Automatic Speech Recognition (CS753)

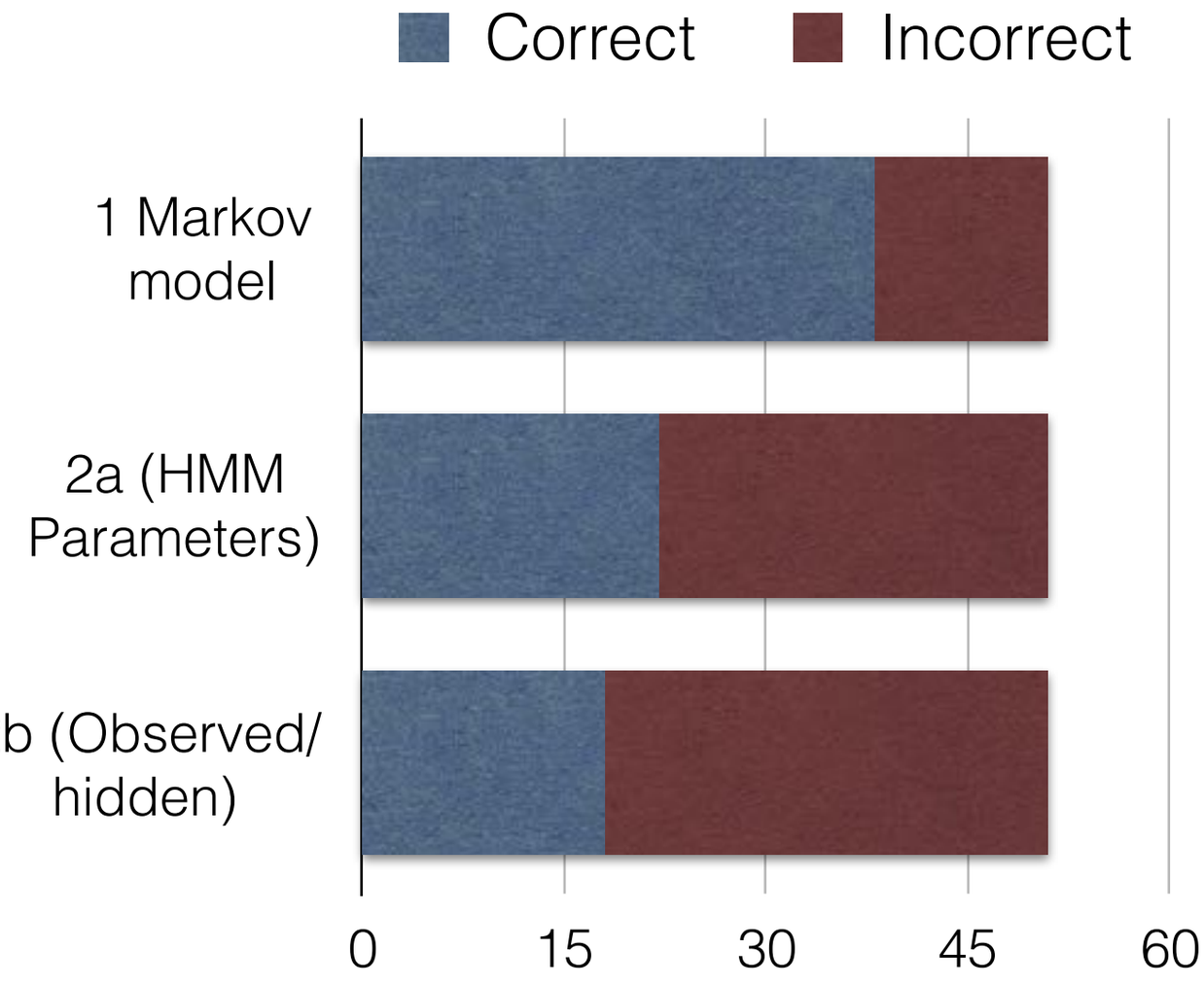
Lecture 10: Deep Neural Network(DNN)-based Acoustic Models

Instructor: Preethi Jyothi

Feb 6, 2017

Quiz 2 Postmortem

- Common Mistakes:
  - 2(a) Omitting mixture weights from parameters
  - 2(b) Mistaking parameters for hidden/ observed variables



