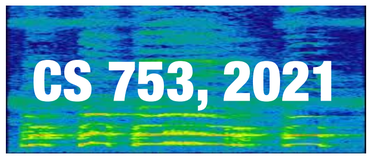
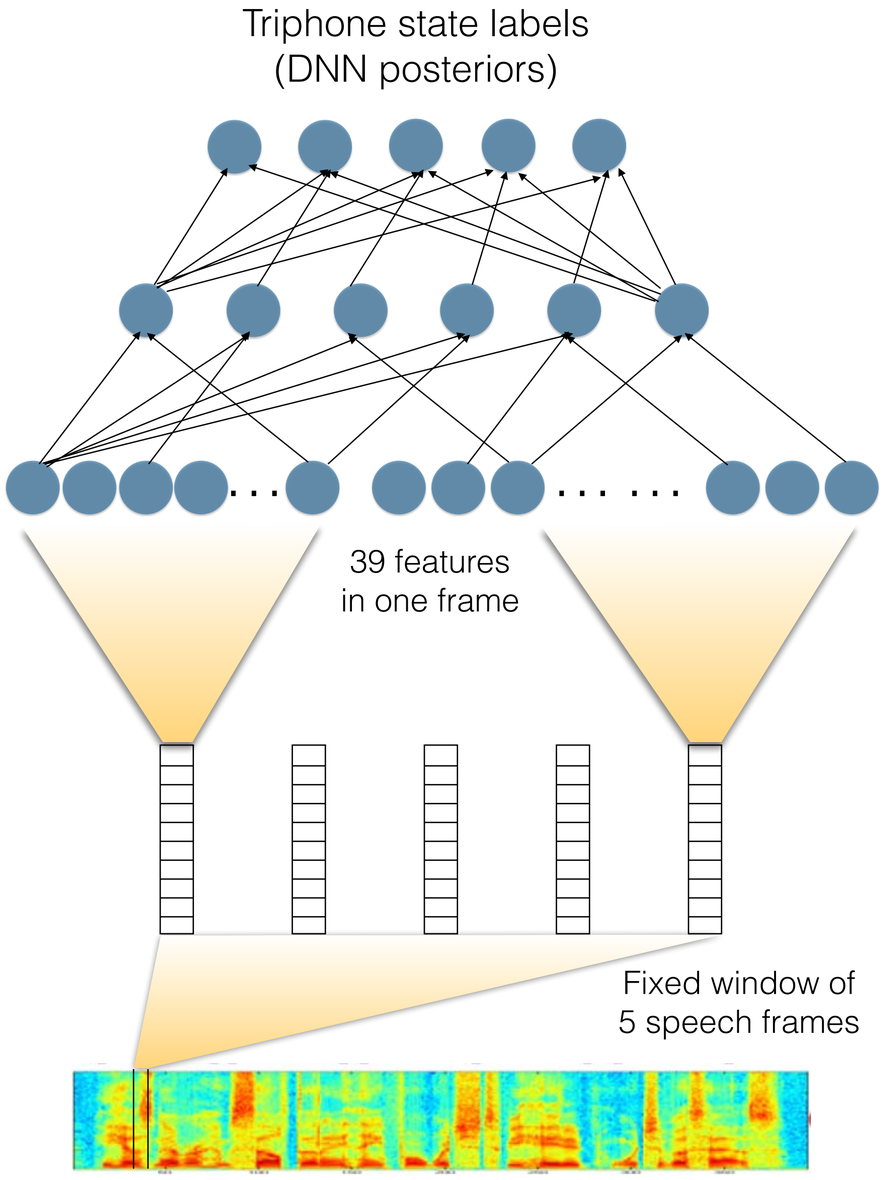
TDNNs and Introduction to RNN Models

Lecture 6b



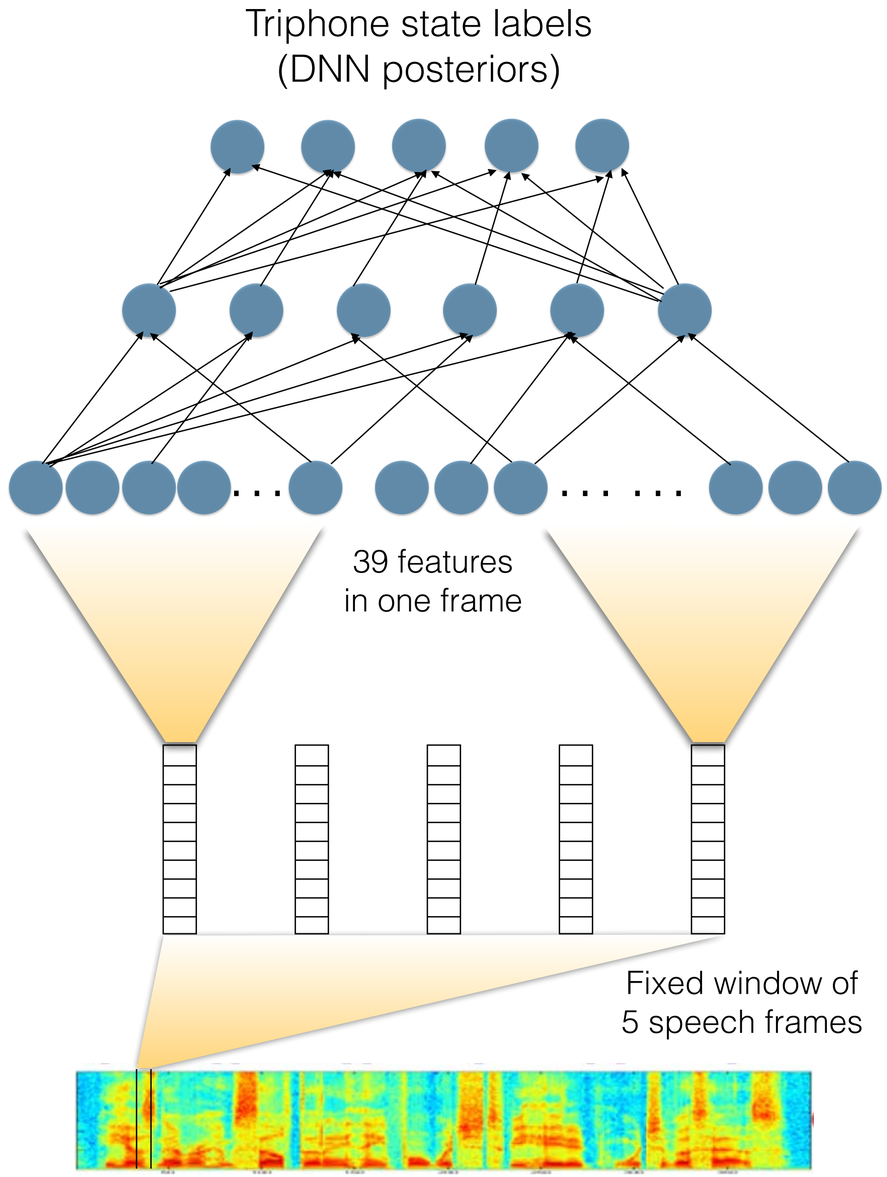
Instructor: Preethi Jyothi, IITB

Recap: Hybrid DNN-HMM Systems



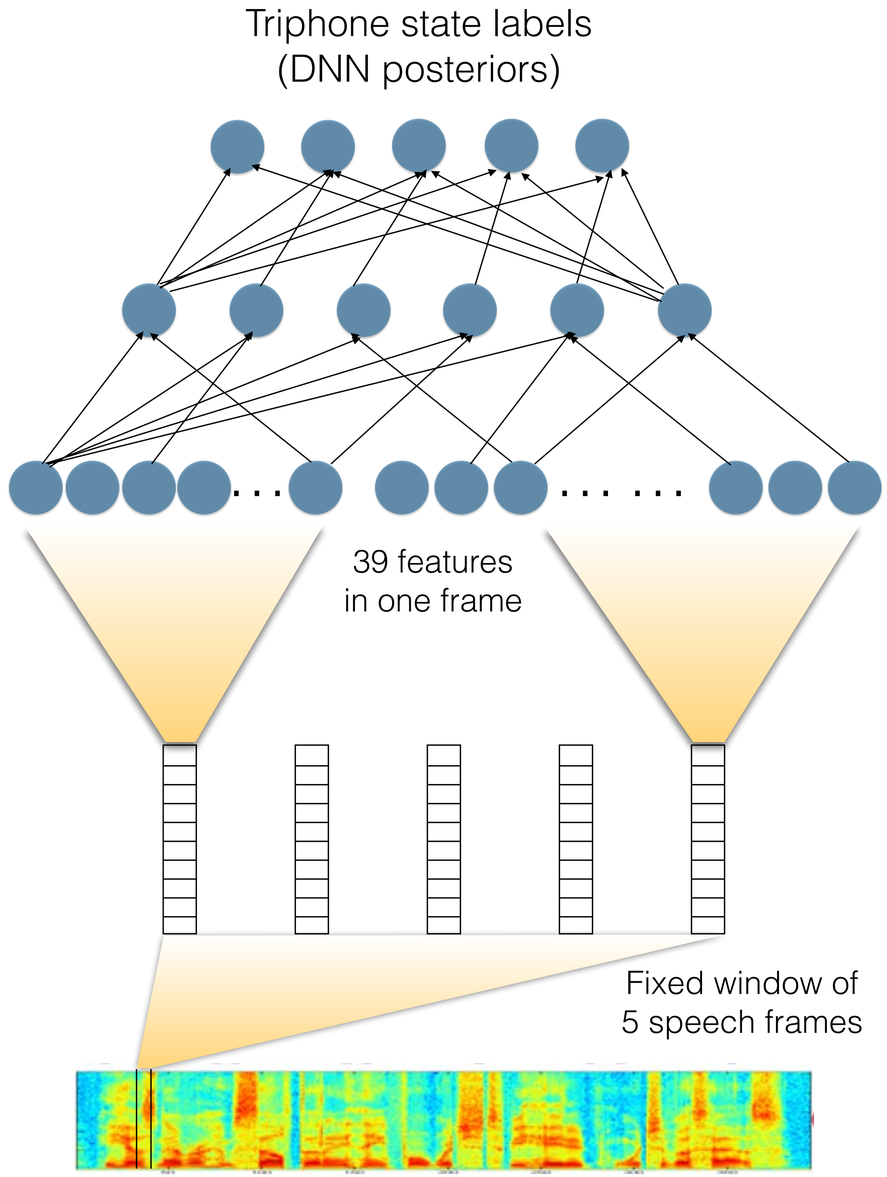
Recap: Hybrid DNN-HMM Systems

- Instead of GMMs, use scaled DNN posteriors as the HMM observation probabilities



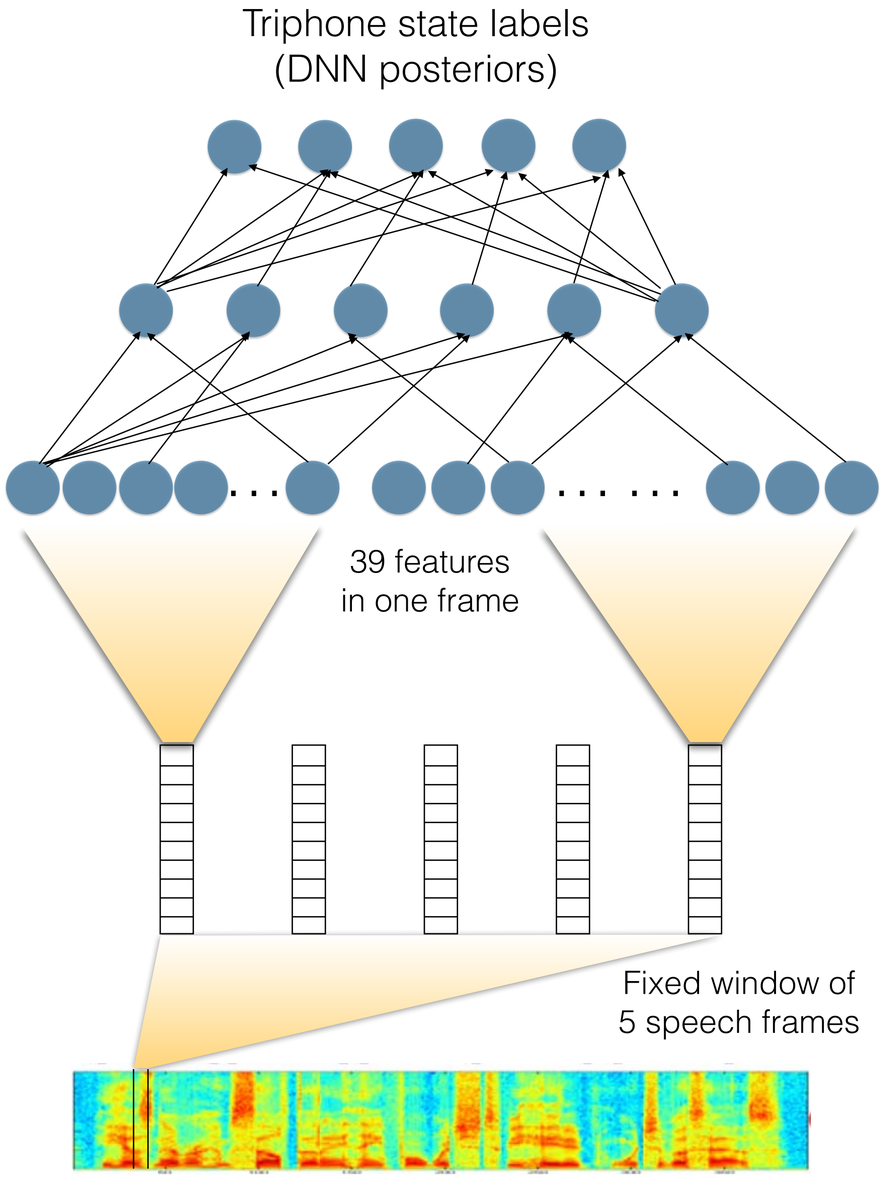
Recap: Hybrid DNN-HMM Systems

- Instead of GMMs, use scaled DNN posteriors as the HMM observation probabilities
- DNN trained using triphone labels derived from a forced alignment “Viterbi” step.

