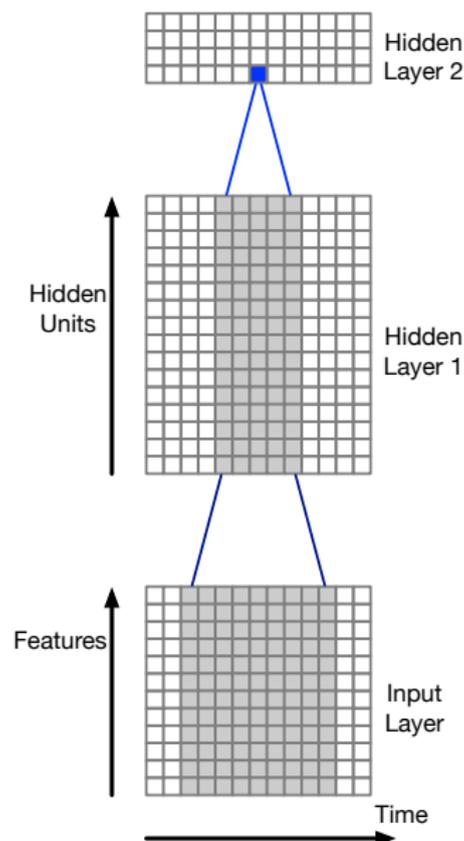
- Higher hidden layers take input from a time window over the previous hidden layer
- Lower hidden layers learn from narrower contexts, higher hidden layers from wider acoustic contexts
- Receptive field increases for higher hidden layers

TDNNs – second hidden layer receptive field



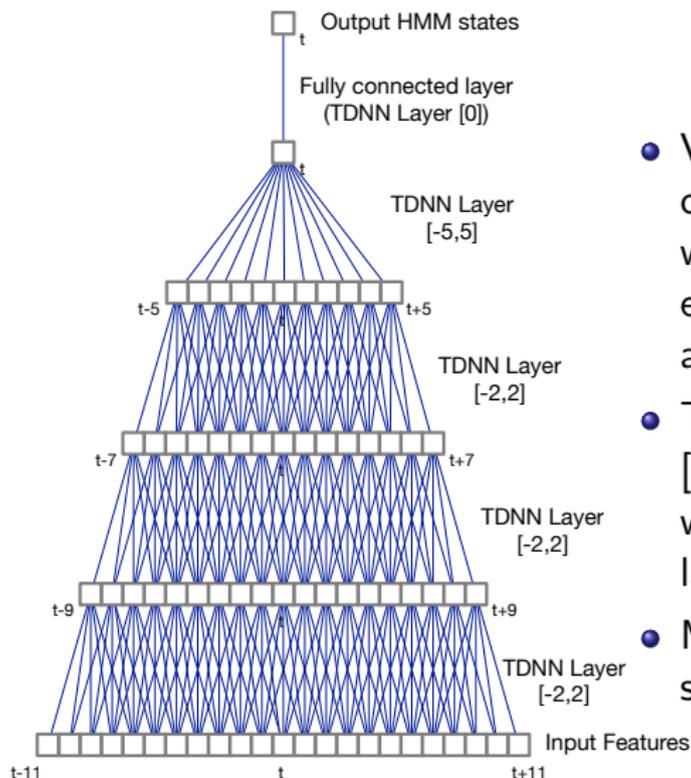
- Higher hidden layers take input from a time window over the previous hidden layer
- Lower hidden layers learn from narrower contexts, higher hidden layers from wider acoustic contexts
- Receptive field increases for higher hidden layers

TDNNs – second hidden layer receptive field



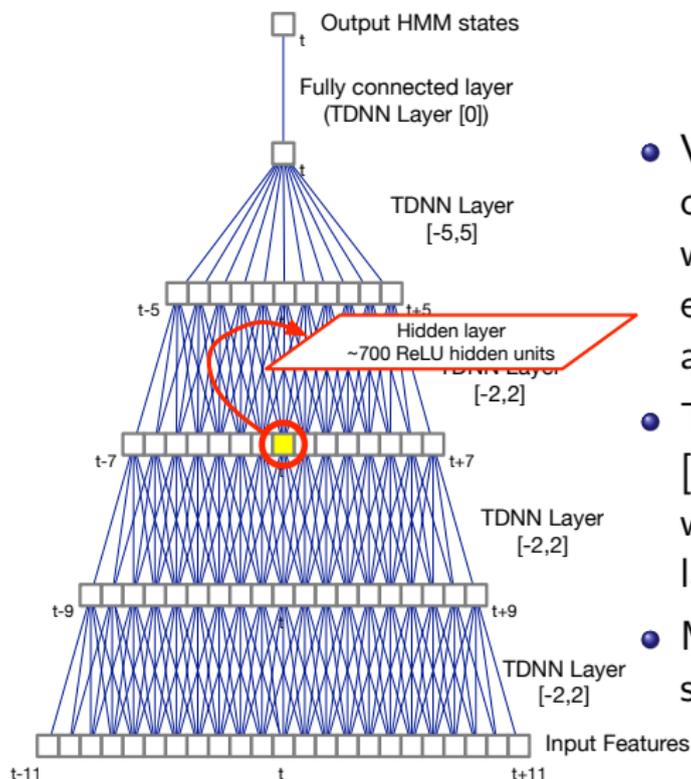
- Higher hidden layers take input from a time window over the previous hidden layer
- Lower hidden layers learn from narrower contexts, higher hidden layers from wider acoustic contexts
- Receptive field increases for higher hidden layers