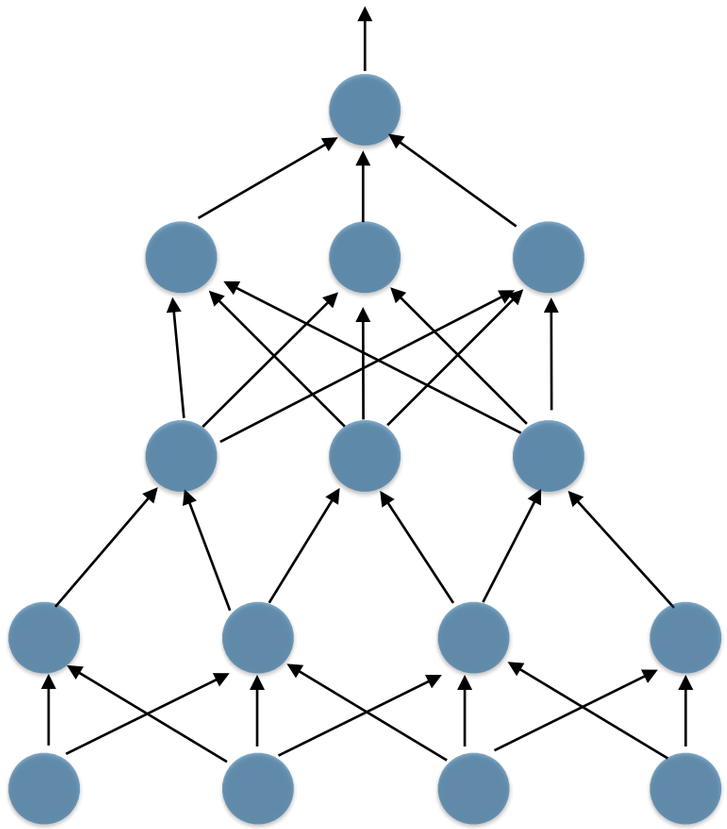
Preferred order of topics to be revised:

HMMs — Tied state triphones,
HMMs — Training (EM/Baum-Welch)
WFSTs in ASR systems
HMMs — Decoding (Viterbi)

Recap: Feedforward Neural Networks

- Input layer, zero or more hidden layers and an output layer

- Nodes in hidden layers compute non-linear (activation) functions of a linear combination of the inputs



- Common activation functions include sigmoid, tanh, ReLU, etc.

- NN outputs typically normalised by applying a softmax function to the output layer

$$\mathrm{softmax}(x_1, \ldots, x_k) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$$

Recap: Training Neural Networks

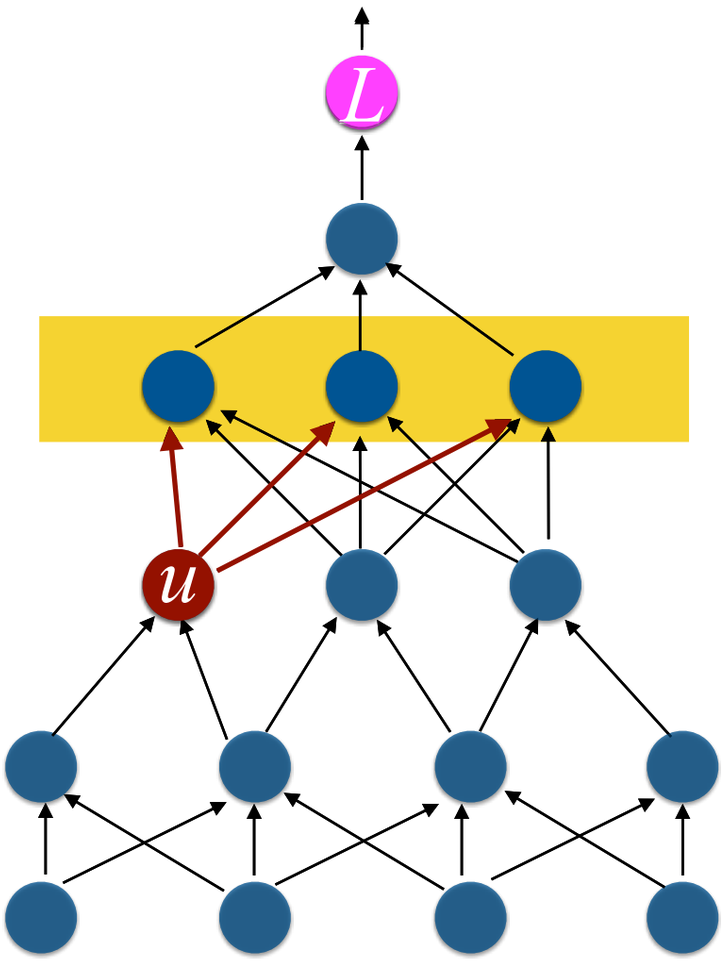
- NNs optimized to minimize a loss function, L, that is a score of the network's performance (e.g. squared error, cross entropy, etc.)

