Automatic Speech Recognition (CS753)

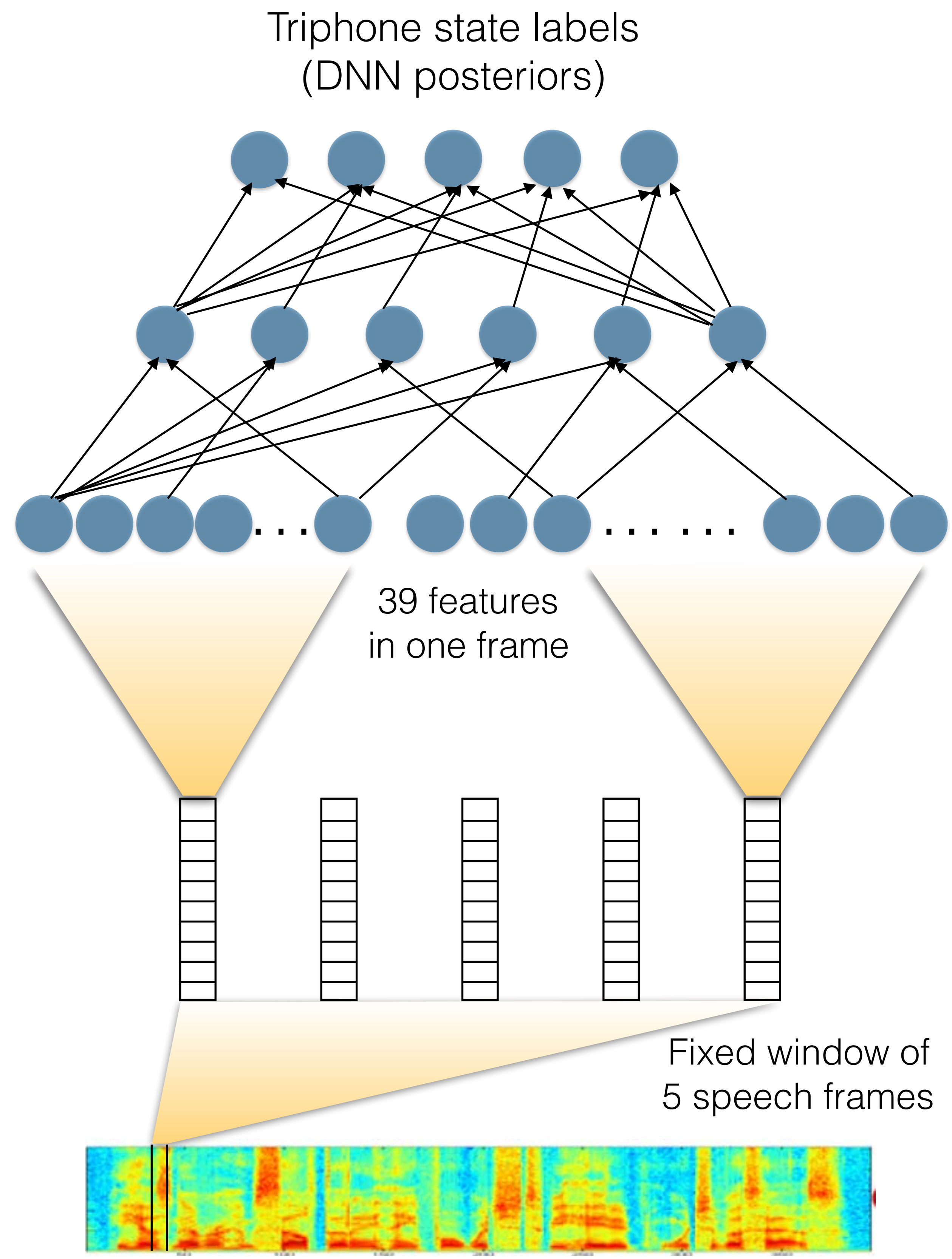
Lecture 11: Recurrent Neural Network (RNN) Models for ASR