Example TDNN Architecture



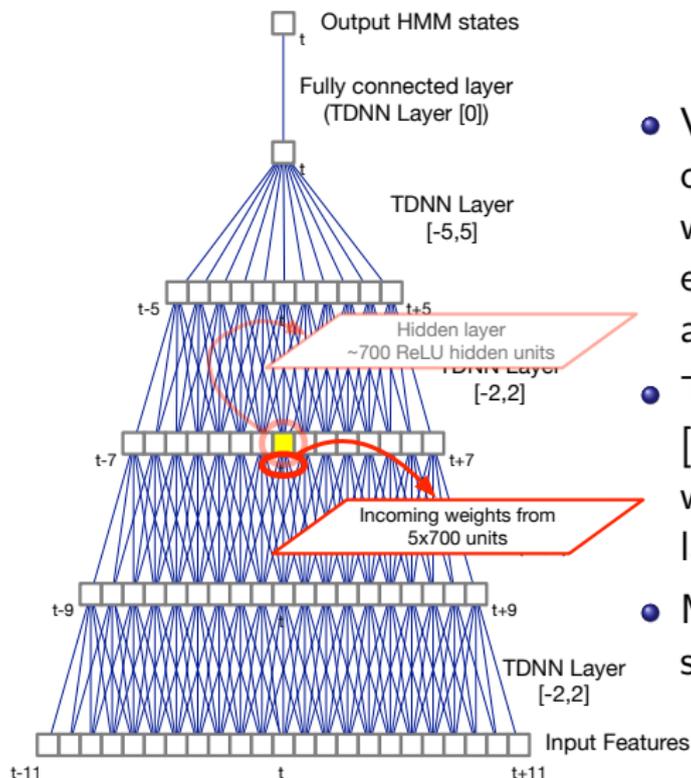
- View a TDNN as a 1D convolutional network with the transforms for each hidden unit tied across time
- TDNN layer with context $[-2,2]$ has 5x as many weights as a regular DNN layer
- More computation, more storage required!

Example TDNN Architecture



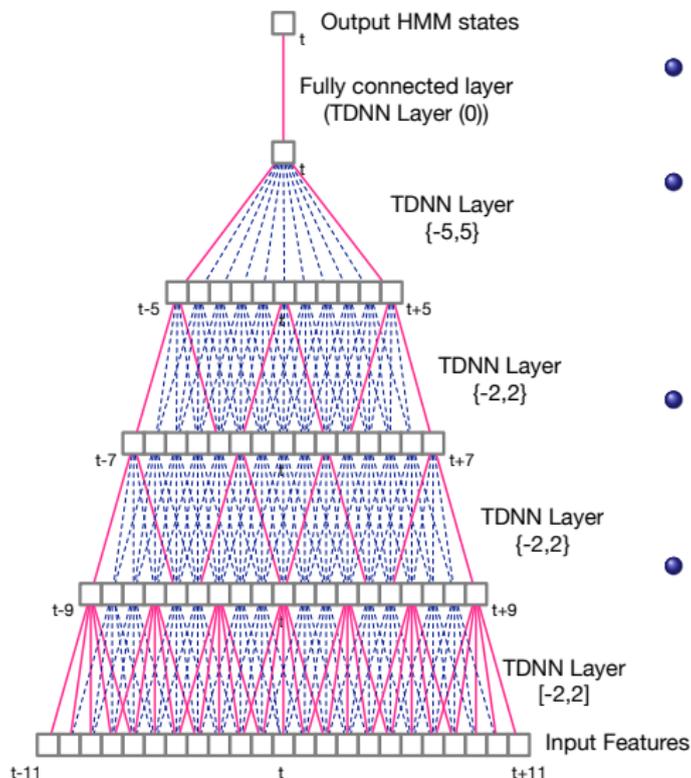
- View a TDNN as a 1D convolutional network with the transforms for each hidden unit tied across time
- TDNN layer with context $[-2,2]$ has 5x as many weights as a regular DNN layer
- More computation, more storage required!

Example TDNN Architecture



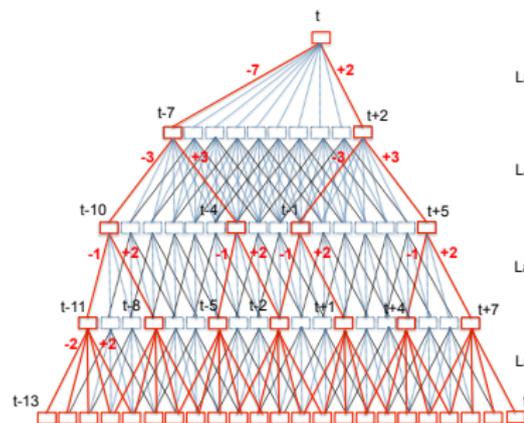
- View a TDNN as a 1D convolutional network with the transforms for each hidden unit tied across time
- TDNN layer with context $[-2,2]$ has 5x as many weights as a regular DNN layer
- More computation, more storage required!