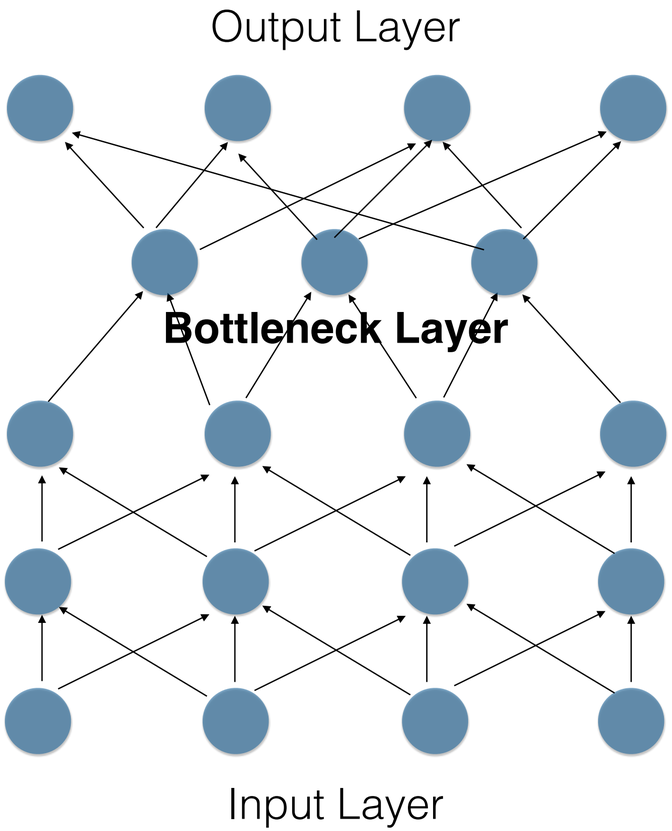


Recap: Hybrid DNN-HMM Systems

- Instead of GMMs, use scaled DNN posteriors as the HMM observation probabilities
- DNN trained using triphone labels derived from a forced alignment "Viterbi" step.
  - *Forced alignment:* Given a training utterance $\{O, W\}$, find the most likely sequence of states (and hence triphone state labels) using a set of trained triphone HMM models, M. Here M is constrained by the triphones in W.

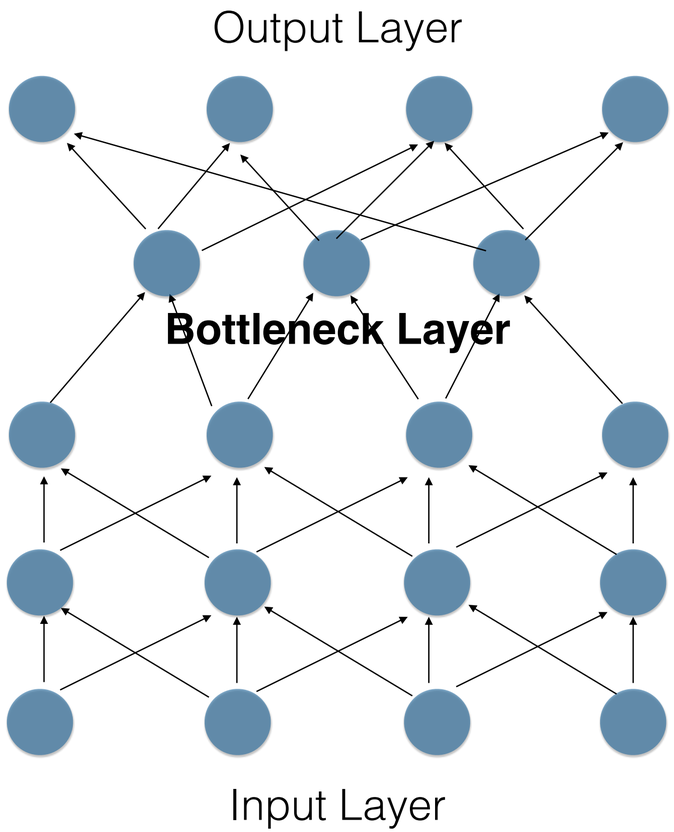


Recap: Tandem DNN-HMM Systems



Recap: Tandem DNN-HMM Systems

- Neural networks are used as “feature extractors” to train HMM-GMM models
- Use a low-dimensional *bottleneck* layer representation to extract features from the bottleneck layer
- These bottleneck features are subsequently fed to GMM-HMMs as input



Feedforward DNNs we've seen so far…

Feedforward DNNs we've seen so far…

- Assume independence among the training instances (modulo the context window of frames)
  - Independent decision made about classifying each individual speech frame
  - Network state is completely reset after each speech frame is processed

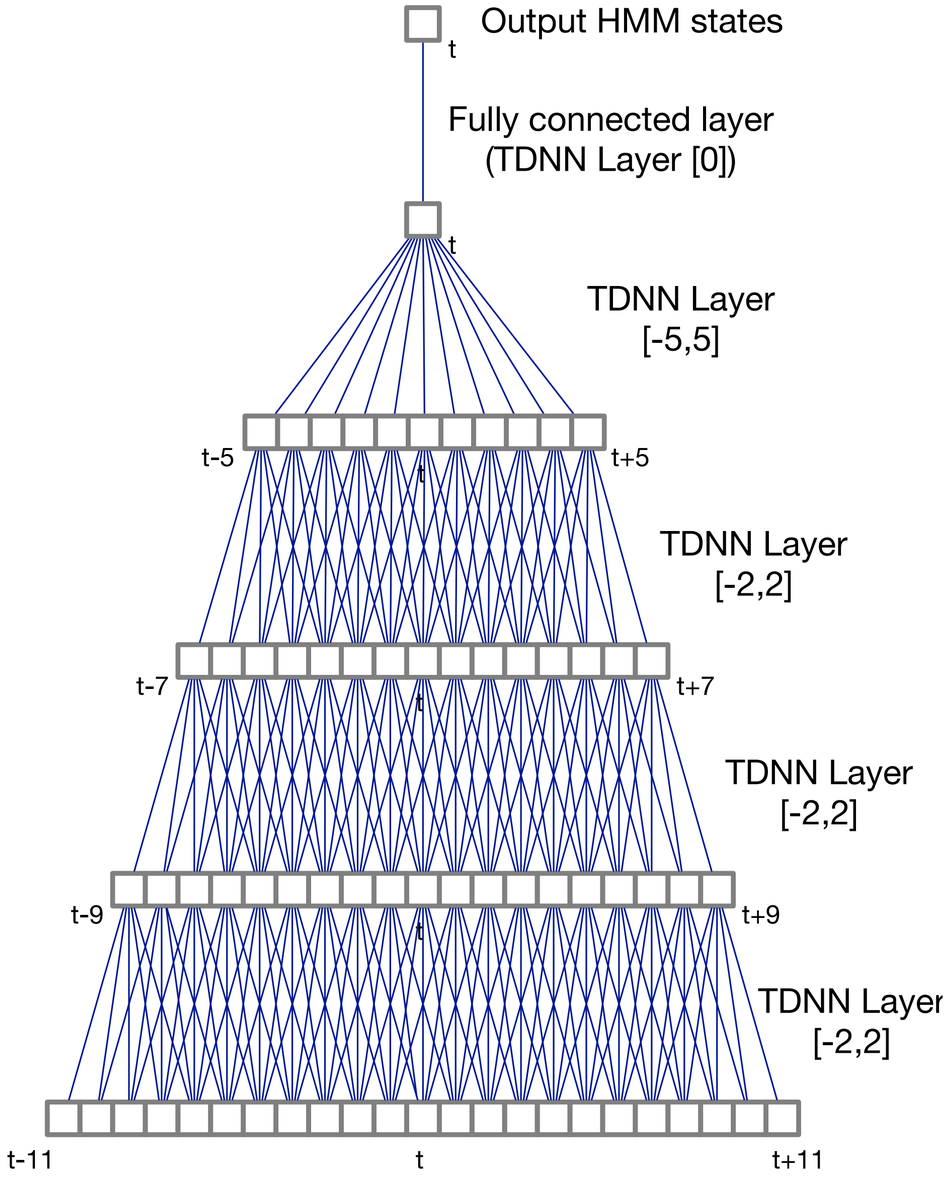
Feedforward DNNs we've seen so far…

- Assume independence among the training instances (modulo the context window of frames)
  - Independent decision made about classifying each individual speech frame
  - Network state is completely reset after each speech frame is processed
- This independence assumption fails for data like speech which has temporal and sequential structure

Feedforward DNNs we've seen so far…

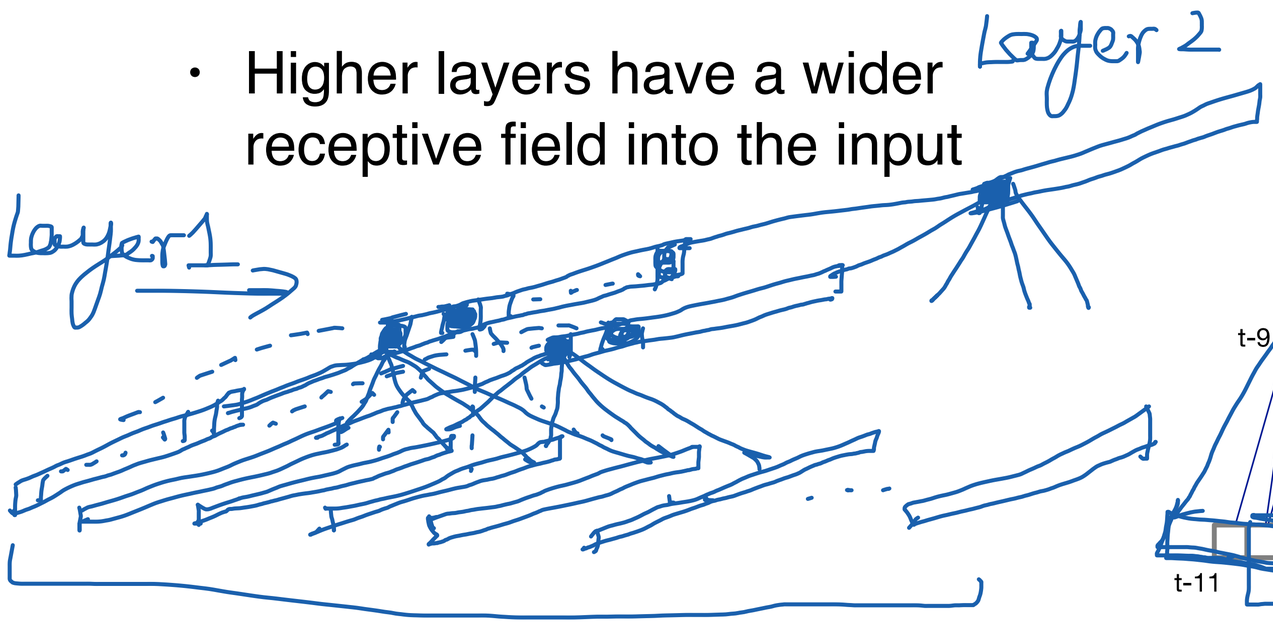
- Assume independence among the training instances (modulo the context window of frames)
  - Independent decision made about classifying each individual speech frame
  - Network state is completely reset after each speech frame is processed
- This independence assumption fails for data like speech which has temporal and sequential structure
- Two model architectures that capture longer ranges of acoustic context:
  1. **Time delay neural networks (TDNNs)**