Instructor: Preethi Jyothi

Feb 9, 2017

Recap: Hybrid DNN-HMM Systems

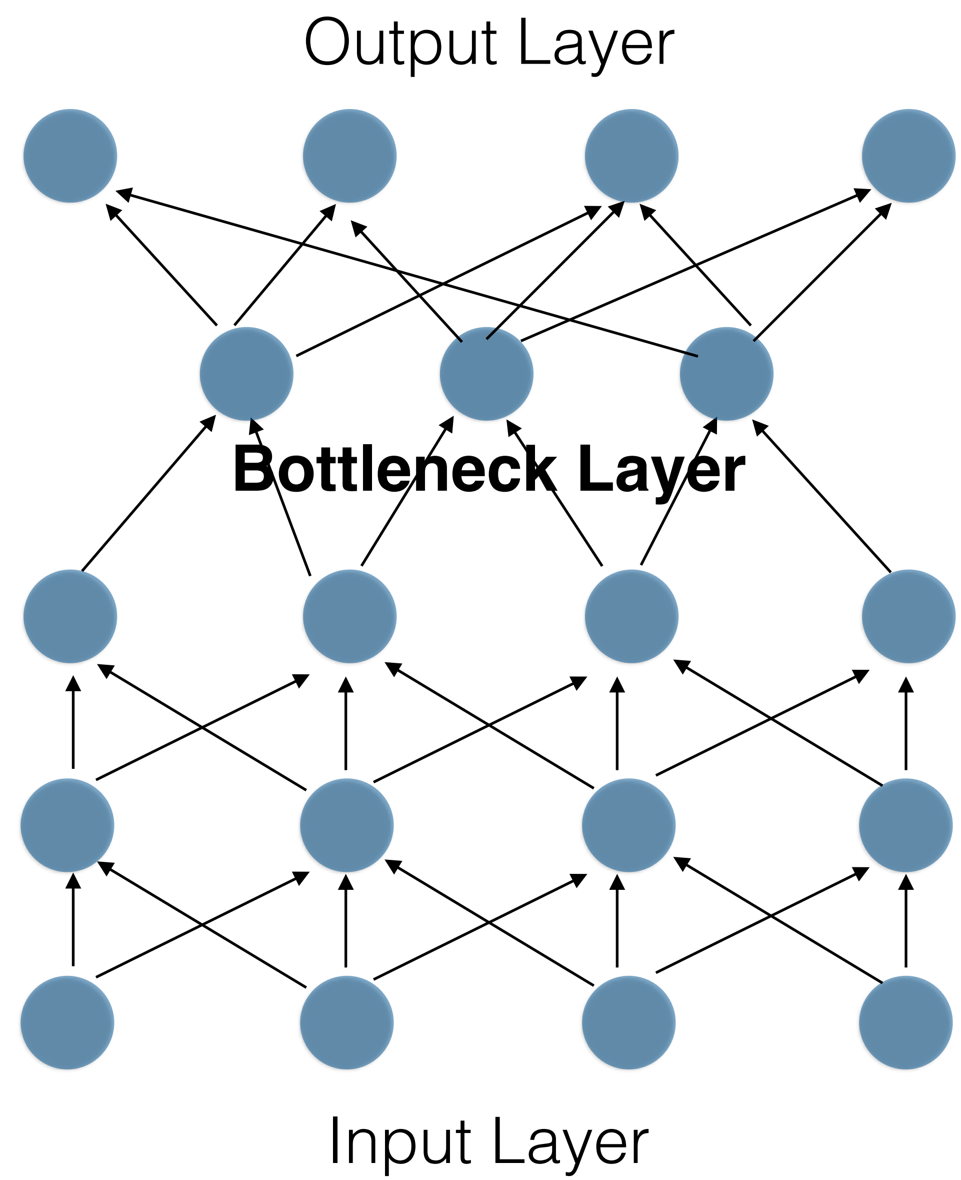
- Instead of GMMs, use scaled DNN posteriors as the HMM observation probabilities
- DNN trained using triphone labels derived from a forced alignment "Viterbi" step.
  - *Forced alignment:* Given a training utterance $\{O,W\}$, find the most likely sequence of states (and hence triphone state labels)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



Recap: Tandem DNN-HMM Systems

- Neural network outputs are used as “features” to train HMM-GMM models

- Use a low-dimensional *bottleneck* layer representation to extract features from the bottleneck layer