- To minimize L, use (mini-batch) stochastic gradient descent

- Need to efficiently compute $\partial L/\partial w$ (and hence $\partial L/\partial u$) for all w

- Use **backpropagation** to compute $\partial L/\partial u$ for every node u in the network

- Key fact backpropagation is based on: Chain rule of differentiation



Neural Networks for ASR

- Two main categories of approaches have been explored:

1. Hybrid neural network-HMM systems: Use NNs to estimate HMM observation probabilities

2. Tandem system: NNs used to generate input features that are fed to an HMM-GMM acoustic model

Neural Networks for ASR

- Two main categories of approaches have been explored:

1. Hybrid neural network-HMM systems: Use NNs to estimate HMM observation probabilities

2. Tandem system: NNs used to generate input features that are fed to an HMM-GMM acoustic model

Decoding an ASR system

- Recall how we decode the most likely word sequence W for an acoustic sequence O:

$$W^* = \arg\max_W \Pr(O|W) \Pr(W)$$

- The acoustic model $\Pr(O|W)$ can be further decomposed as (here, Q,M represent triphone, monophone sequences resp.):

$$\begin{aligned}
\Pr(O|W) &= \sum_{Q,M} \Pr(O, Q, M|W) \\
&= \sum_{Q,M} \Pr(O|Q, M, W) \Pr(Q|M, W) \Pr(M|W) \\
&\approx \sum_{Q,M} \Pr(O|Q) \Pr(Q|M) \Pr(M|W)
\end{aligned}$$

Hybrid system decoding

$$\Pr(O|W) \approx \sum_{Q,M} \Pr(O|Q) \Pr(Q|M) \Pr(M|W)$$

You've seen Pr($O|Q$) estimated using a Gaussian Mixture Model. Let's use a neural network instead to model Pr($O|Q$).

$$\Pr(O|Q) = \prod_t \Pr(o_t|q_t)$$

$$\Pr(o_t|q_t) = \frac{\Pr(q_t|o_t)\Pr(o_t)}{\Pr(q_t)}$$

$$\propto \frac{\Pr(q_t|o_t)}{\Pr(q_t)}$$

where o_t is the acoustic vector at time t and q_t is a triphone HMM state Here, Pr($q_t|o_t$) are posteriors from a trained neural network. Pr($o_t|q_t$) is then a scaled posterior.

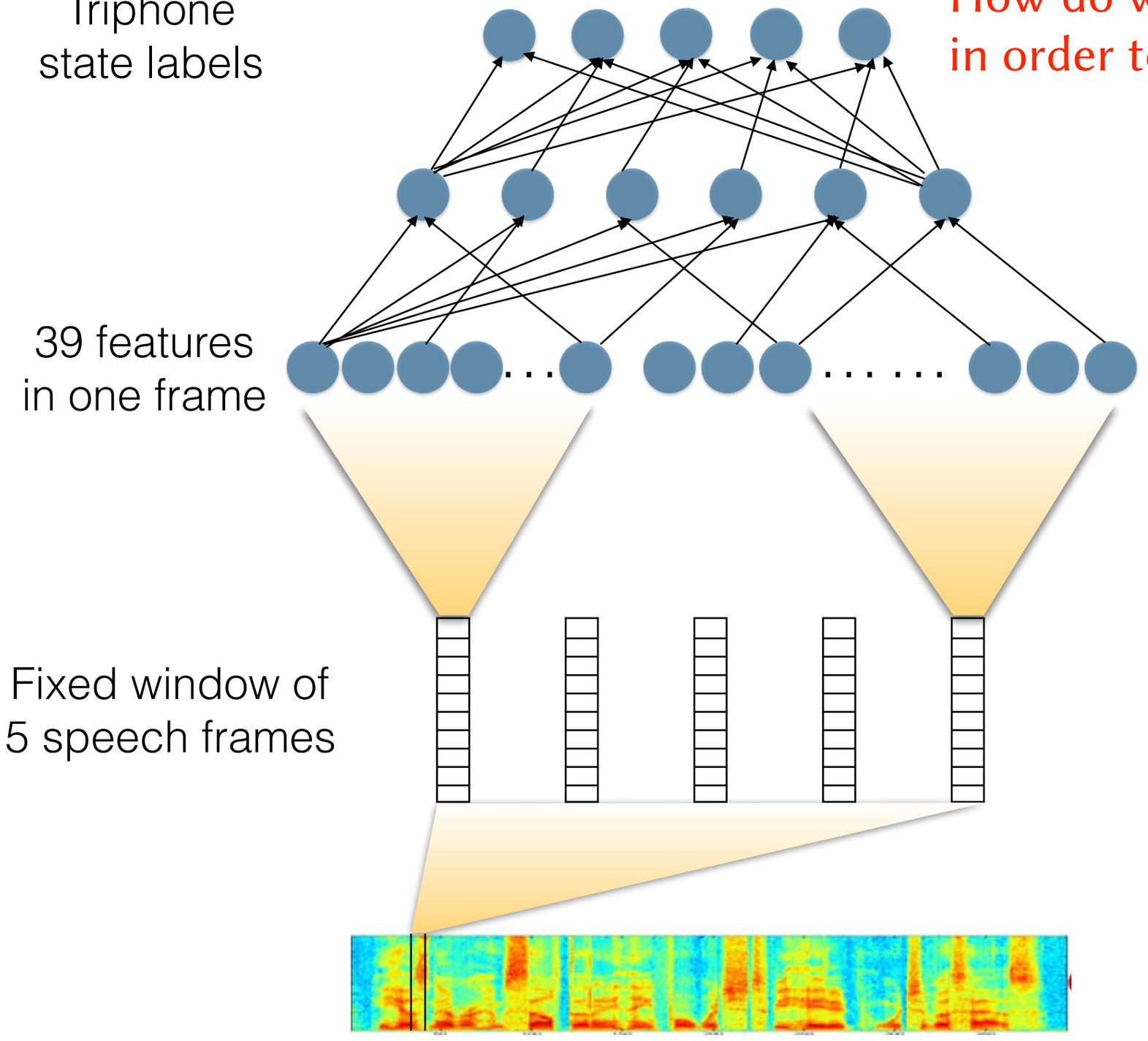
Computing $\Pr(q_t|o_t)$ using a deep NN