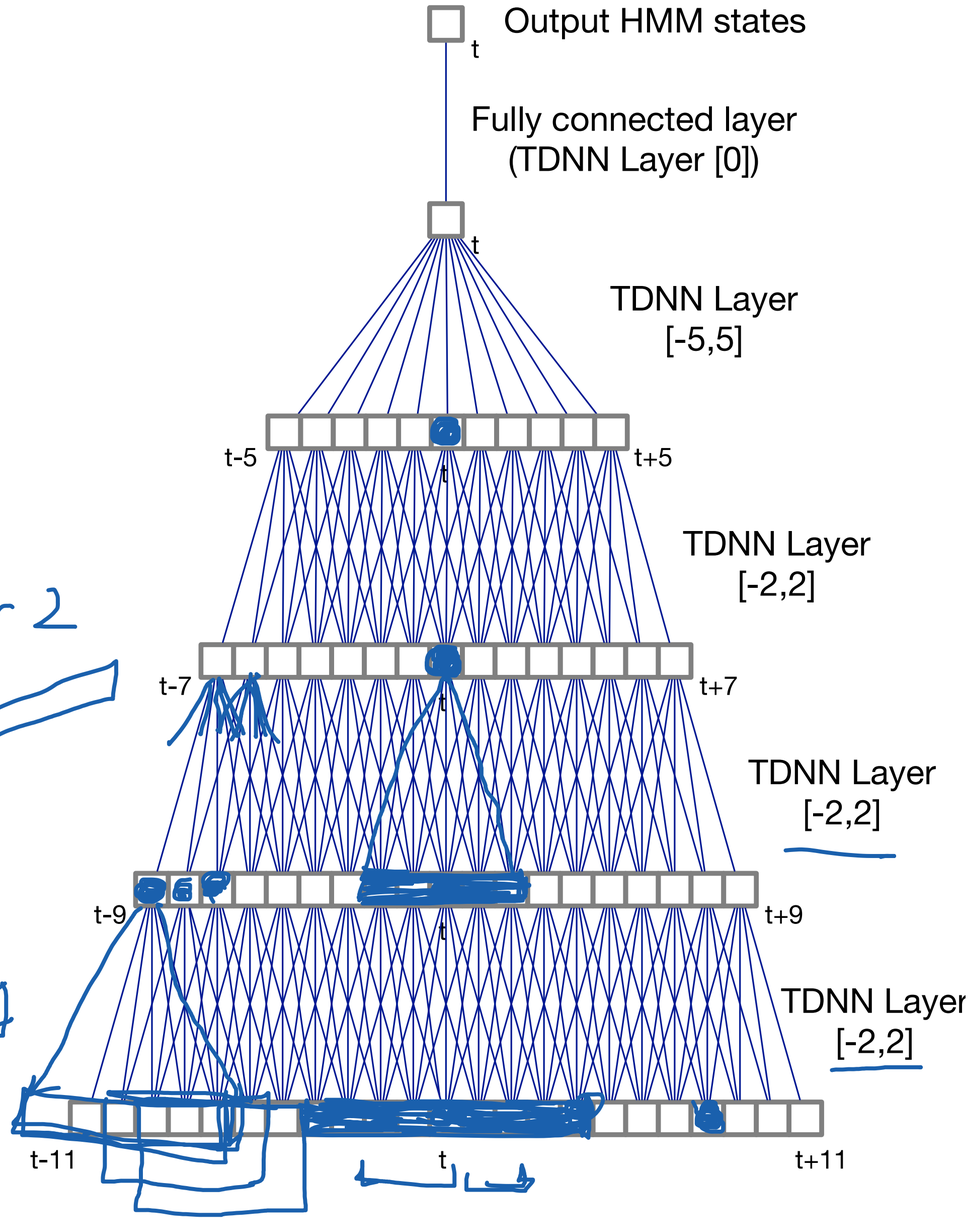
Time Delay Neural Networks



Time Delay Neural Networks

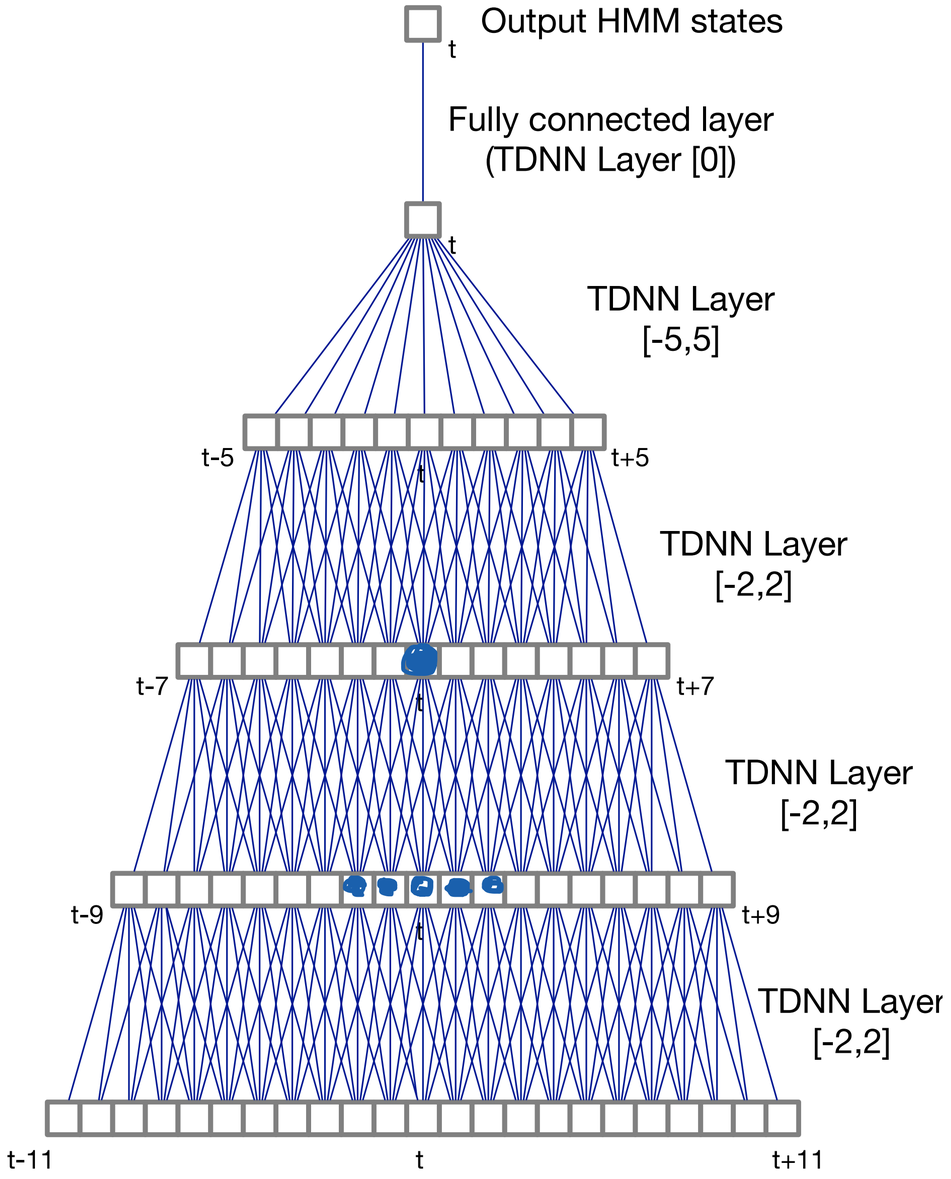
- Each layer in a TDNN acts at a different temporal resolution
  - Processes a context window from the previous layer
  - Higher layers have a wider receptive field into the input





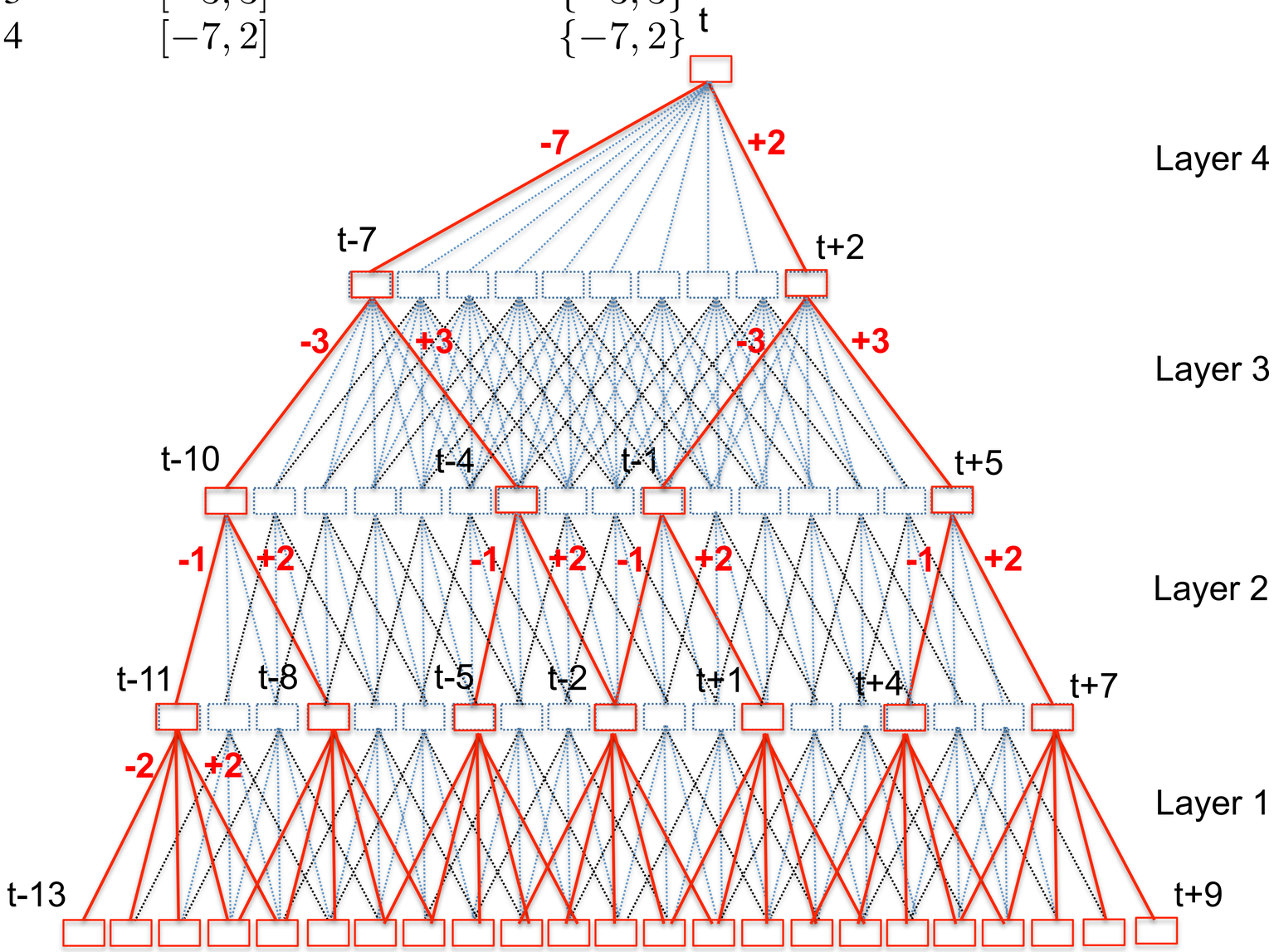
Time Delay Neural Networks

- Each layer in a TDNN acts at a different temporal resolution
  - Processes a context window from the previous layer
  - Higher layers have a wider receptive field into the input
- However, a lot more computation needed than DNNs!



Time Delay Neural Networks

Layer	Input context	Input context with sub-sampling
1	$[-2, +2]$	$[-2, 2]$
2	$[-1, 2]$	$\{-1, 2\}$
3	$[-3, 3]$	$\{-3, 3\}$
4	$[-7, 2]$	$\{-7, 2\}$

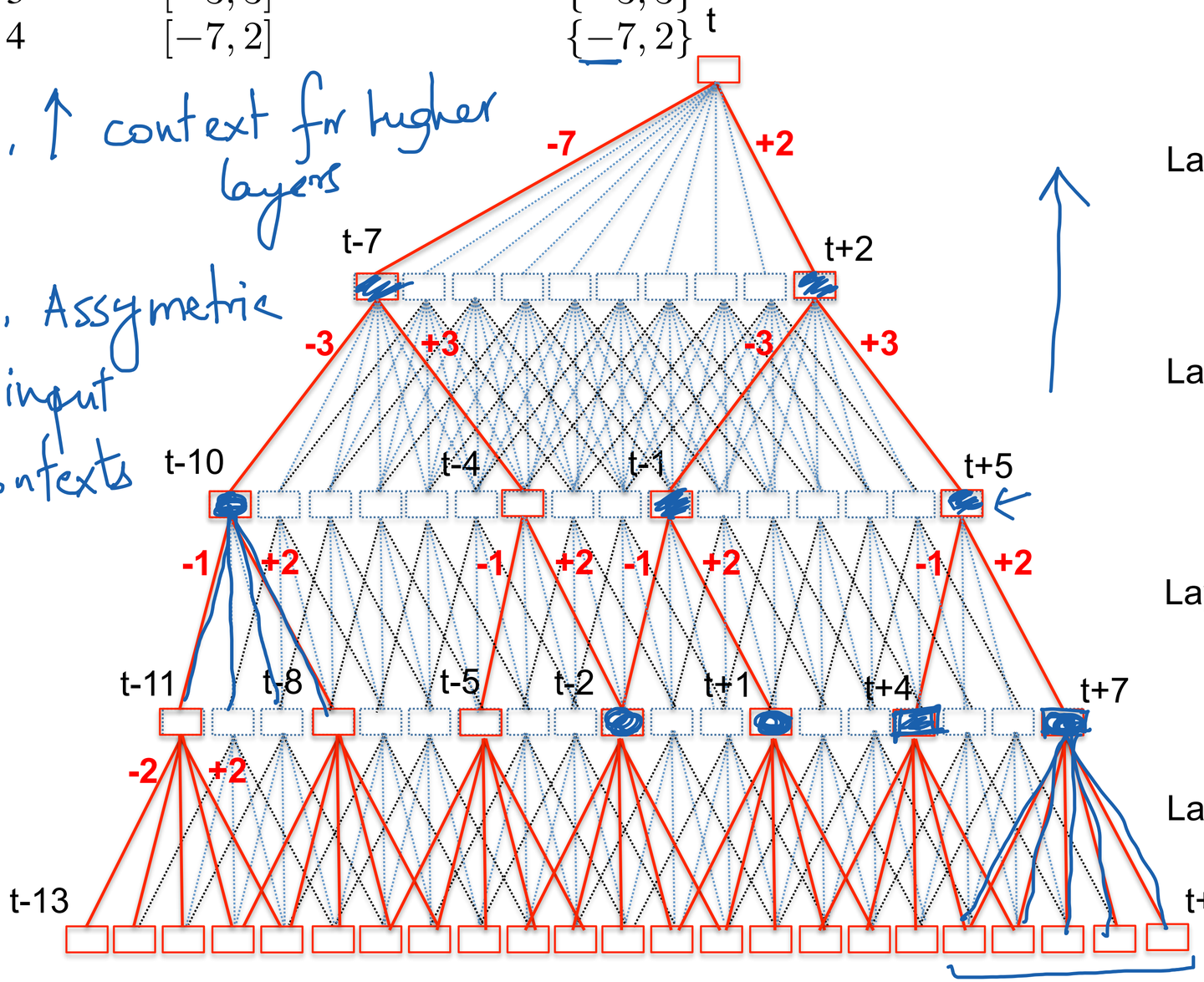


Time Delay Neural Networks

Layer	Input context	Input context with sub-sampling
1	$[-2, +2]$	$[-2, 2]$
2	$[-1, 2]$	$\{-1, 2\}$
3	$[-3, 3]$	$\{-3, 3\}$
4	$[-7, 2]$	$\{-7, 2\}$

1. ↑ context for higher layers

2. Assymetric input contexts



- Large overlaps between input contexts computed at neighbouring time steps
- Assuming neighbouring activations are correlated, how do we exploit this?
  - Subsample by allowing gaps between frames.
  - Splice increasingly wider context in higher layers.