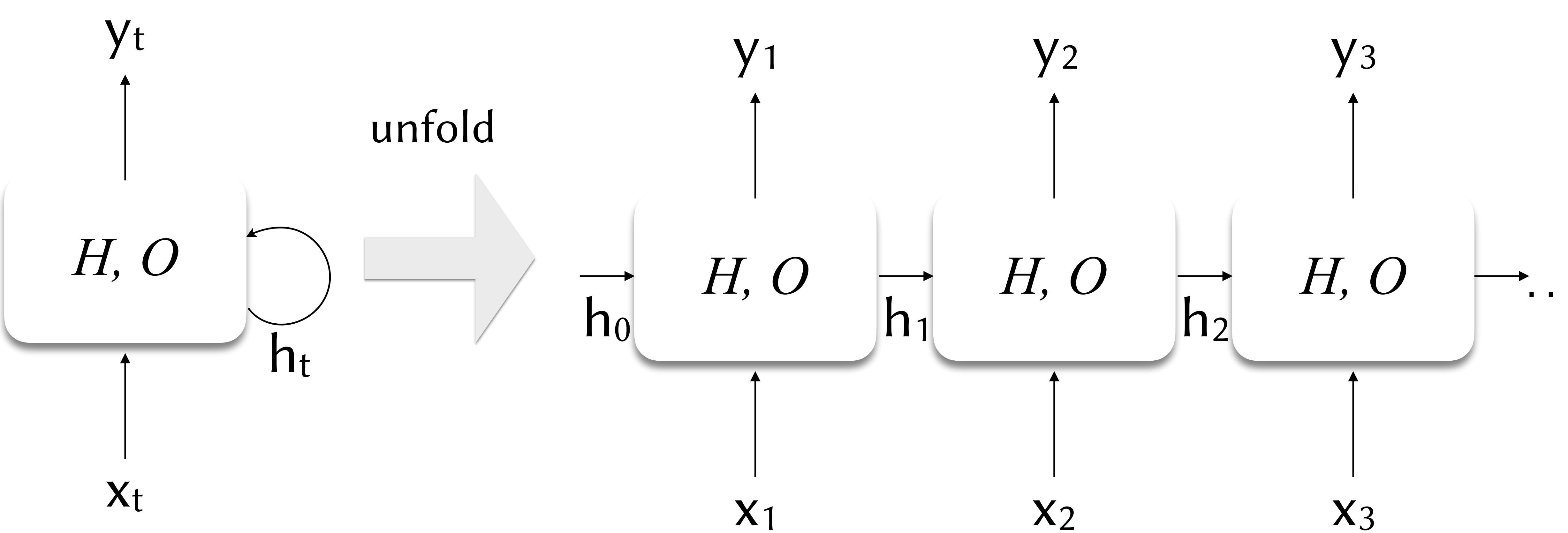
Feedforward DNNs we've seen so far…

- 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

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



Two main equations govern RNNs:

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

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

where W, V, U are matrices of input-hidden weights, hidden-hidden weights and hidden-output weights resp; $b^{(y)}$ and $b^{(y)}$ are bias vectors

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?