Triphone state labels

How do we get these labels in order to train the NN?

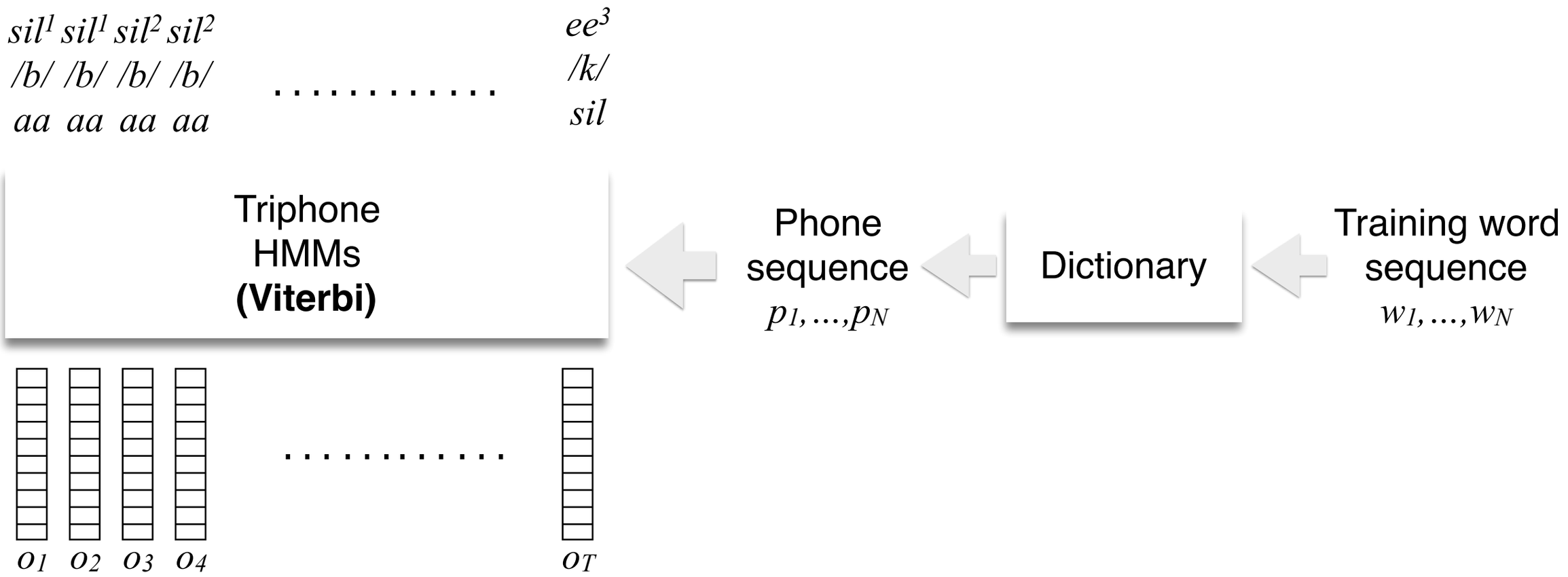
39 features in one frame

Fixed window of 5 speech frames



Triphone labels

- Forced alignment: Use current acoustic model to find the most likely sequence of HMM states given a sequence of acoustic vectors. (Algorithm to help compute this?)