Time Delay Neural Networks

Model	Network Context	Layerwise Context					WER	
		1	2	3	4	5	Total	SWB
DNN-A	$[-7, 7]$	$[-7, 7]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	22.1	15.5
DNN-A_2	$[-7, 7]$	$[-7, 7]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	**21.6**	**15.1**
DNN-B	$[-13, 9]$	$[-13, 9]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	22.3	15.7
DNN-C	$[-16, 9]$	$[-16, 9]$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	22.3	15.7
TDNN-A	$[-7, 7]$	$[-2, 2]$	$\{-2, 2\}$	$\{-3, 4\}$	$\{0\}$	$\{0\}$	21.2	14.6
TDNN-B	$[-9, 7]$	$[-2, 2]$	$\{-2, 2\}$	$\{-5, 3\}$	$\{0\}$	$\{0\}$	21.2	14.5
TDNN-C	$[-11, 7]$	$[-2, 2]$	$\{-1, 1\}$	$\{-2, 2\}$	$\{-6, 2\}$	$\{0\}$	20.9	14.2
TDNN-D	$[-13, 9]$	$[-2, 2]$	$\{-1, 2\}$	$\{-3, 4\}$	$\{-7, 2\}$	$\{0\}$	**20.8**	**14.0**
TDNN-E	$[-16, 9]$	$[-2, 2]$	$\{-2, 2\}$	$\{-5, 3\}$	$\{-7, 2\}$	$\{0\}$	20.9	14.2

Feedforward DNNs we've seen so far…

Feedforward DNNs we've seen so far…

- Assume independence among the training instances
  - Independent decision made about classifying each individual speech frame
  - Network state is completely reset after each speech frame is processed

Feedforward DNNs we've seen so far…

- Assume independence among the training instances
  - Independent decision made about classifying each individual speech frame
  - Network state is completely reset after each speech frame is processed
- This independence assumption fails for data like speech which has temporal and sequential structure

Feedforward DNNs we've seen so far…

- Assume independence among the training instances
  - Independent decision made about classifying each individual speech frame
  - Network state is completely reset after each speech frame is processed
- This independence assumption fails for data like speech which has temporal and sequential structure
- Two model architectures that capture longer ranges of acoustic context:
  1. Time delay neural networks (TDNNs)
  2. **Recurrent neural networks (RNNs)**

Recurrent Neural Networks

Recurrent Neural Networks

- Recurrent Neural Networks (RNNs) work naturally with sequential data and process it one element at a time

Recurrent Neural Networks

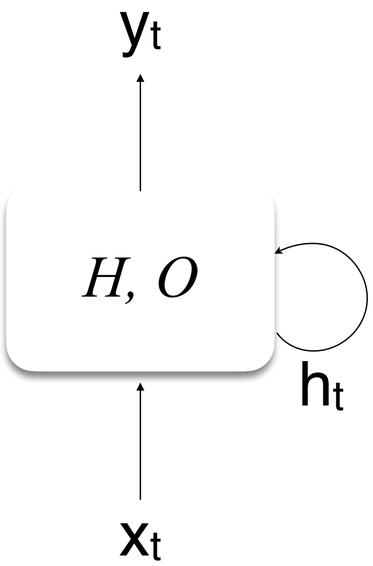
- Recurrent Neural Networks (RNNs) work naturally with sequential data and process it one element at a time

- HMMs also similarly attempt to model time dependencies. How's it different?

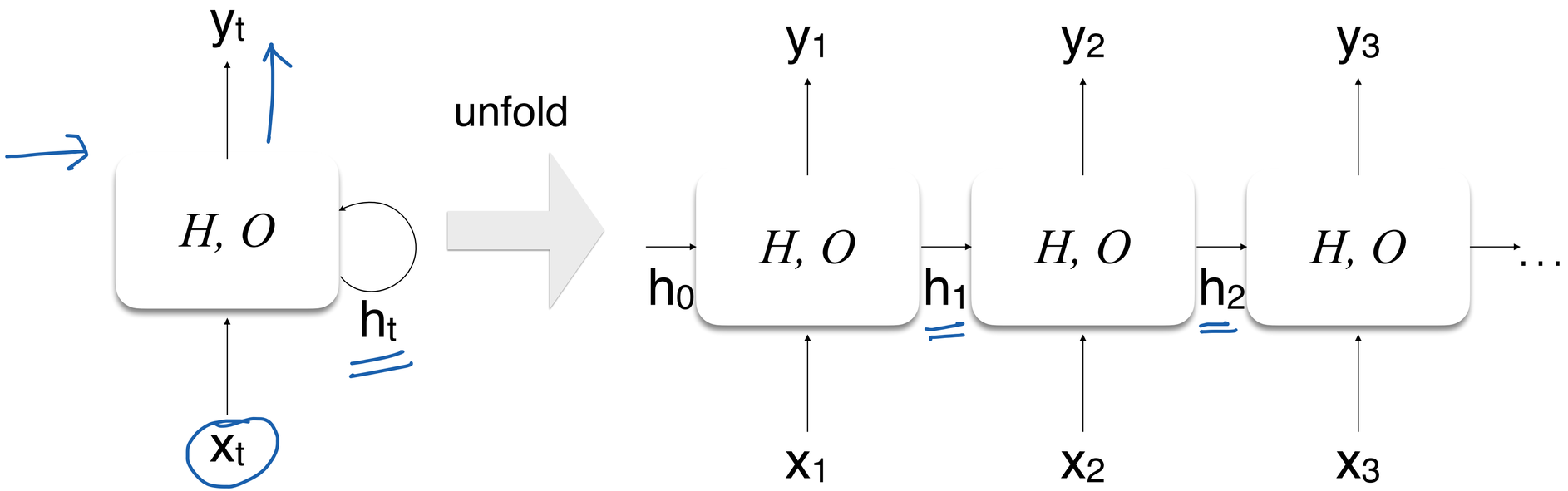
  - HMMs are limited by the size of the state space. Inference becomes intractable if the state space grows very large!

  - What about RNNs?

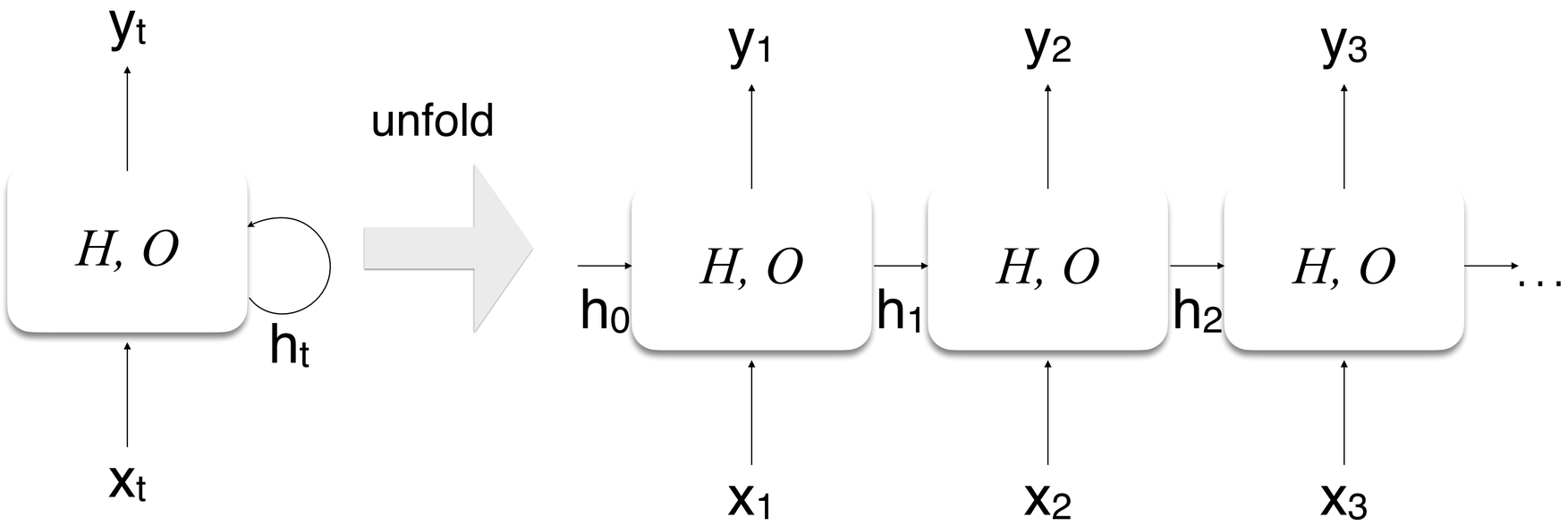
RNN definition



RNN definition



RNN definition



Two main equations govern RNNs:

$$h_t = H(Wx_t + Vh_{t-1} + b^{(h)})$$

$$y_t = O(Uh_t + b^{(y)})$$

where W, V, U are matrices of input-hidden weights, hidden-hidden weights and hidden-output weights resp; $b^{(h)}$ and $b^{(y)}$ are bias vectors and H is the activation function applied to the hidden layer

Recurrent Neural Networks

- Recurrent Neural Networks (RNNs) work naturally with sequential data and process it one element at a time

- HMMs also similarly attempt to model time dependencies. How's it different?

  - HMMs are limited by the size of the state space. Inference becomes intractable if the state space grows very large!

  - What about RNNs? RNNs are designed to capture long-range dependencies unlike HMMs: Network state is exponential in the number of nodes in a hidden layer

Training RNNs

Training RNNs

- An unrolled RNN is just a very deep feedforward network

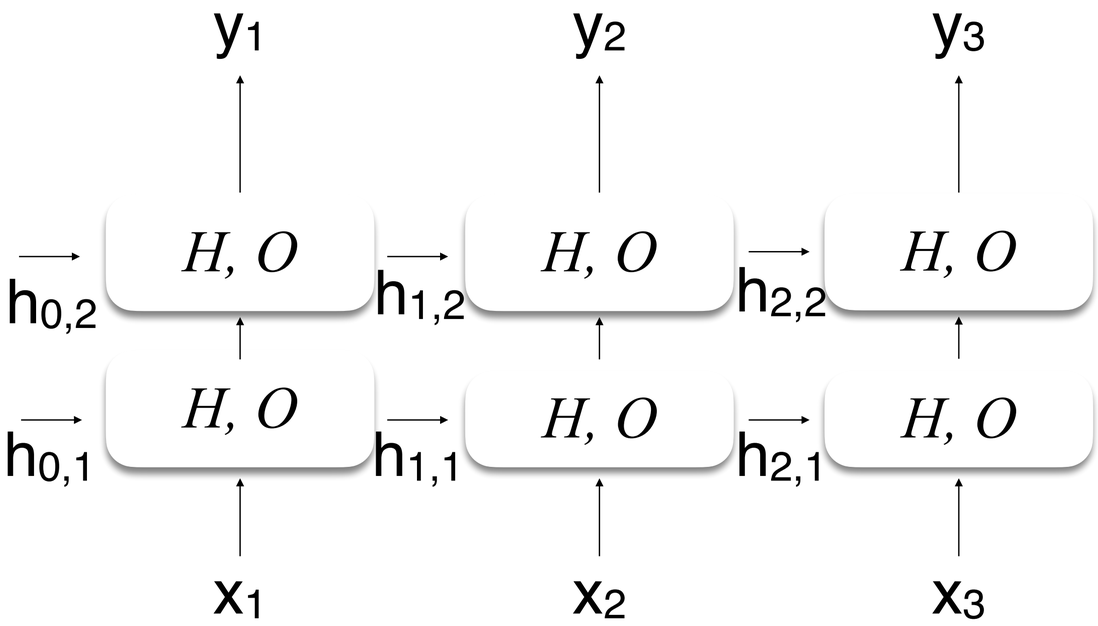
Training RNNs

- An unrolled RNN is just a very deep feedforward network
- For a given input sequence:
  - create the unrolled network
  - add a loss function node to the network
  - then, use backpropagation to compute the gradients

Training RNNs

- An unrolled RNN is just a very deep feedforward network
- For a given input sequence:
  - create the unrolled network
  - add a loss function node to the network
  - then, use backpropagation to compute the gradients
- This algorithm is known as backpropagation through time (BPTT)

Deep RNNs



- RNNs can be stacked in layers to form deep RNNs
- Empirically shown to perform better than shallow RNNs on ASR [G13]

[G13] A. Graves, A . Mohamed, G. Hinton, “Speech Recognition with Deep Recurrent Neural Networks”, ICASSP, 2013.

Vanilla RNN Model

$$h_t = H(Wx_t + Vh_{t-1} + b^{(h)})$$

$$y_t = O(Uh_t + b^{(y)})$$

H : element wise application of the sigmoid or tanh function

O : the softmax function

Vanilla RNN Model

$$h_t = H(Wx_t + Vh_{t-1} + b^{(h)})$$

$$y_t = O(Uh_t + b^{(y)})$$

H : element wise application of the sigmoid or tanh function

O : the softmax function

Run into problems of exploding and vanishing gradients.