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

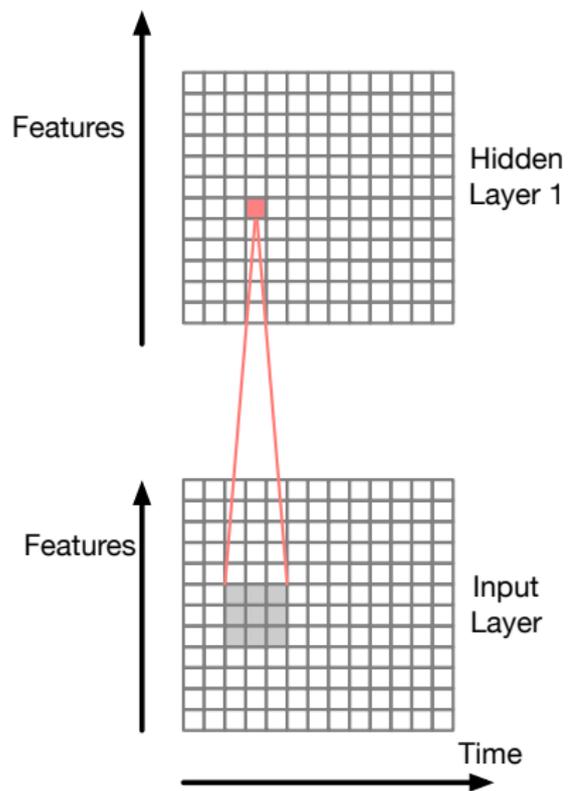


Layer 4 Peddinti (2015)

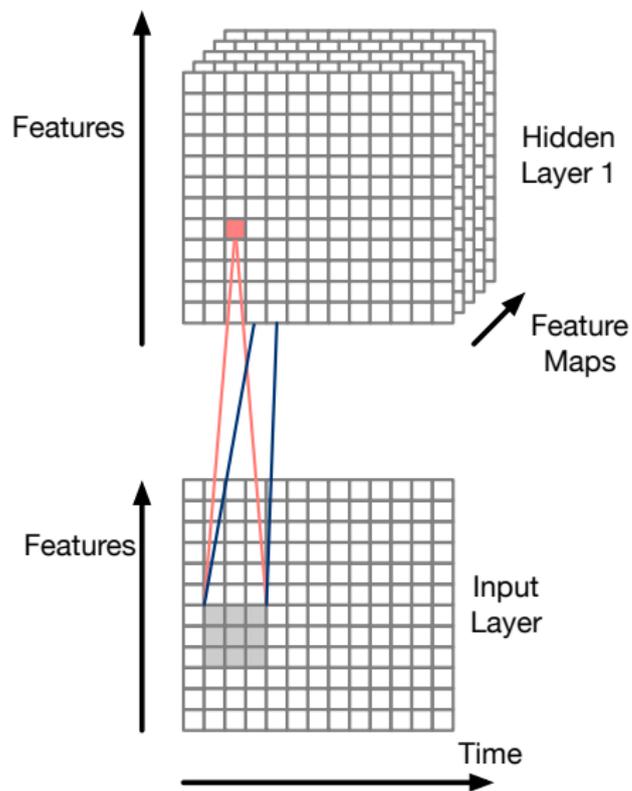
Layer	Context	Sub-sampled Context
1	$[-2, 2]$	$[-2, 2]$
2	$[-1, 2]$	$\{-1, 2\}$
3	$[-3, 3]$	$\{-3, 3\}$
4	$[-7, 2]$	$\{-7, 2\}$
5	$\{0\}$	$\{0\}$