- The "Viterbi paths" for the training data is referred to as forced alignment

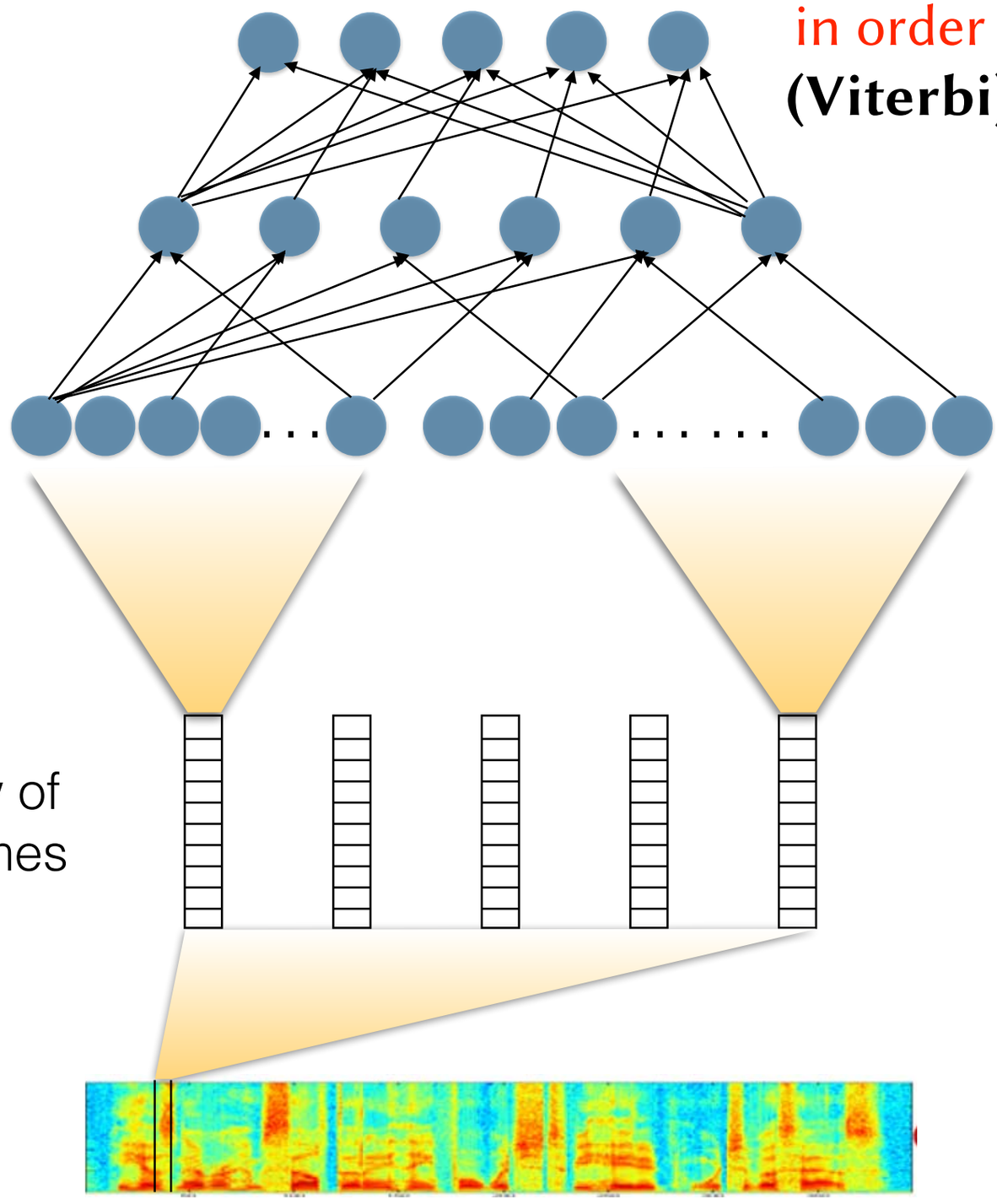


Computing $\Pr(q_t|o_t)$ using a deep NN

Triphone state labels

39 features in one frame

Fixed window of 5 speech frames



How do we get these labels in order to train the NN?