Exploding/Vanishing Gradients

- In deep networks, gradients in early layers are computed as the product of terms from all the later layers

- This leads to unstable gradients:

  - If the terms in later layers are large enough, gradients in early layers (which is the product of these terms) can grow exponentially large: *Exploding gradients*

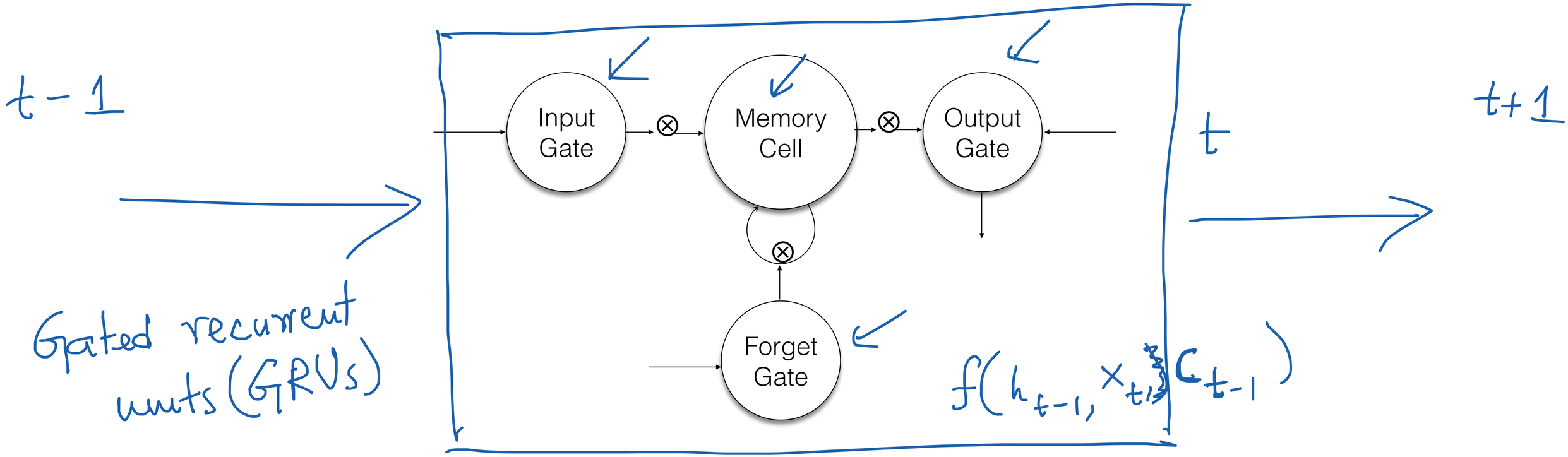
  - If the terms are in later layers are small, gradients in early layers will tend to exponentially decrease: *Vanishing gradients*

Exploding/Vanishing Gradients

- In deep networks, gradients in early layers are computed as the product of terms from all the later layers
- This leads to unstable gradients:
  - If the terms in later layers are large enough, gradients in early layers (which is the product of these terms) can grow exponentially large: *Exploding gradients*
  - If the terms are in later layers are small, gradients in early layers will tend to exponentially decrease: *Vanishing gradients*
- To address this problem in RNNs, Long Short Term Memory (LSTM) units were proposed [HS97]

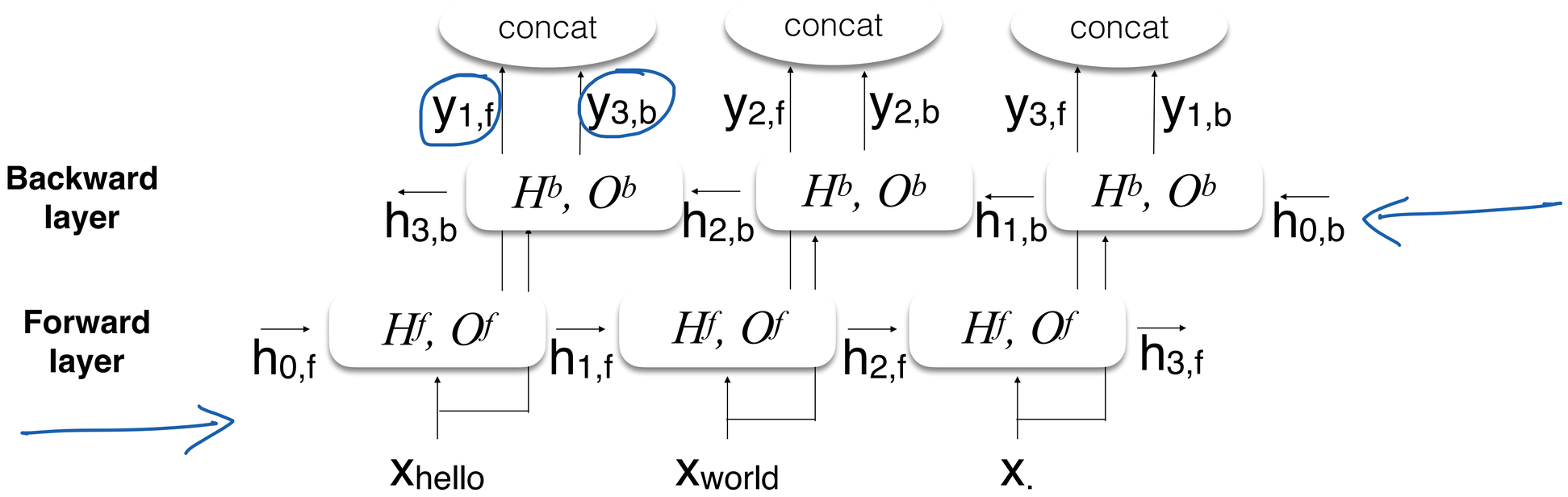
[HS97] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, 1997.

Long Short Term Memory Cells



- Memory cell: Neuron that stores information over long time periods
- Forget gate: When on, memory cell retains previous contents. Otherwise, memory cell forgets contents.
- When input gate is on, write into memory cell
- When output gate is on, read from the memory cell

Bidirectional RNNs



- BiRNNs process the data in both directions with two separate hidden layers
- Outputs from both hidden layers are concatenated at each position

ASR with RNNs

ASR with RNNs

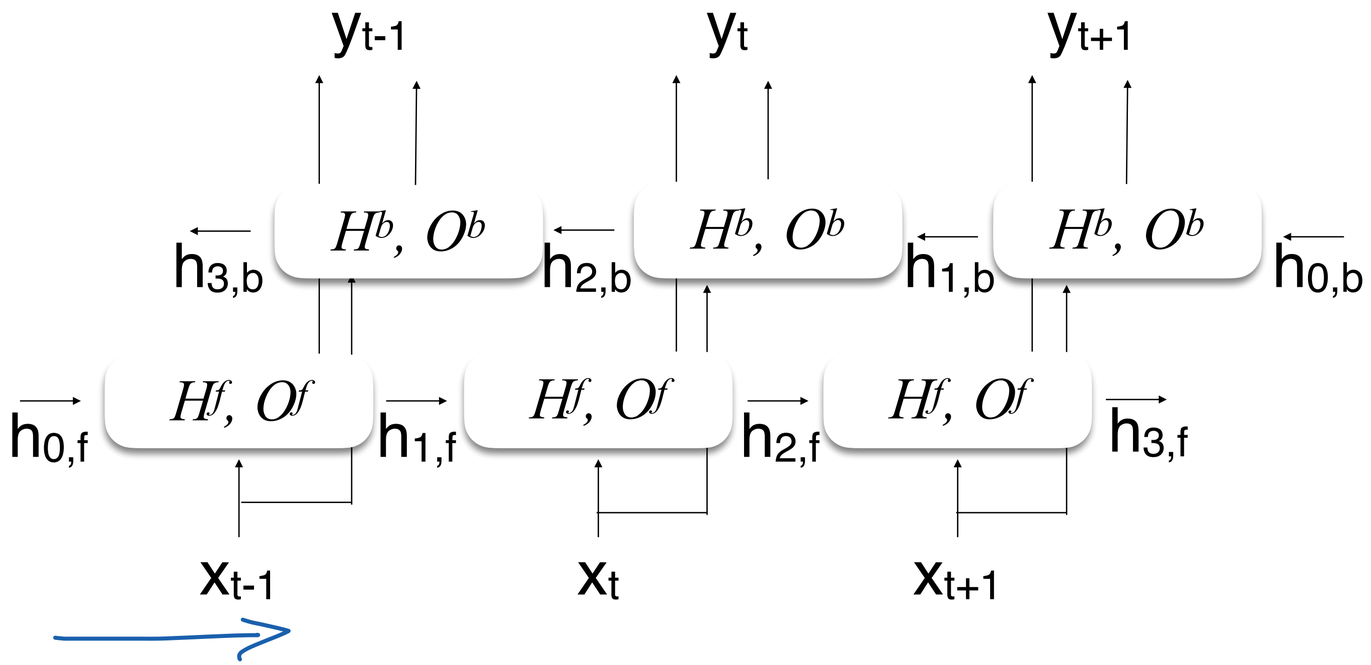
- We have seen how neural networks can be used for acoustic models in ASR systems
  - Main limitation: Frame-level training targets derived from HMM-based alignments

ASR with RNNs

- We have seen how neural networks can be used for acoustic models in ASR systems
  - Main limitation: Frame-level training targets derived from HMM-based alignments
- Goal: Single RNN model that addresses this issues and does not rely on HMM-based alignments [G14]

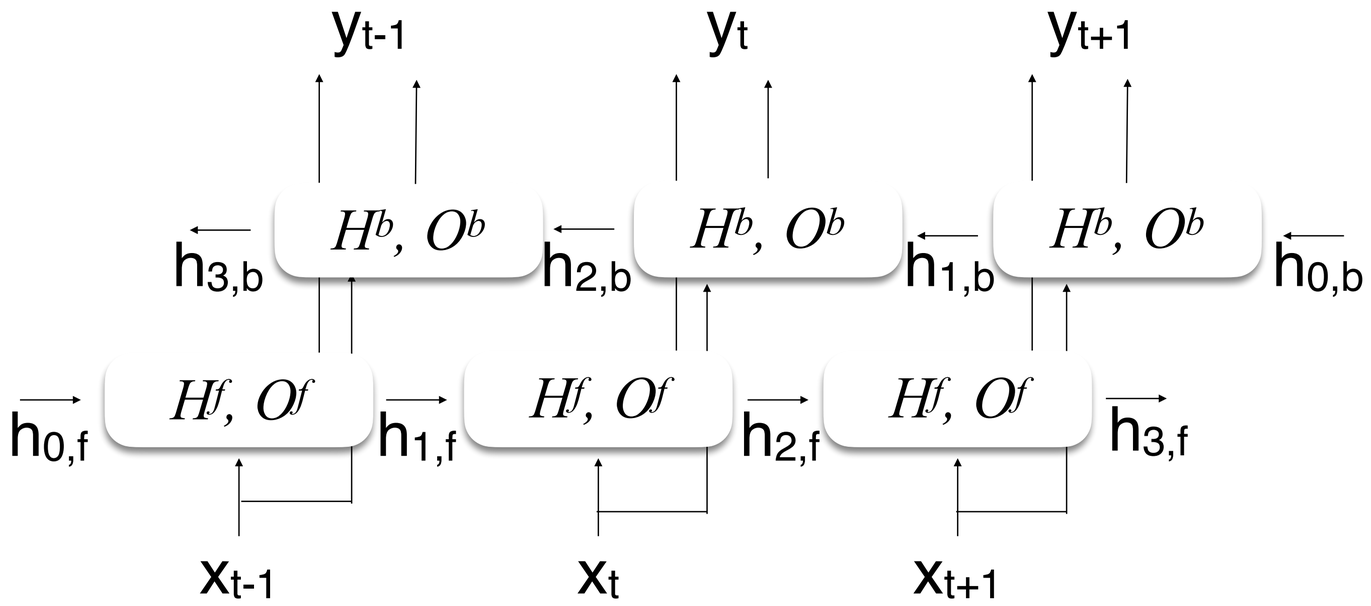
[G14] A. Graves, N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks”, ICML, 2014.

RNN-based Acoustic Model



- H was implemented using LSTMs in [G13]. Input: Acoustic feature vectors, one per frame; Output: Characters + space

RNN-based Acoustic Model

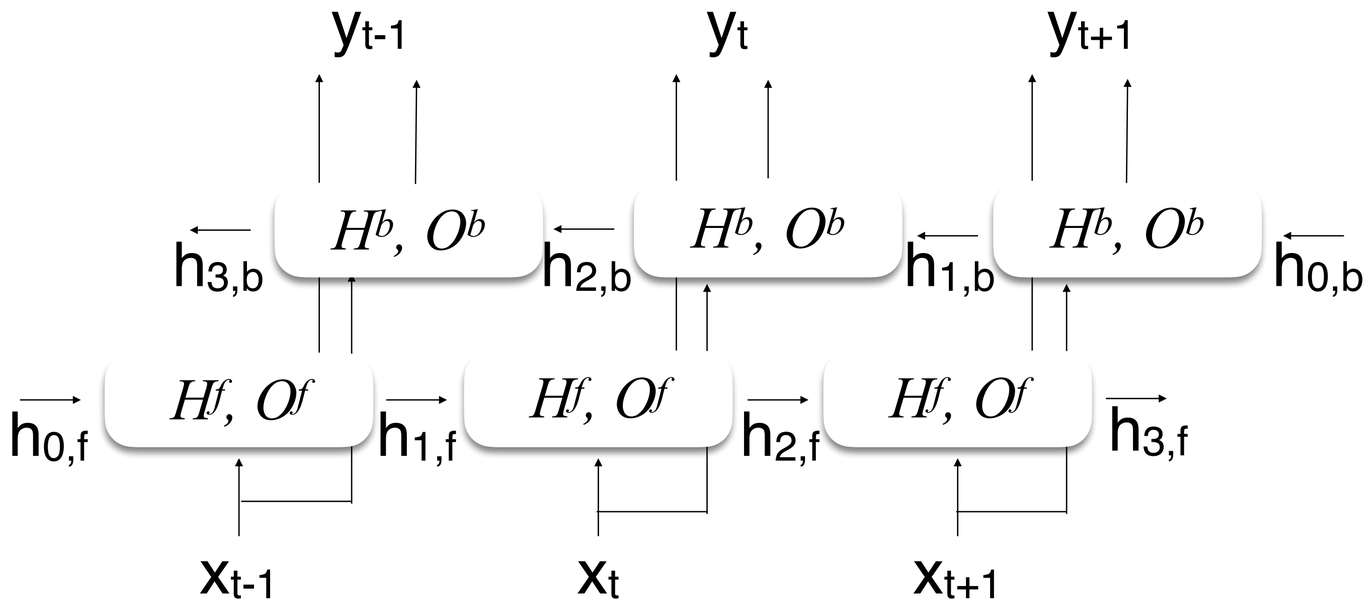


- H was implemented using LSTMs in [G13]. Input: Acoustic feature vectors, one per frame; Output: Characters + space

---

[G13] A. Graves, et al., “Speech recognition with deep recurrent neural networks”, ICASSP, 2013.