- 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

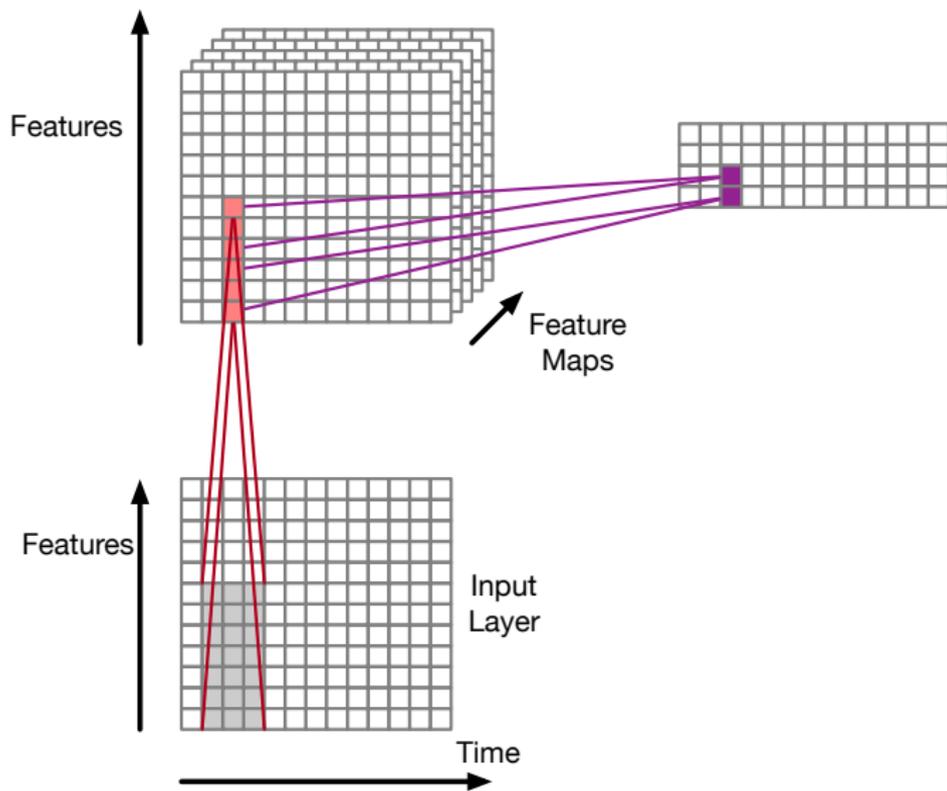
Convolutional networks



Convolutional networks



Convolutional networks



Example CNN architectures

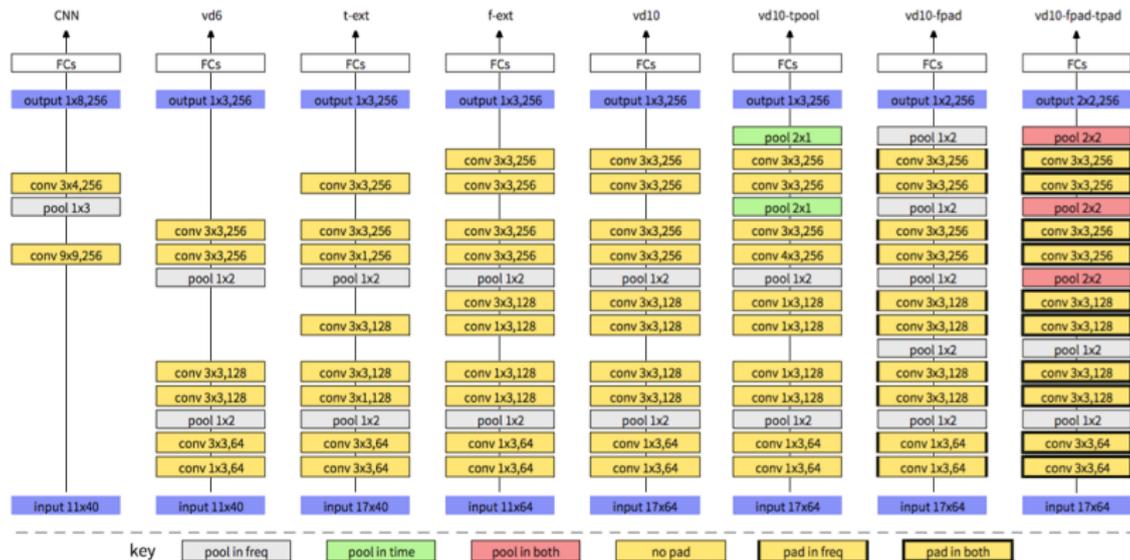
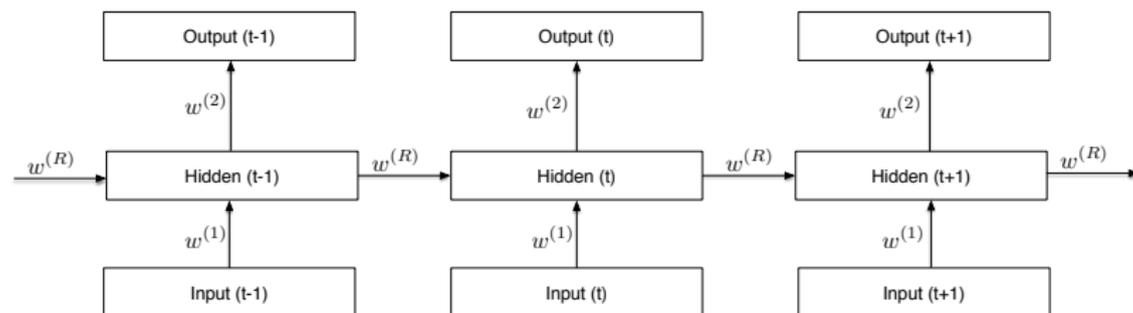


Figure from Qian & Woodland (2016)