(Viterbi) Forced alignment

Computing priors Pr(q_t)

- To compute HMM observation probabilities, Pr($o_t|q_t$), we need both Pr($q_t|o_t$) and Pr(q_t)

- The posterior probabilities Pr($q_t|o_t$) are computed using a trained neural network

- Pr(q_t) are relative frequencies of each triphone state as determined by the forced Viterbi alignment of the training data

Hybrid Networks

- The hybrid networks are trained with a minimum cross-entropy criterion

$$L(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i)$$

- Advantages of hybrid systems:

    1. No assumptions made about acoustic vectors being uncorrelated: Multiple inputs used from a window of time steps

    2. Discriminative objective function

Neural Networks for ASR

- Two main categories of approaches have been explored:

1. Hybrid neural network-HMM systems: Use NNs to estimate HMM observation probabilities

2. Tandem system: NNs used to generate input features that are fed to an HMM-GMM acoustic model

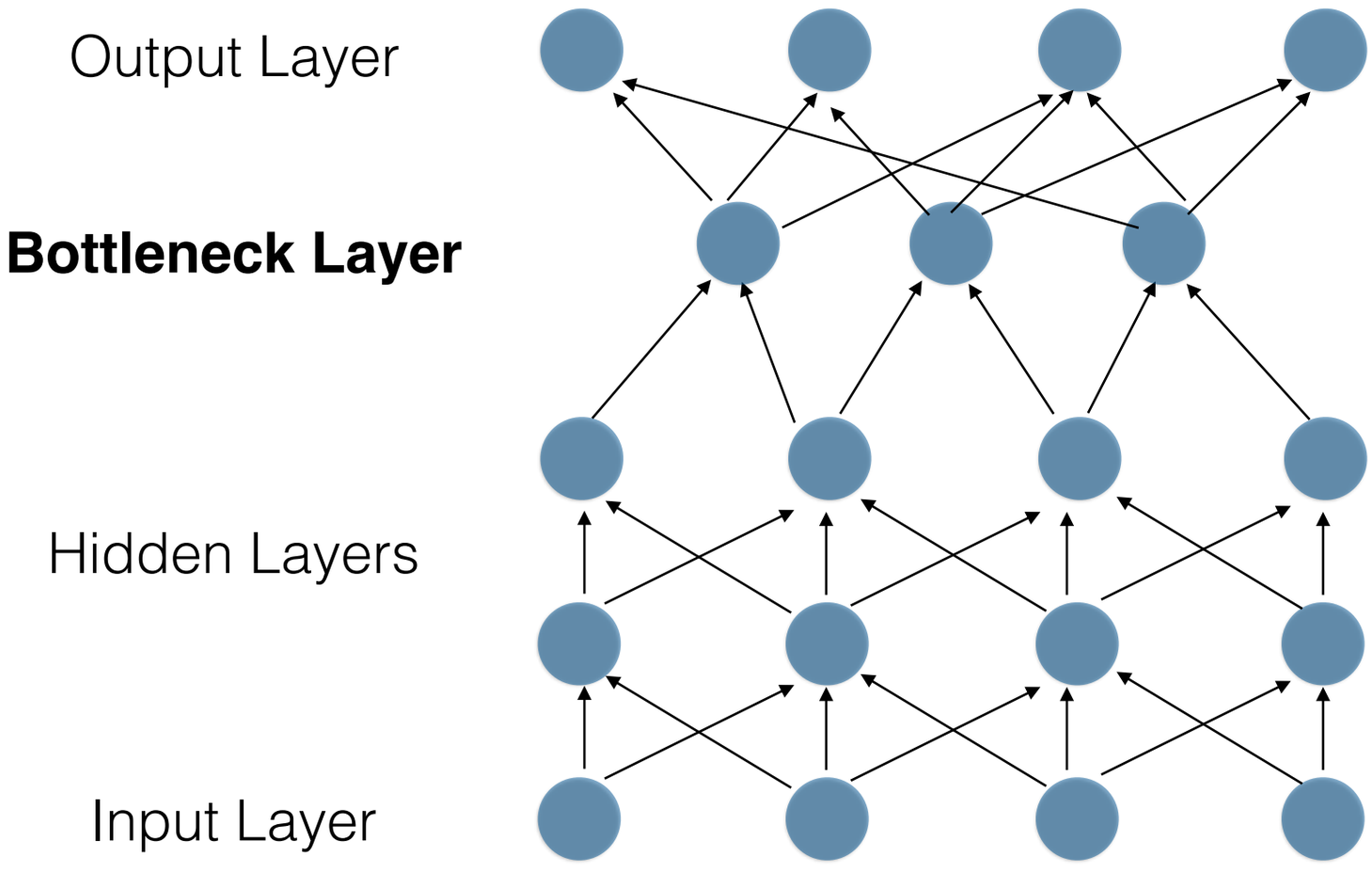
Tandem system

- First, train an NN to estimate the posterior probabilities of each subword unit (monophone, triphone state, etc.)