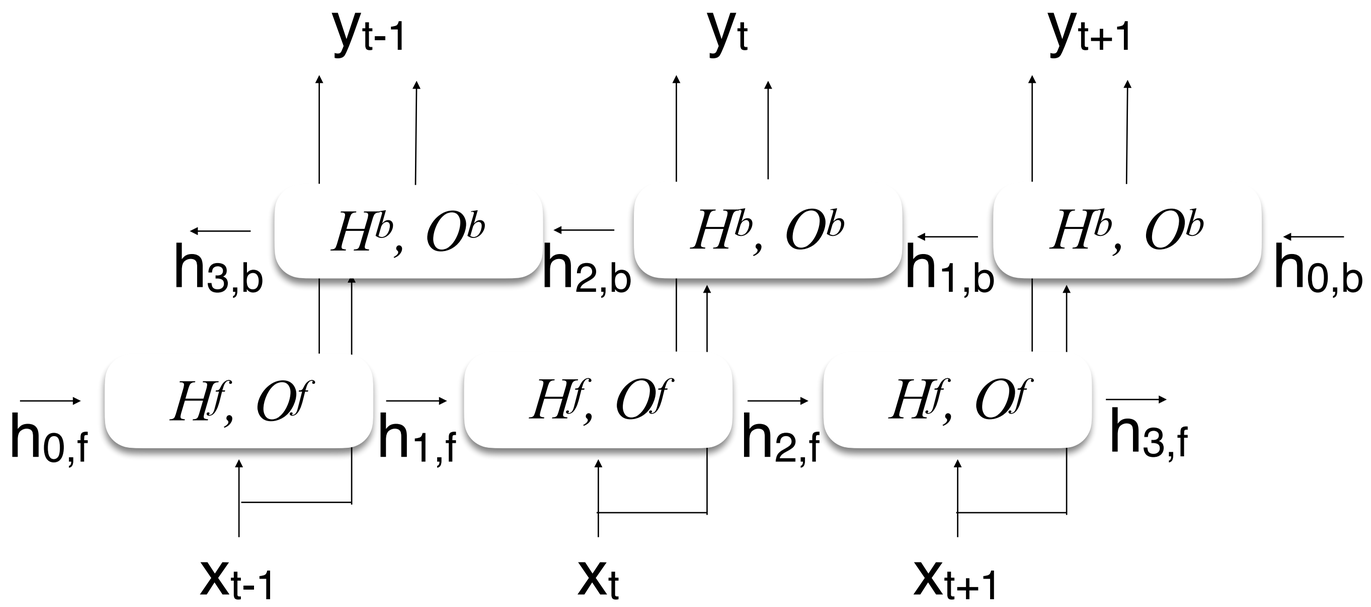
RNN-based Acoustic Model



- H was implemented using LSTMs in [G13]. Input: Acoustic feature vectors, one per frame; Output: Characters + space
- Deep bidirectional LSTM networks were used to do phone recognition on TIMIT

[G13] A. Graves, et al., “Speech recognition with deep recurrent neural networks”, ICASSP, 2013.

RNN-based Acoustic Model



- H was implemented using LSTMs in [G13]. Input: Acoustic feature vectors, one per frame; Output: Characters + space
- Deep bidirectional LSTM networks were used to do phone recognition on TIMIT
- Trained using the Connectionist Temporal Classification (CTC) loss [covered in later class]

[G13] A. Graves, et al., "Speech recognition with deep recurrent neural networks", ICASSP, 2013.

RNN-based Acoustic Model

NETWORK	WEIGHTS	EPOCHS	PER
CTC-3L-500H-TANH	3.7M	107	37.6%
CTC-1L-250H	0.8M	82	23.9%
CTC-1L-622H	3.8M	87	23.0%
CTC-2L-250H	2.3M	55	21.0%
CTC-3L-421H-UNI	3.8M	115	19.6%
CTC-3L-250H	3.8M	124	18.6%
CTC-5L-250H	6.8M	150	18.4%

TIMIT phoneme recognition results

RNN-based Acoustic Model

NETWORK	WEIGHTS	EPOCHS	PER
CTC-3L-500H-TANH	3.7M	107	37.6%
CTC-1L-250H	0.8M	82	23.9%
CTC-1L-622H	3.8M	87	23.0%
CTC-2L-250H	2.3M	55	21.0%
CTC-3L-421H-UNI	3.8M	115	19.6%
CTC-3L-250H	3.8M	124	18.6%
CTC-5L-250H	6.8M	150	18.4%

TIMIT phoneme recognition results

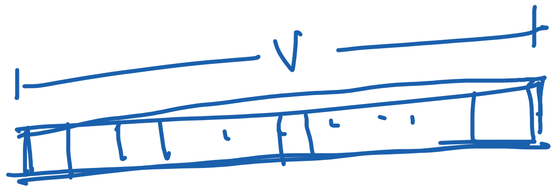
[G13] A. Graves, et al., “Speech recognition with deep recurrent neural networks”, ICASSP, 2013.

RNNs For Language Models?

Word representations in Ngram models

- In standard Ngram models, words are represented in the discrete space involving the vocabulary
- Limits the possibility of truly interpolating probabilities of unseen Ngrams
- Can we build a representation for words in the continuous space?

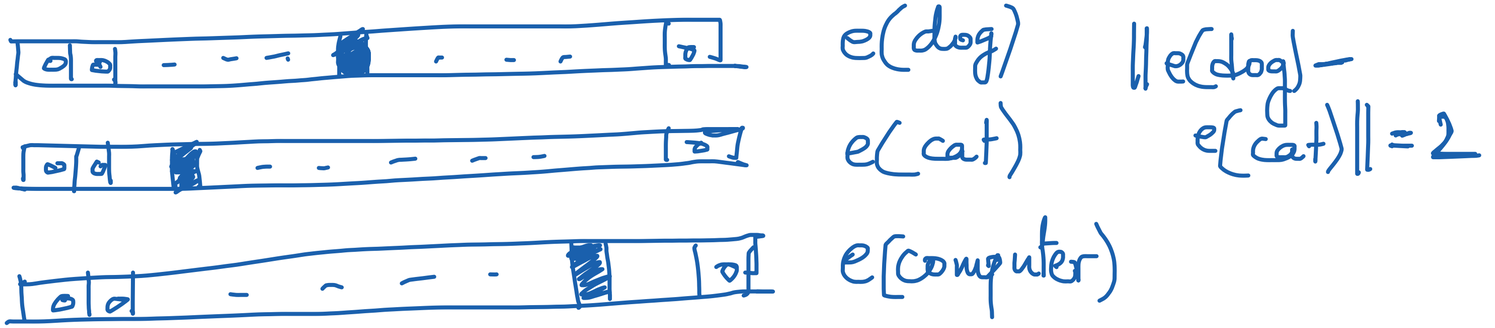
Word representations



- 1-hot representation:
  - Each word is given an index in {1, … , V}. The 1-hot vector $f_i \in R^V$ contains zeros everywhere except for the i^{th} dimension being 1

Word representations

- 1-hot representation:
  - Each word is given an index in {1, … , V}. The 1-hot vector $f_i \in R^V$ contains zeros everywhere except for the i^{th} dimension being 1
- 1-hot form, however, doesn't encode information about word similarity



Word representations

- 1-hot representation:
  - Each word is given an index in $\{1, \ldots, V\}$. The 1-hot vector $f_i \in R^V$ contains zeros everywhere except for the i^{th} dimension being 1
- 1-hot form, however, doesn't encode information about word similarity
- Distributed (or continuous) representation: Each word is associated with a dense vector. Based on the "distributional hypothesis".
  E.g. dog → {-0.02, -0.37, 0.26, 0.25, -0.11, 0.34}

Word embeddings

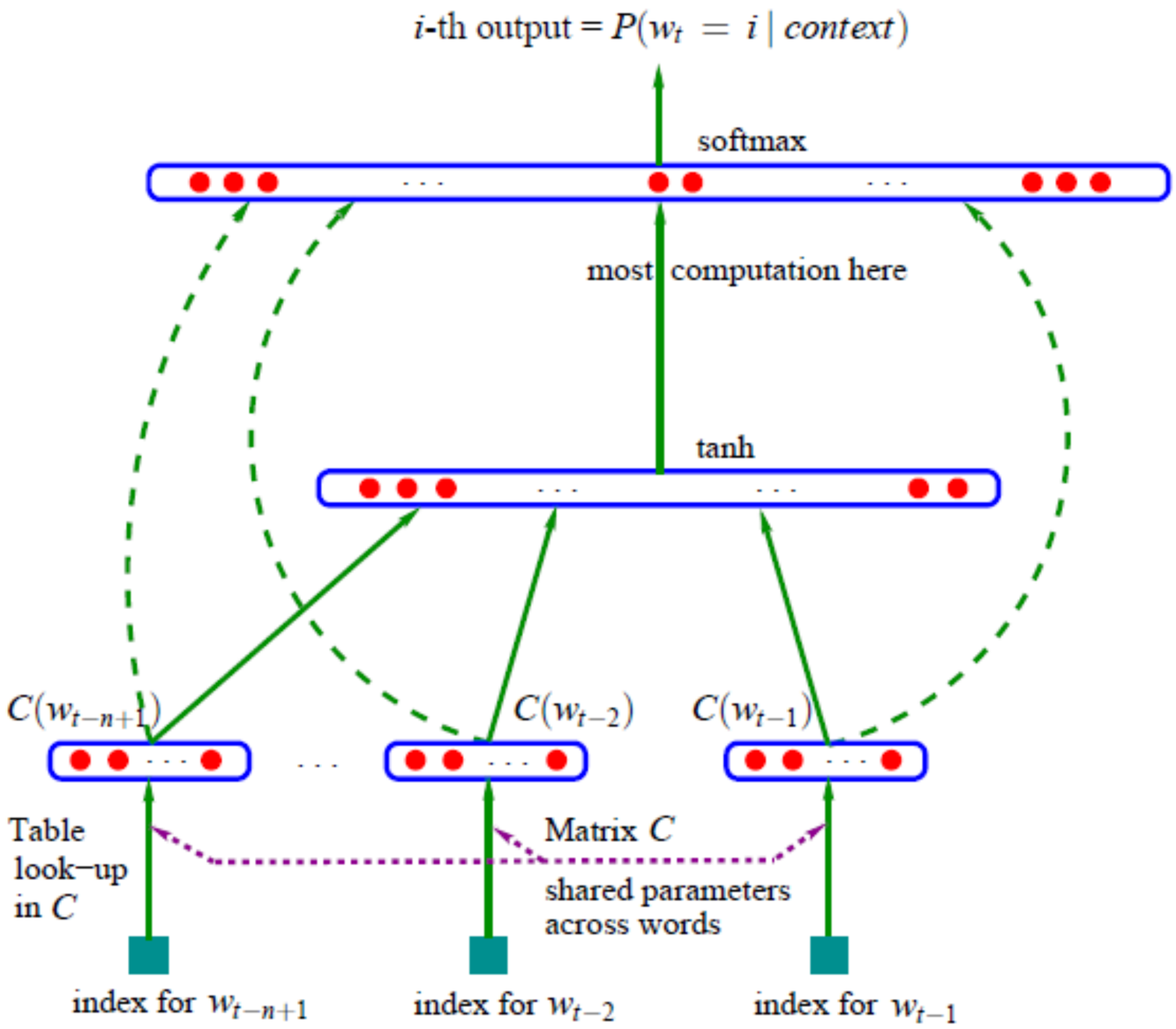
- These distributed representations in a continuous space are also referred to as "word embeddings"
  - Low dimensional
  - Similar words will have similar vectors

- Word embeddings capture semantic properties (such as *man* is to *woman* as *boy* is to *girl*, etc.) and morphological properties (*glad* is similar to *gladly*, etc.)

- The word embeddings could be learned via the first layer of a neural network [B03].