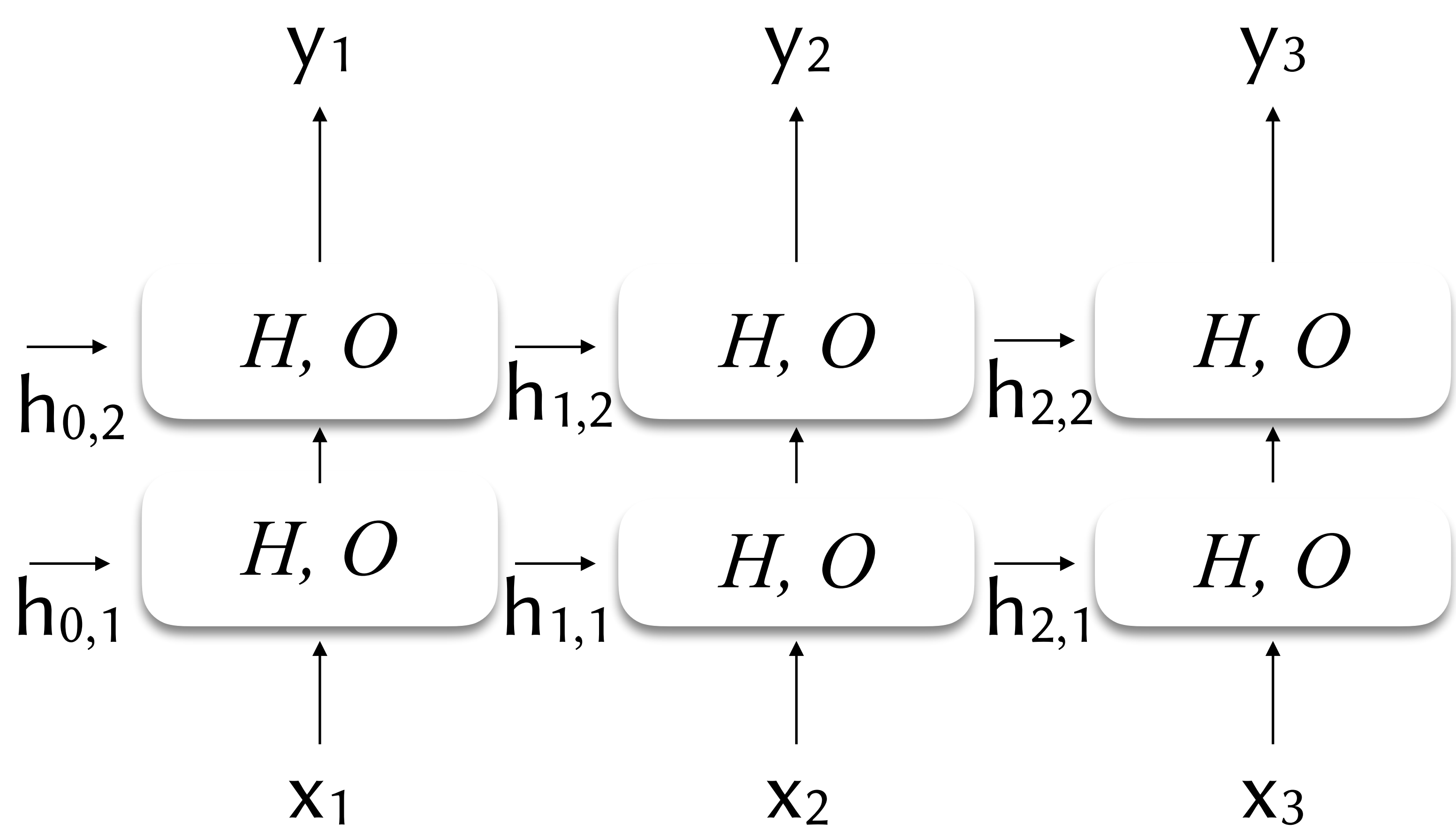
    - HMMs are limited by the size of the state space. Inference becomes intractable if the state space grows very large!