- In a hybrid system, these posteriors (after scaling) would be used as observation probabilities for the HMM acoustic models

- In the tandem system, the NN outputs are used as “feature” inputs to HMM-GMM models

Bottleneck Features



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

These bottleneck features are in turn used as inputs to HMM-GMM models

History of Neural Networks in ASR

- Neural networks for speech recognition were explored as early as 1987

- Deep neural networks for speech
  - Beat state-of-the-art on the TIMIT corpus [M09]
  - Significant improvements shown on large-vocabulary systems [D11]
  - Dominant ASR paradigm [H12]

[M09] A. Mohamed, G. Dahl, and G. Hinton, “Deep belief networks for phone recognition,” NIPS Workshop on Deep Learning for Speech Recognition, 2009.
[D11] G. Dahl, D. Yu, L. Deng, and A. Acero, “Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition,” TASL 20(1), pp. 30–42, 2012.
[H12] G. Hinton, et al., “Deep Neural Networks for Acoustic Modeling in Speech Recognition”, IEEE Signal Processing Magazine, 2012.

What's new?

- Hybrid systems were introduced in the late 80s. Why have NN-based systems come back to prominence?

- Important developments
  - Vast quantities of data available for ASR training
  - Fast GPU-based training
  - Improvements in optimization/initialization techniques
  - Deeper networks enabled by fast training
  - Larger output spaces enabled by fast training and availability of data

Pretraining

- Use unlabelled data to find good regions of the weight space that will help model the distribution of inputs

- Generative pretraining:

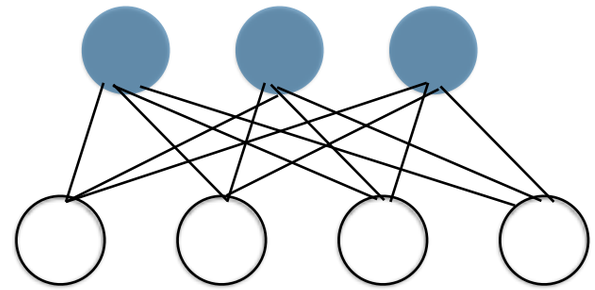
  ➡ Learn layers of feature detectors one at a time with states of feature detector in one layer acting as observed data for training the next layer.

  ➡ Provides better initialisation for a discriminative “fine-tuning phase” that uses backpropagation to adjust the weights from the “pretraining phase”

Pretraining contd.

- Learn a single layer of feature detectors by fitting a generative model to the input data: Use Restricted Boltzmann Machines (RBMs) [H02]

- An RBM is an undirected model: layer of visible units connected to a layer of hidden units, but no intra-visible or intra-hidden unit connections



$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{a}\mathbf{v} - \mathbf{b}\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v}$$

where $\mathbf{a}$, $\mathbf{b}$ are biases of the visible, hidden units and $\mathbf{W}$ is the weight matrix between the layers

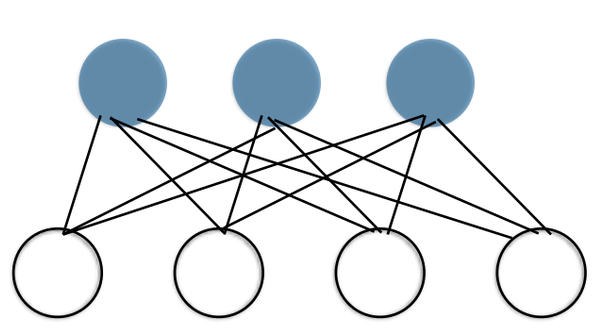
---

[H02] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," Neural Comput., 14, 1771–1800, '02.

Pretraining contd.