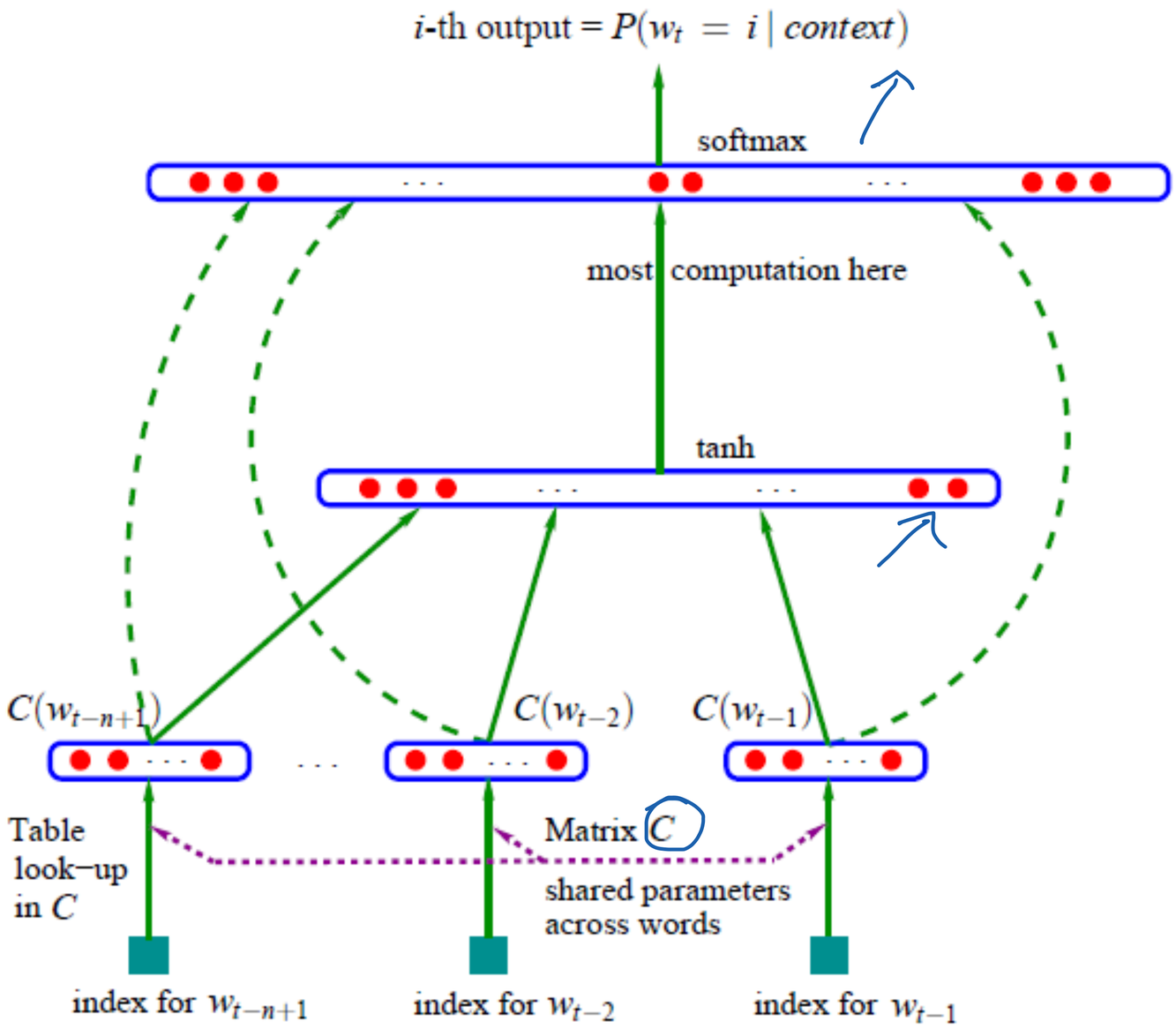
[B03]: Bengio et al., "A neural probabilistic LM", JMLR, 03

Word embeddings



[B03]: Bengio et al., “A neural probabilistic LM”, JMLR, 03

Word embeddings



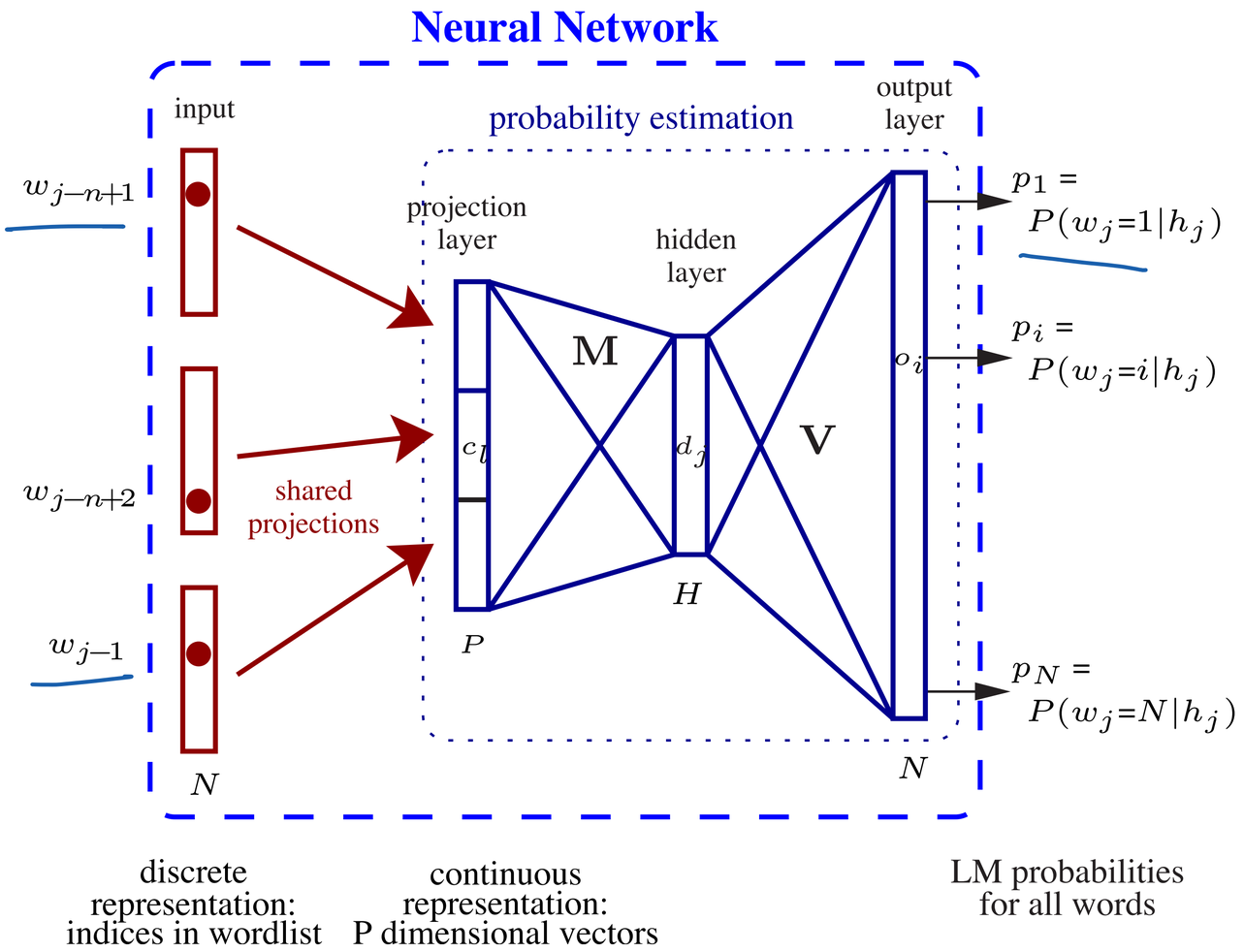
Introduced the architecture that forms the basis of all current neural language and word embedding models

- Embedding layer
- One or more middle/hidden layers
- Softmax output layer

[B03]: Bengio et al., “A neural probabilistic LM”, JMLR, 03

NN language model

- Project all the words of the context $h_j = w_{j-n+1},\ldots,w_{j-1}$ to their dense forms
- Then, calculate the language model probability $Pr(w_j = i | h_j)$ for the given context h_j



NN language model

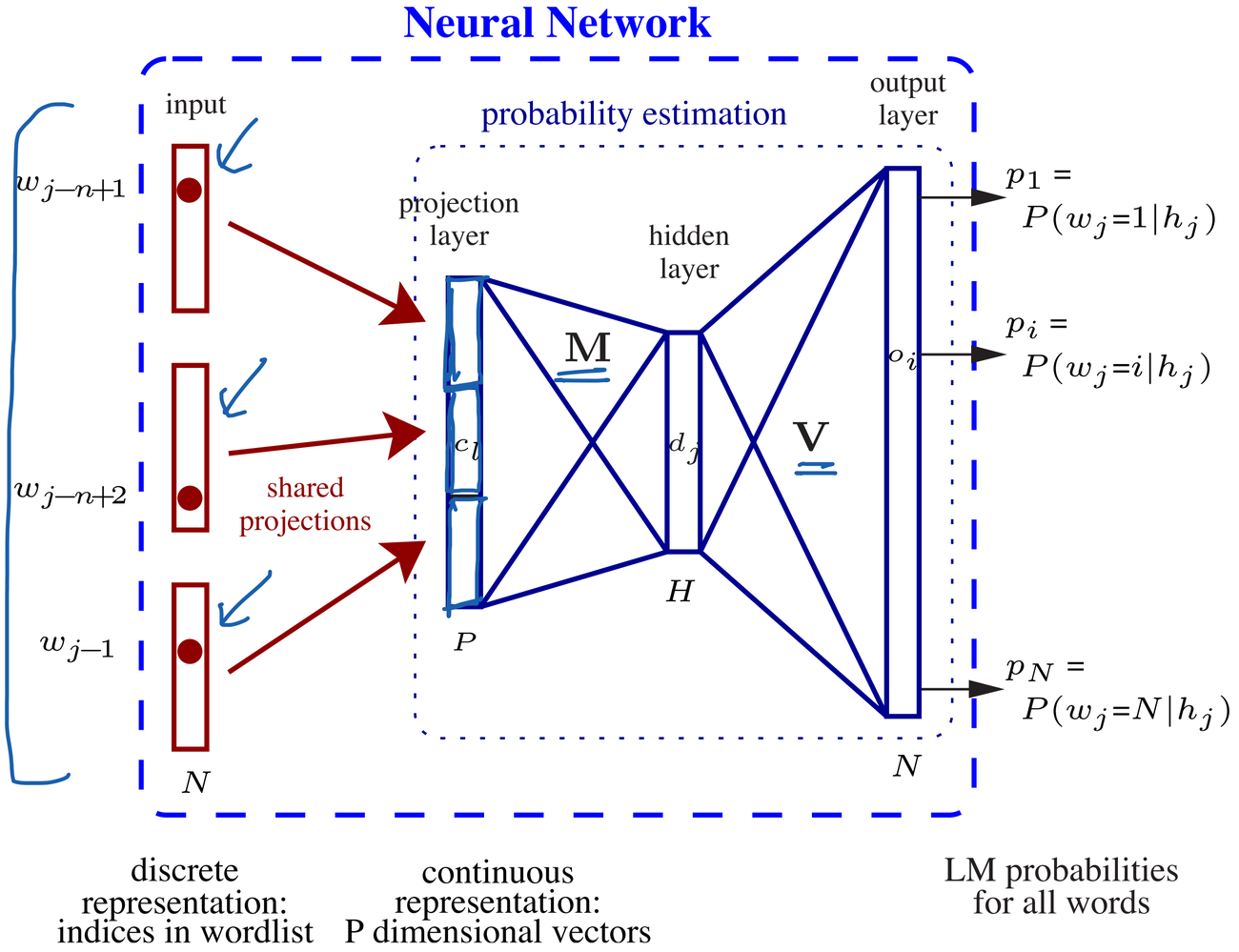
- Dense vectors of all the words in context are concatenated forming the first hidden layer of the neural network
- Second hidden layer:

$$d_j = \tanh\left(\sum_l m_{jl} c_l + b_j\right) \quad \forall j = 1, \ldots, H$$

- Output layer:

$$o_i = \sum_j v_{ij} d_j + b'_i \quad \forall i = 1, \ldots, N$$

- p_i → softmax output from the i^{th} neuron → $\Pr(w_j = i \mid h_j)$



NN language model

- Model is trained to minimise the following loss function:

$$L = \sum_{i=1}^{N} t_i \log p_i + \epsilon \left(\sum_{kl} m_{kl}^2 + \sum_{ik} v_{ik}^2 \right)$$

NN language model

- Model is trained to minimise the following loss function:

$$L = \sum_{i=1}^{N} t_i \log p_i + \epsilon \left(\sum_{kl} m_{kl}^2 + \sum_{ik} v_{ik}^2 \right)$$

NN language model

- Model is trained to minimise the following loss function:

$$L = \sum_{i=1}^{N} t_i \log p_i + \epsilon \left(\sum_{kl} m_{kl}^2 + \sum_{ik} v_{ik}^2 \right)$$

- Here, t_i is the target output 1-hot vector (1 for next word in the training instance, 0 elsewhere)

NN language model

- Model is trained to minimise the following loss function:

$$L = \sum_{i=1}^{N} t_i \log p_i + \epsilon \left(\sum_{kl} m_{kl}^2 + \sum_{ik} v_{ik}^2 \right)$$

- Here, t_i is the target output 1-hot vector (1 for next word in the training instance, 0 elsewhere)

- First part: Cross-entropy between the target distribution and the distribution estimated by the NN

NN language model

- Model is trained to minimise the following loss function:

$$L = \sum_{i=1}^{N} t_i \log p_i + \epsilon \left(\sum_{kl} m_{kl}^2 + \sum_{ik} v_{ik}^2 \right)$$

- Here, t_i is the target output 1-hot vector (1 for next word in the training instance, 0 elsewhere)

- First part: Cross-entropy between the target distribution and the distribution estimated by the NN

- Second part: Regularization term

Longer word context?

- What have we seen so far: A feedforward NN used to compute an Ngram probability $Pr(w_j = i|h_j)$ (where h_j encodes the Ngram history)

Longer word context?