    - 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

- 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

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

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

H : element wise application of the sigmoid or tanh function

O : the softmax function

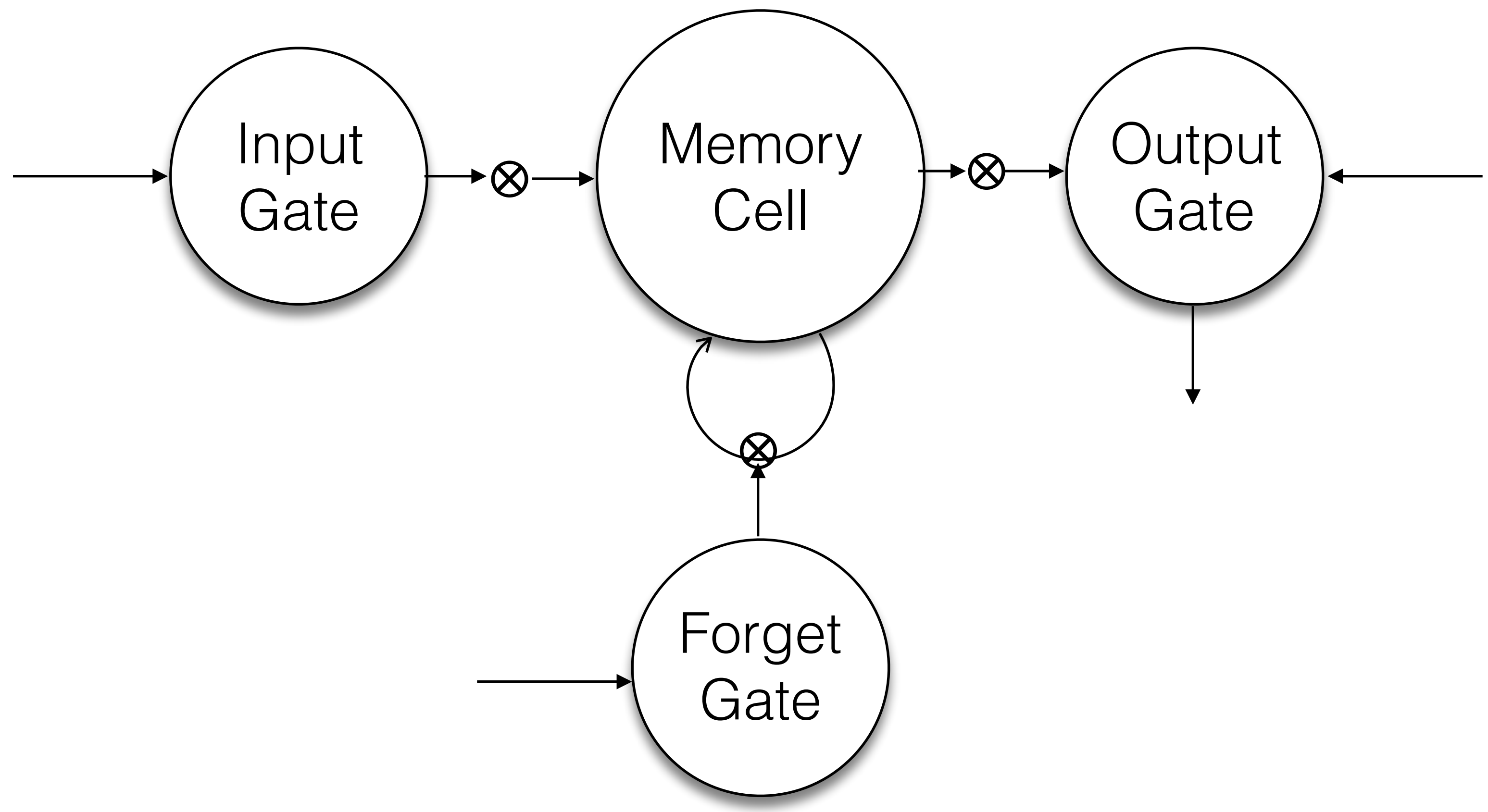
Run into problems of exploding and vanishing gradients.

Exploding/Vanishing Gradients

- In deep networks, gradients in early layers is 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]