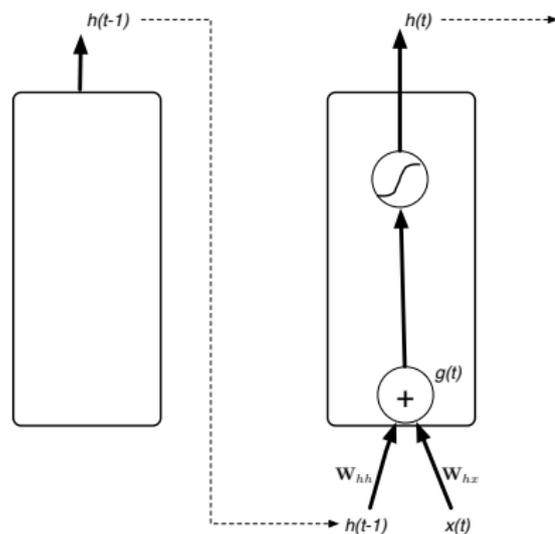
Recurrent Networks

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

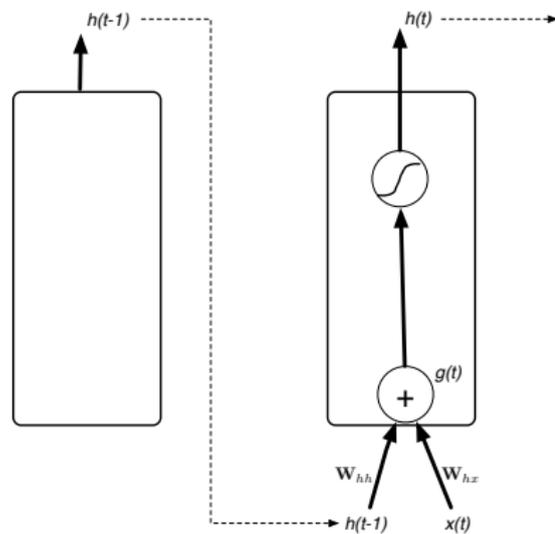
Simple recurrent network unit



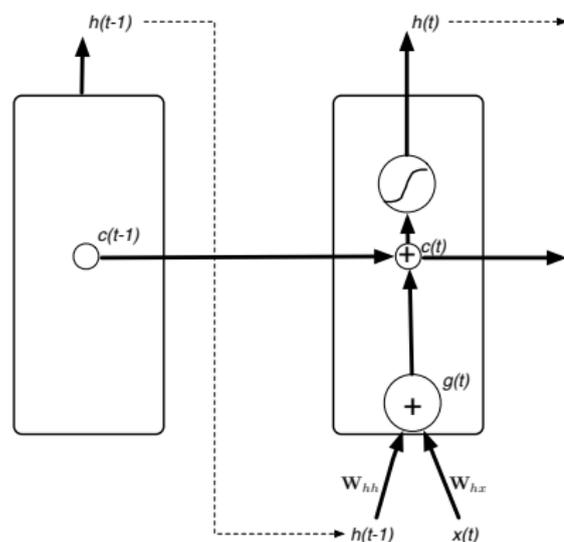
$$\mathbf{g}(t) = \mathbf{W}_{hx}\mathbf{x}(t) + \mathbf{W}_{hh}\mathbf{h}(t-1) + \mathbf{b}_h$$

$$\mathbf{h}(t) = \tanh(\mathbf{g}(t))$$

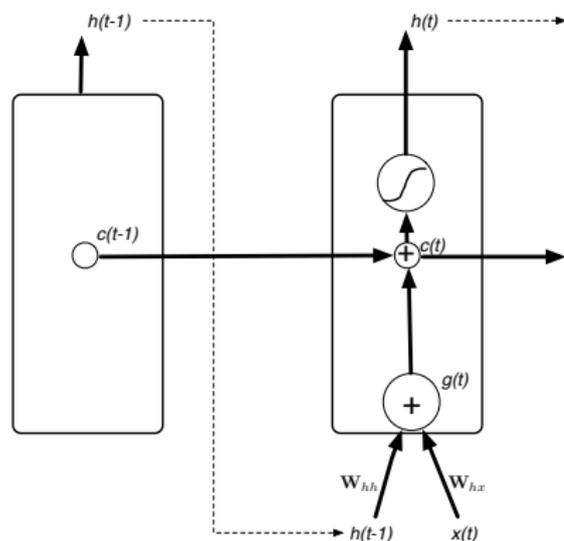
LSTM



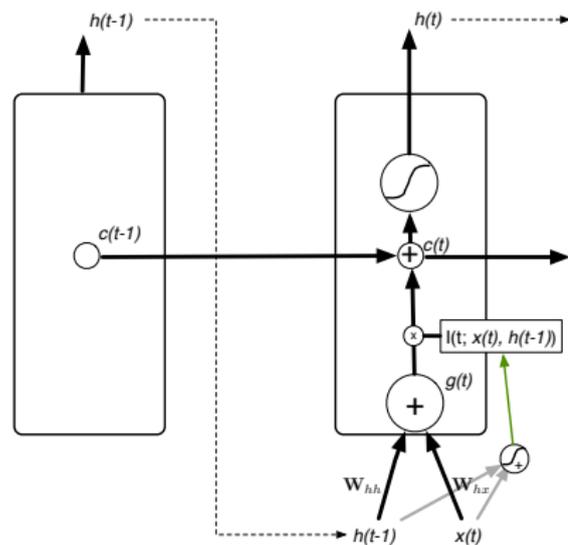
- **Internal recurrent state**
 (“cell”) $c(t)$ combines
 previous state $c(t - 1)$
 and LSTM input $g(t)$



LSTM – Internal recurrent state

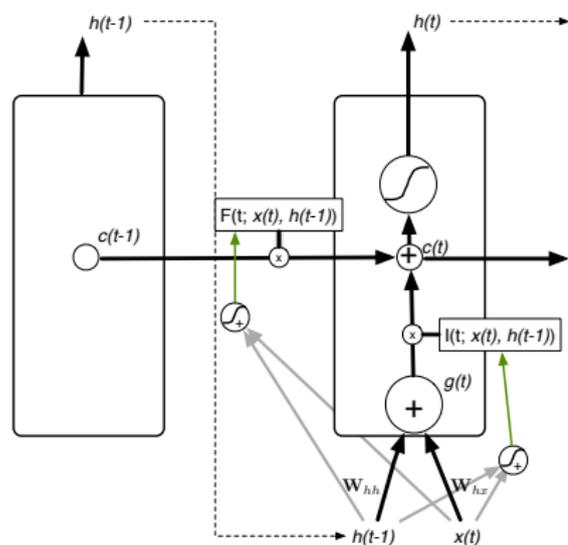


- **Internal recurrent state** (“cell”) $c(t)$ combines previous state $c(t - 1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state