- What have we seen so far: A feedforward NN used to compute an Ngram probability $Pr(w_j = i|h_j)$ (where h_j encodes the Ngram history)

- We know Ngrams are limiting:
  Alice who had attempted *the assignment asked* the lecturer

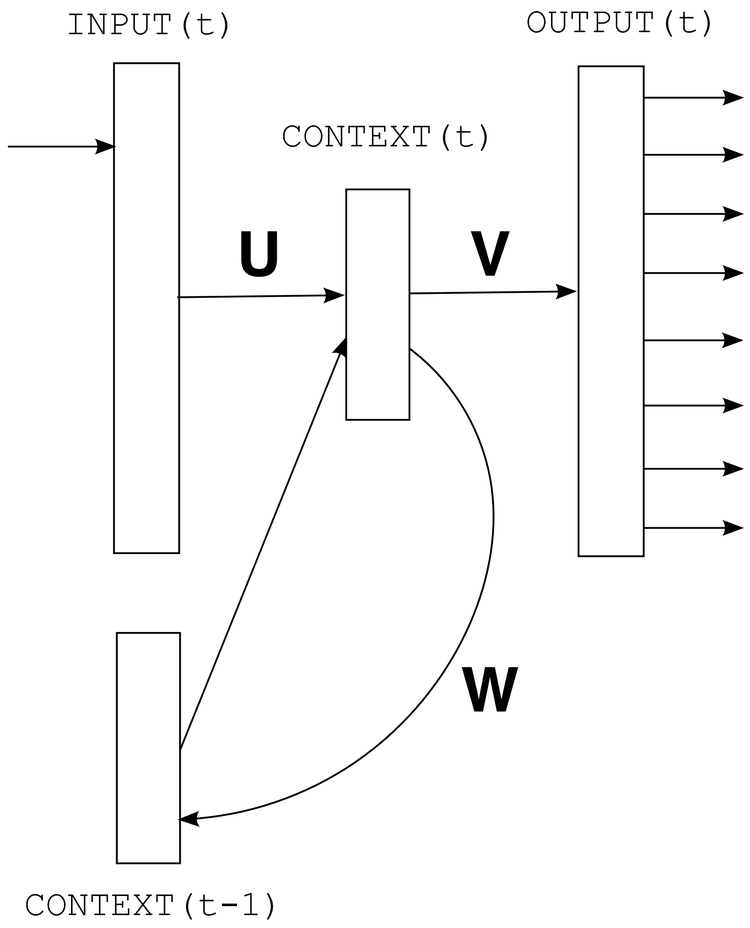
Longer word context?

- What have we seen so far: A feedforward NN used to compute an Ngram probability $Pr(w_j = i|h_j)$ (where h_j encodes the Ngram history)

- We know Ngrams are limiting:
  Alice who had attempted *the assignment asked* the lecturer

- How can we predict the next word based on the entire sequence of preceding words? Use recurrent neural networks (RNNs)

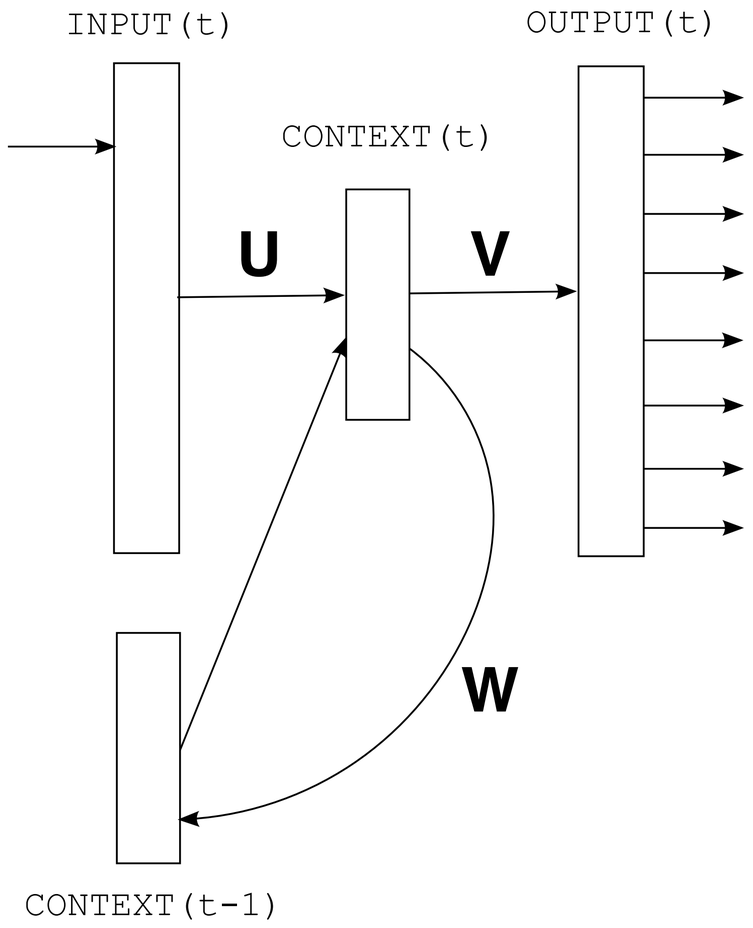
Simple RNN language model



- Current word, x_t
  Hidden state, s_t
  Output, y_t

Image from: Mikolov et al., “Recurrent neural network based language model”, Interspeech 10

Simple RNN language model

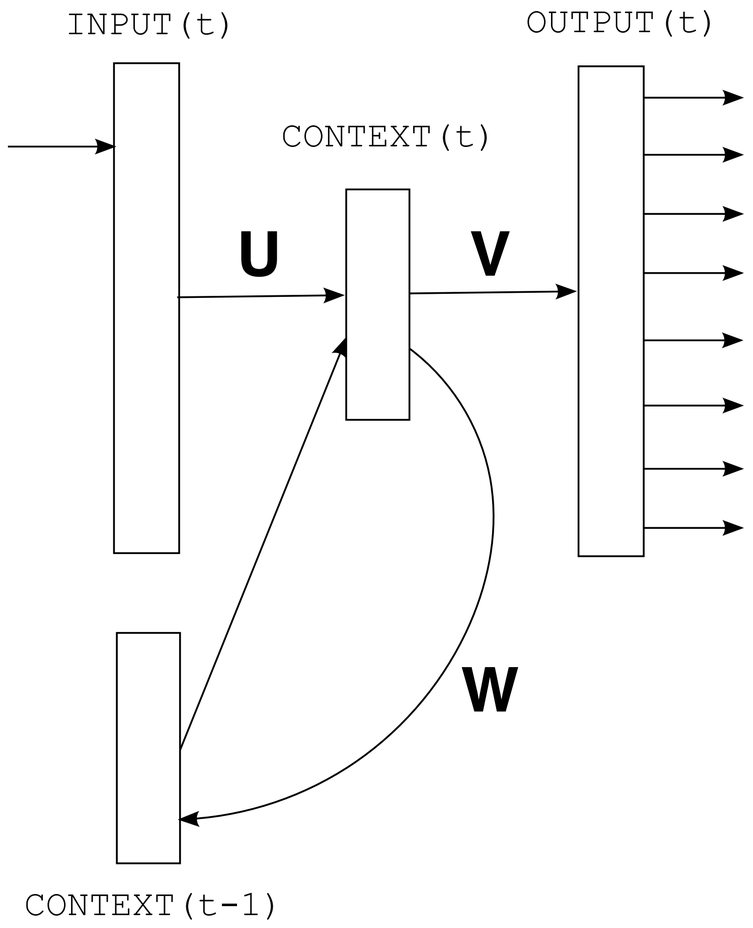


- Current word, x_t
  Hidden state, s_t
  Output, y_t

$$s_t = f(Ux_t + Ws_{t-1})$$
$$o_t = \mathrm{softmax}(Vs_t)$$

Image from: Mikolov et al., “Recurrent neural network based language model”, Interspeech 10

Simple RNN language model



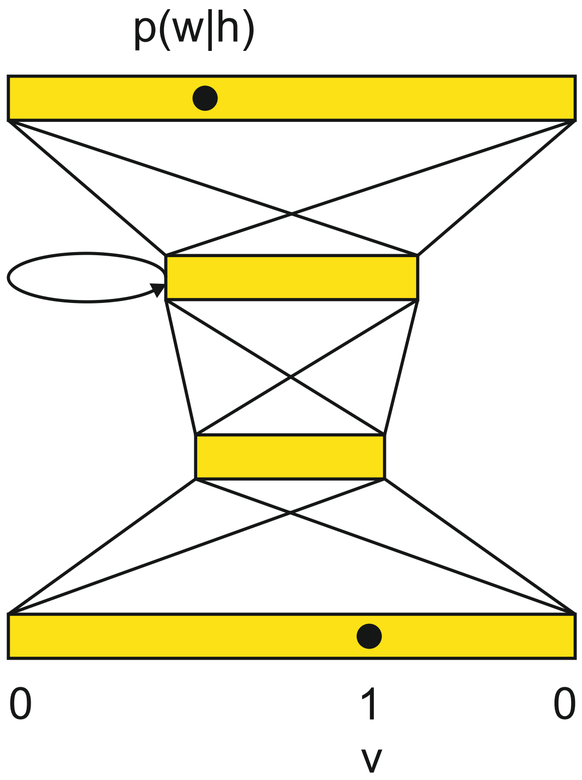
- Current word, x_t
  Hidden state, s_t
  Output, y_t

$$s_t = f(Ux_t + Ws_{t-1})$$
$$o_t = \mathrm{softmax}(Vs_t)$$

- RNN is trained using the cross-entropy criterion

Image from: Mikolov et al., “Recurrent neural network based language model”, Interspeech 10

LSTM-LMs



- Vanilla RNN-LMs unlikely to show full potential of recurrent models due to issues like vanishing gradients
- LSTM-LMs: Similar to RNN-LMs except use LSTM units in the 2nd hidden (recurrent) layer

Image from: Sundermeyer et al., “LSTM NNs for Language Modeling”, IS 10

Comparing RNN-LMs with LSTM-LMs

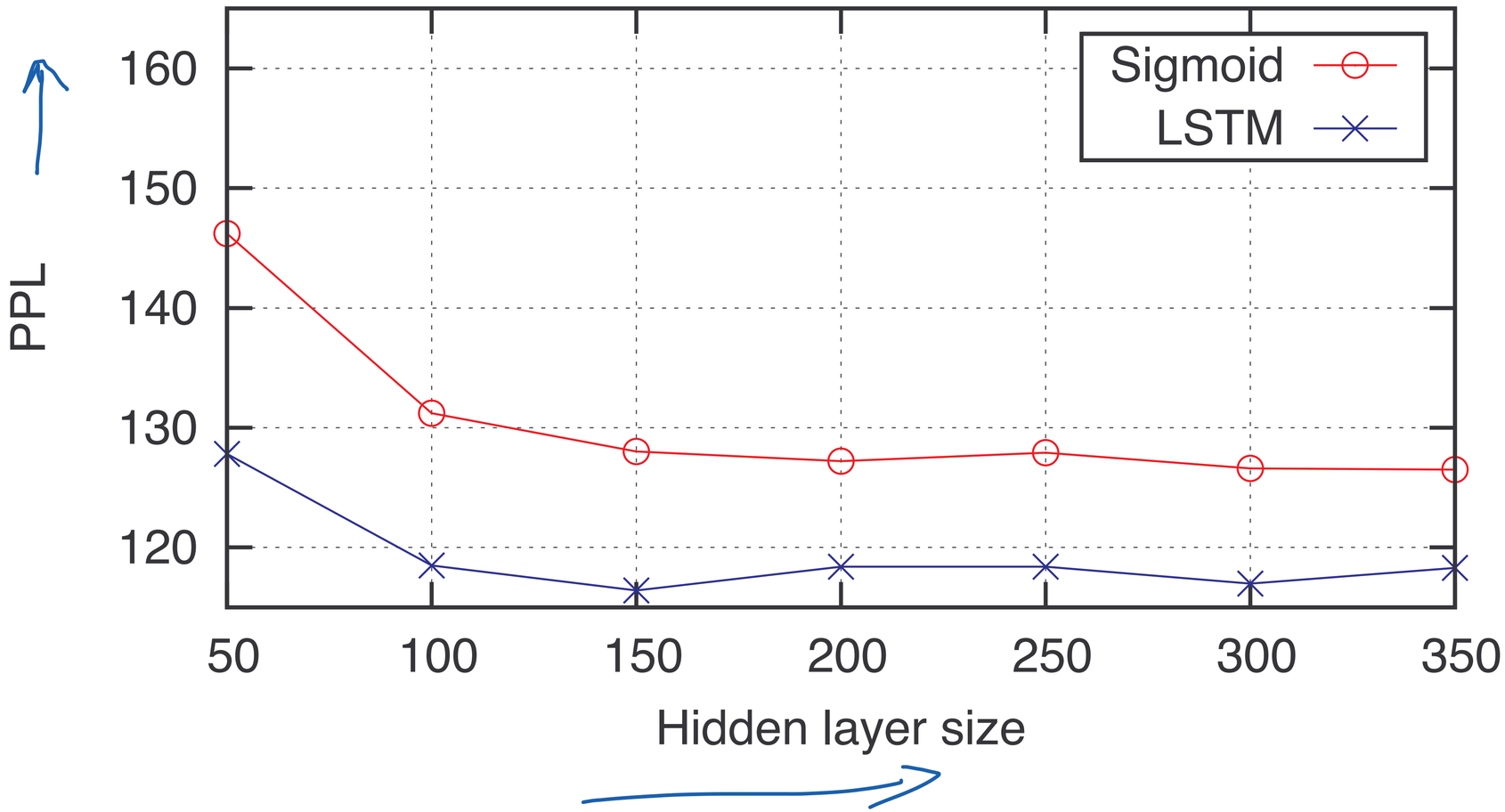


Image from: Sundermeyer et al., “LSTM NNs for Language Modeling”, 10