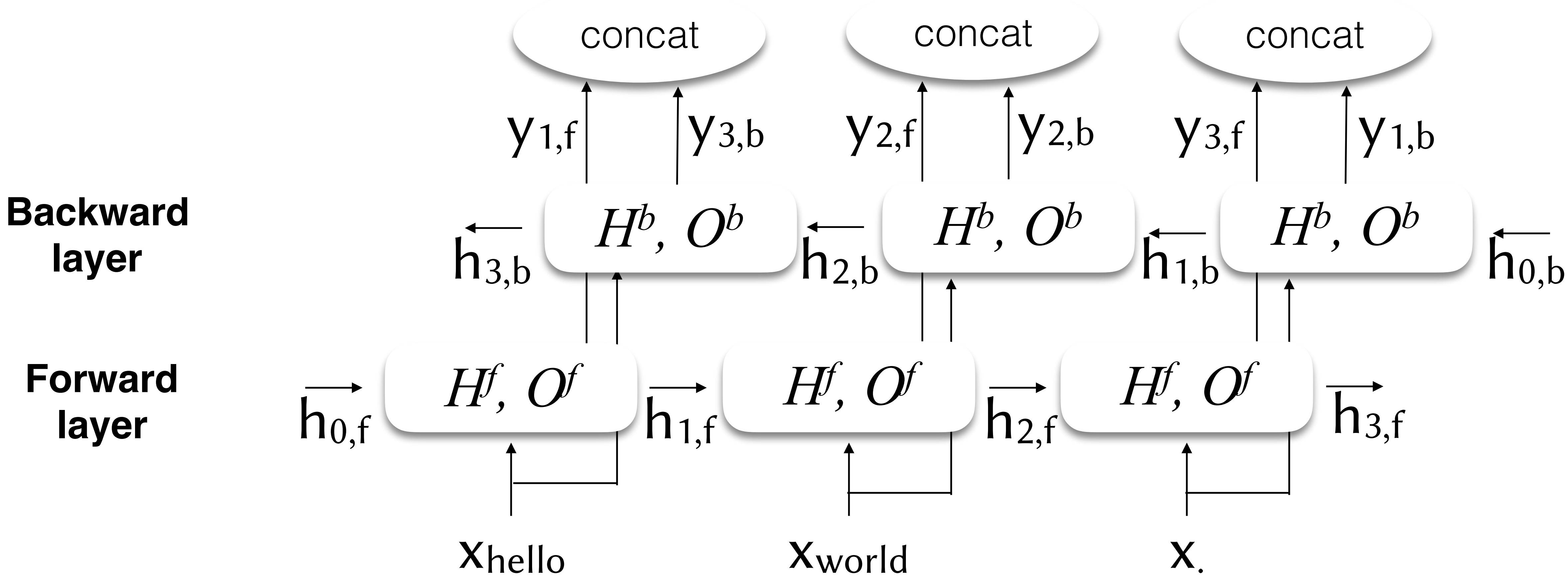
[HS97] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

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

RNN-based ASR system

CS 753
Feb 9, 2017

ASR with RNNs

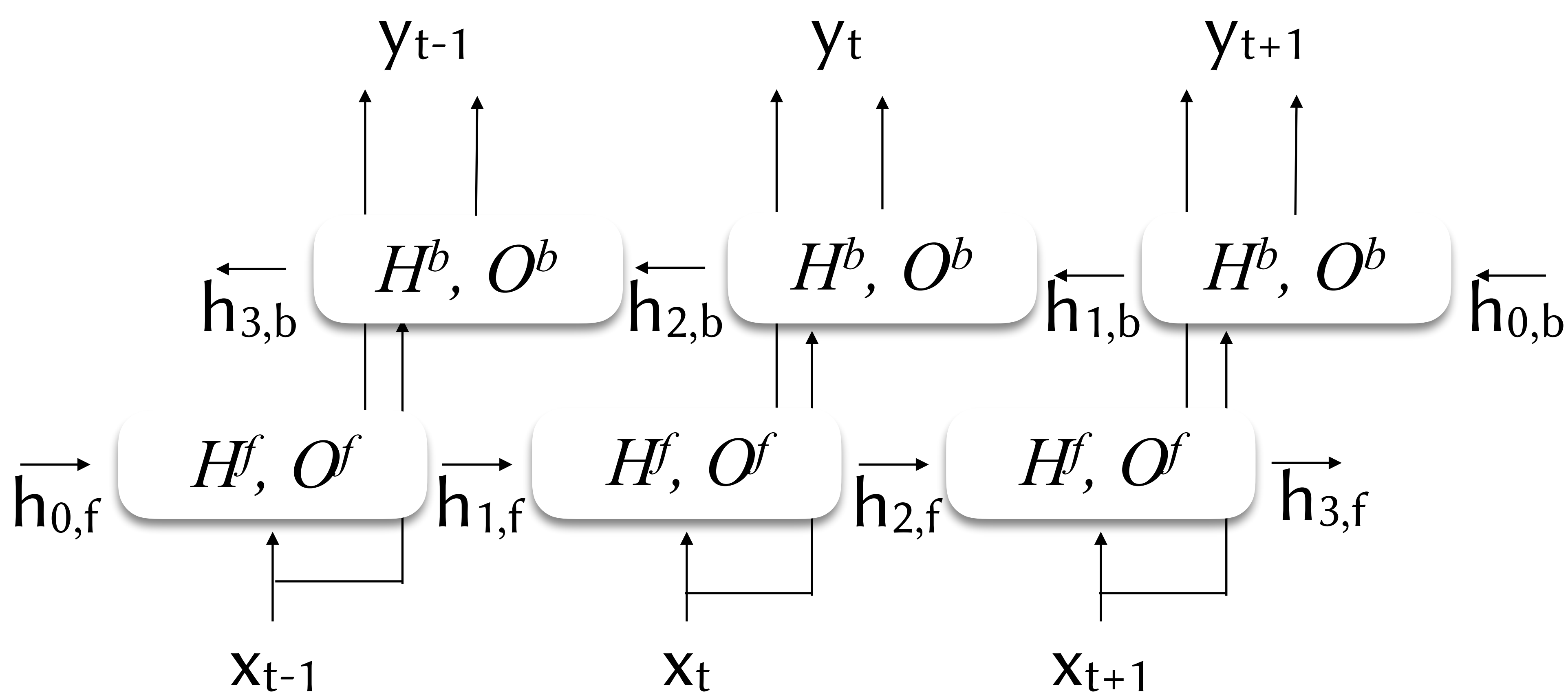
- Neural networks in ASR systems are typically a single component (aka acoustic models) in a complex pipeline