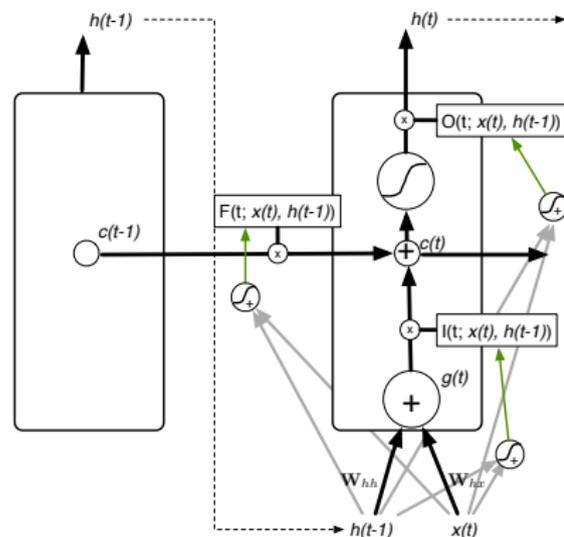
- **Internal recurrent state** (“cell”) $c(t)$ combines previous state $c(t - 1)$ and LSTM input $g(t)$
- **Gates** - weights dependent on the current input and the previous state
- **Input gate**: controls how much input to the unit $g(t)$ is written to the internal state $c(t)$

LSTM – Input and Forget Gate

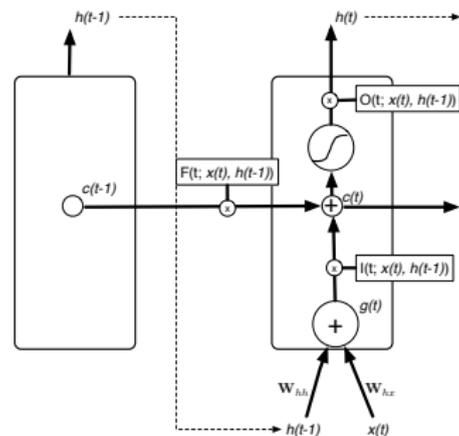


- **Internal recurrent state** (“cell”) $c(t)$ combines previous state $c(t - 1)$ and LSTM input $g(t)$
- Gates - weights dependent on the current input and the previous state
- **Input gate:** controls how much input to the unit $g(t)$ is written to the internal state $c(t)$
- **Forget gate:** controls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