- Limitations:
    1. Frame-level training targets derived from HMM-based alignments
    2. Objective function optimized in NNs is very different from the final evaluation metric

- Goal: Single RNN model that addresses these issues and replaces as much of the speech pipeline as possible [G14]

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

RNN Architecture



- H was implemented using LSTMs in [G14]. Input: Acoustic feature vectors, one per frame; Output: Characters + space
- Deep bidirectional LSTM networks were used

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

Connectionist Temporal Classification (CTC)

$$\Pr(y|x) = \sum_{a \in \mathcal{B}^-1(y)} \Pr(a|x) \;\; \text{where} \;\; \Pr(a|x) = \prod_{t=1}^{T} \Pr(a_t, t|x) \qquad \text{... (1)}$$

$$\text{For a target } y^*, \;\; \mathrm{CTC}(x) = -\log \Pr(y^*|x) \qquad \text{... (2)}$$

$$\mathcal{L}(x) = \sum_{y} \Pr(y|x)\mathcal{L}(x,y) = \sum_{a} \Pr(a|x)\mathcal{L}(x,\mathcal{B}(a)) \qquad \text{... (3)}$$

- For an input sequence x of length T, Eqn (1) gives the probability of an output transcription y; a is a CTC alignment of y
- Given a target transcription y^*, the CTC objective function to be minimised is given in Eqn (2)
- Modify loss function as shown in Eqn (3) to be a better match to the final test criteria; here, $\mathcal{L}(x, y)$ is a transcription loss function
- $\mathcal{L}(x)$ needs to be minimised: Use a Monte-carlo sampling-based algorithm

Decoding

- First approximation: For a given test input sequence x, pick the most probable output at each time step

$$\arg\max_y \Pr(y|x) \approx \mathcal{B}(\arg\max_a \Pr(a|x))$$

- More accurate decoding uses a search algorithm that also makes use of a dictionary and a language model. (Decoding search algorithms will be discussed in detail in later lectures.)

WER results

System	LM	WER
RNN-CTC	Dictionary only	24.0
RNN-CTC	Bigram	10.4
RNN-CTC	Trigram	8.7
RNN-WER	Dictionary only	21.9
RNN-WER	Bigram	9.8
RNN-WER	Trigram	8.2
Baseline	Bigram	9.4
Baseline	Trigram	7.8

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

Some erroneous examples produced by the end-to-end RNN

Target: “There’s unrest but we’re not going to lose them to Dukakis”

Output: “There’s unrest but we’re not going to lose them to *Dekakis*”

Target: “T. W. A. also plans to hang its boutique shingle in airports at Lambert Saint”

Output: “T. W. A. also plans *tohing* its *bootik single* in airports at Lambert Saint”