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

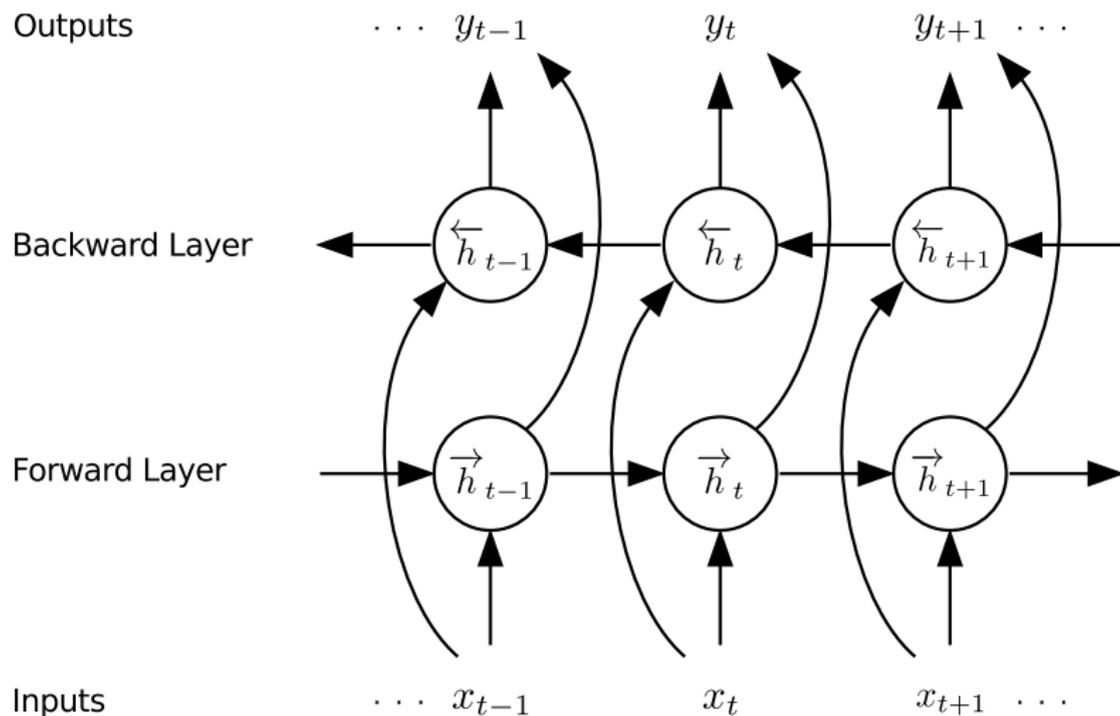


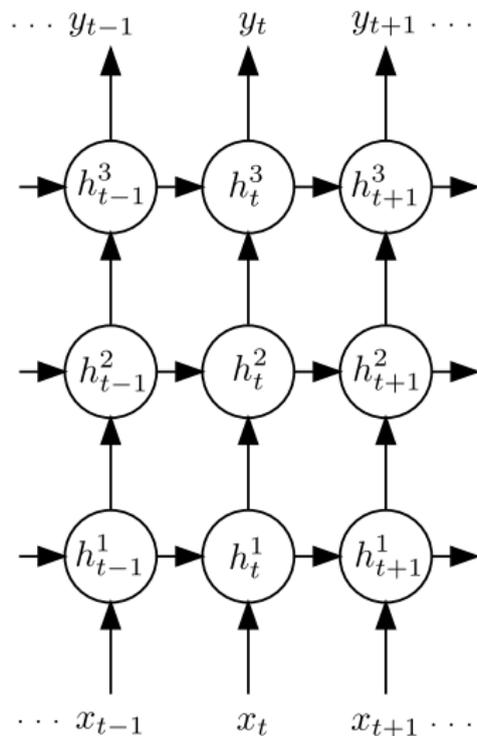
$$\begin{aligned}
 I(t) &= \sigma(W_{ix}x(t) + W_{ih}h(t-1) + b_i) \\
 F(t) &= \sigma(W_{fx}x(t) + W_{fh}h(t-1) + b_f) \\
 O(t) &= \sigma(W_{ox}x(t) + W_{oh}h(t-1) + b_o) \\
 g(t) &= W_{hx}x(t) + W_{hh}h(t-1) + b_h \\
 c(t) &= F(t) \circ c(t-1) + I(t) \circ g(t) \\
 h(t) &= O(t) \circ \tanh(c(t))
 \end{aligned}$$

Avoids the vanishing gradient problem of conventional RNNs

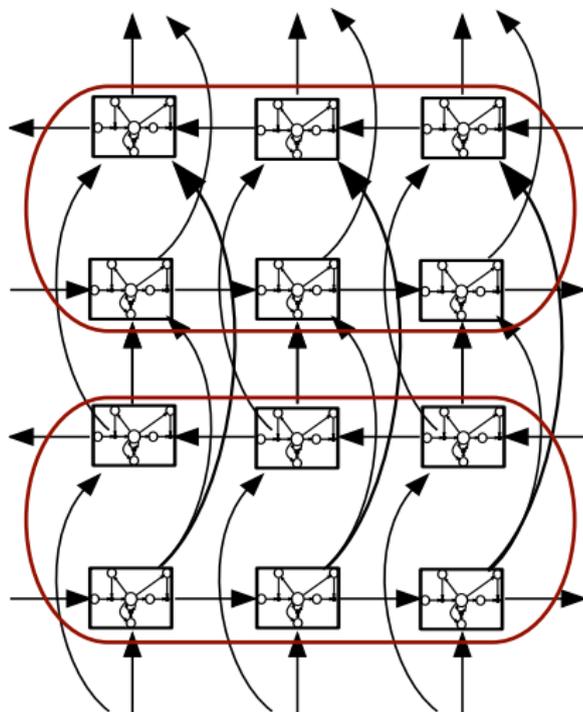
C Olah (2015), Understanding LSTMs, <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Bidirectional RNN



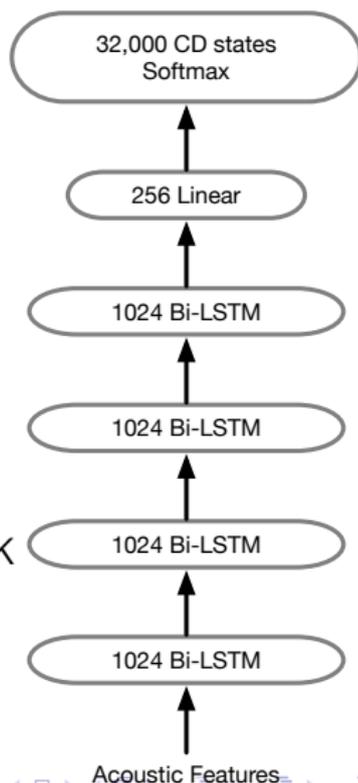


Deep Bidirectional LSTM



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)



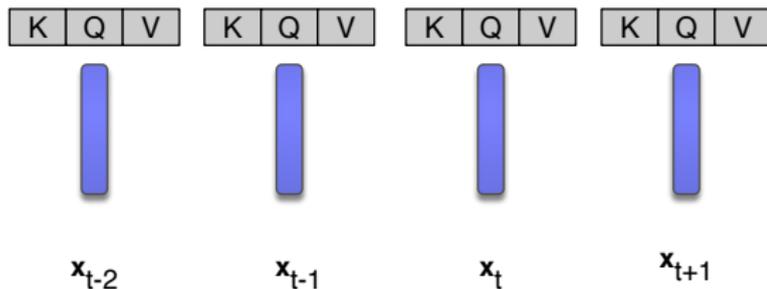
Transformers for ASR

- Transformers have a *self-attention* mechanism
- Now commonly used in end-to-end ASR systems (see later lectures)
- Can be seen as a generalisation of the RNN, better able to attend to more distant input features
- Can be problematic on the relatively long input sequences used in ASR
- Commonly a *positional* encoder is used to compensate for the lack of time information

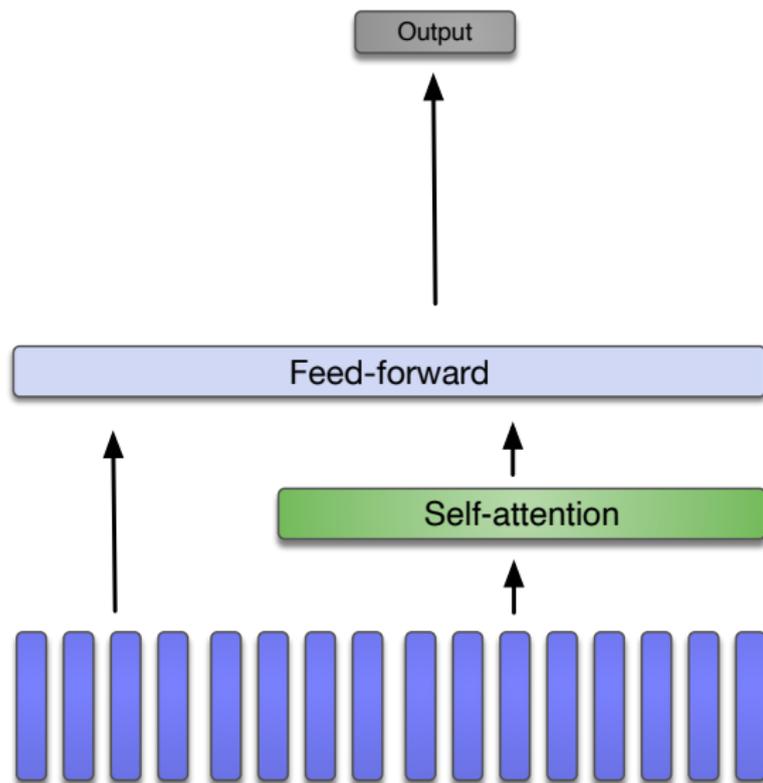
Transformers for ASR

Output

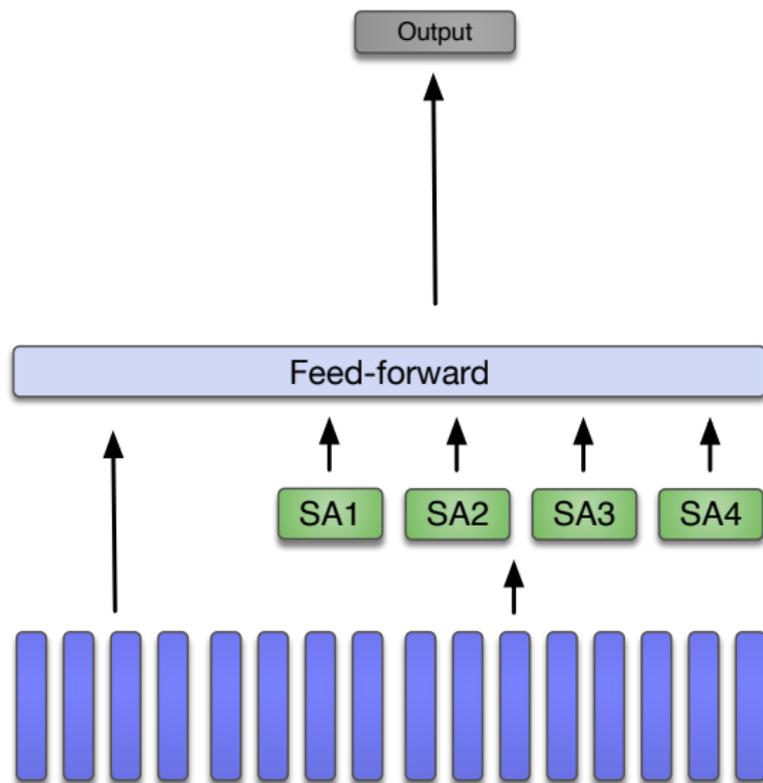
Sum



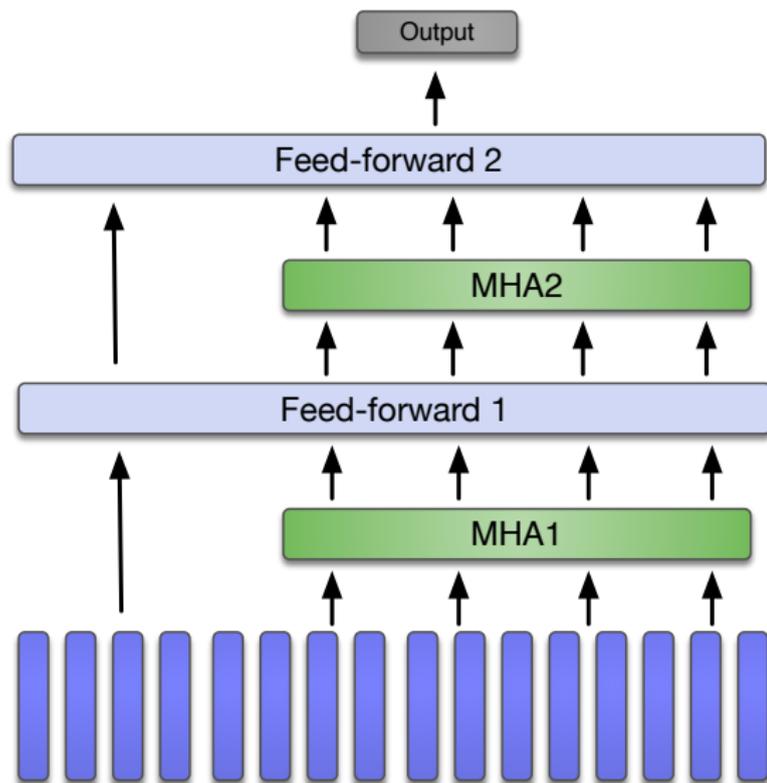
Transformers for ASR



Transformers for ASR



Transformers for ASR



Summary and Conclusions

- Scaling DNNs for large vocabulary speech recognition
- LSTM recurrent networks and TDNNs offer different ways to model temporal context

- 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*