- Learn the weights and biases of the RBM to minimise the empirical negative log-likelihood of the training data

- How? Use an efficient learning algorithm called contrastive divergence [H02]



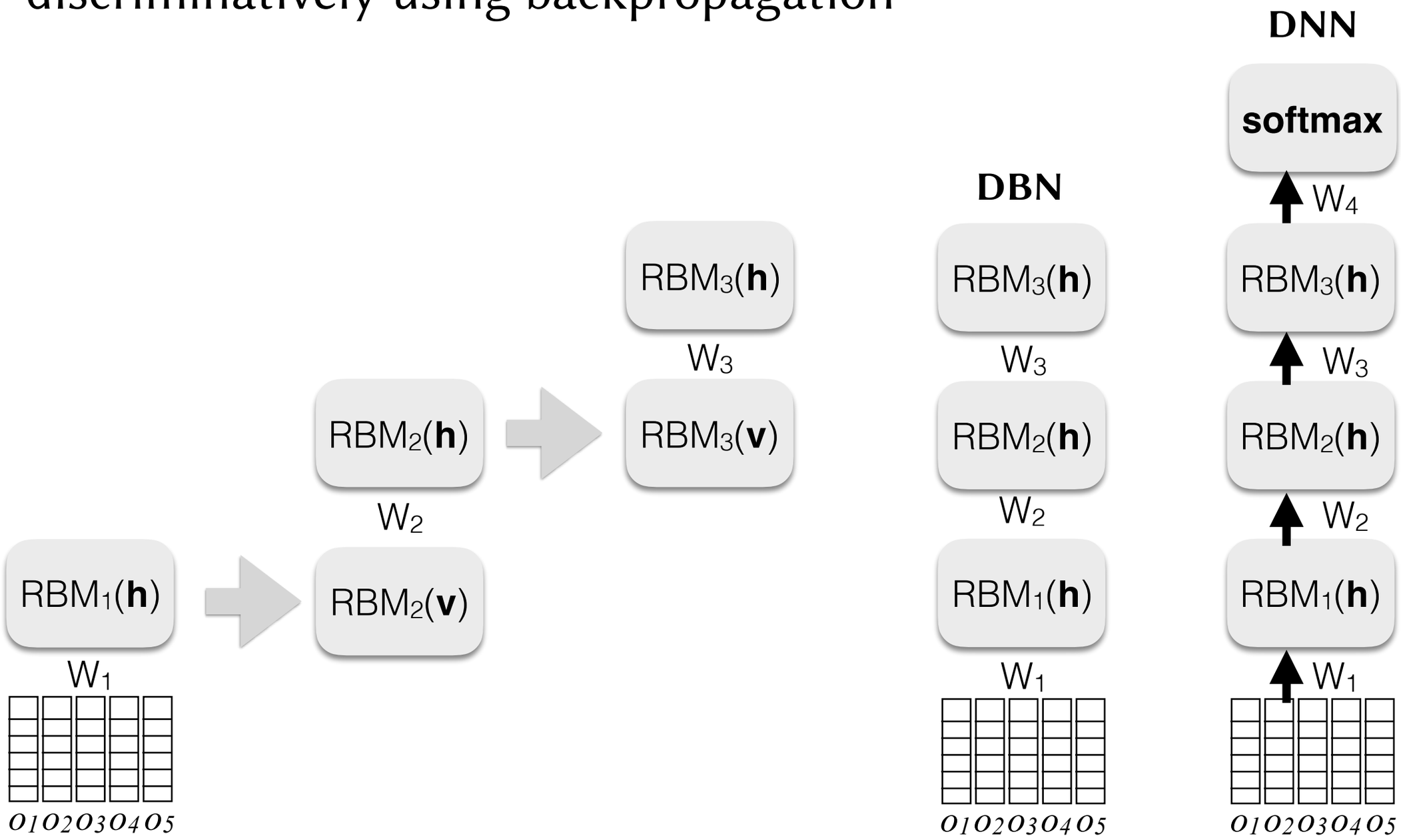
- RBMs can be stacked to make a “deep belief network”:
  1) Inferred hidden states can be used as data to train a second RBM 2) repeat this step

---

[H02] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” Neural Comput., 14, 1771–1800, ’02.

Discriminative fine-tuning

- After learning a DBN by layerwise training of the RBMs, resulting weights can be used as initialisation for a deep feedforward NN
- Introduce a final softmax layer and train the whole DNN discriminatively using backpropagation



Pretraining

- Pretraining is fast as it is done layer-by-layer with contrastive divergence

- Other pretraining techniques include stacked autoencoders, greedy discriminative pretraining. (Details not discussed in this class.)

- Turns out pretraining is not a crucial step for large speech corpora

Summary of DNN-HMM acoustic models

Comparison against HMM-GMM on different tasks

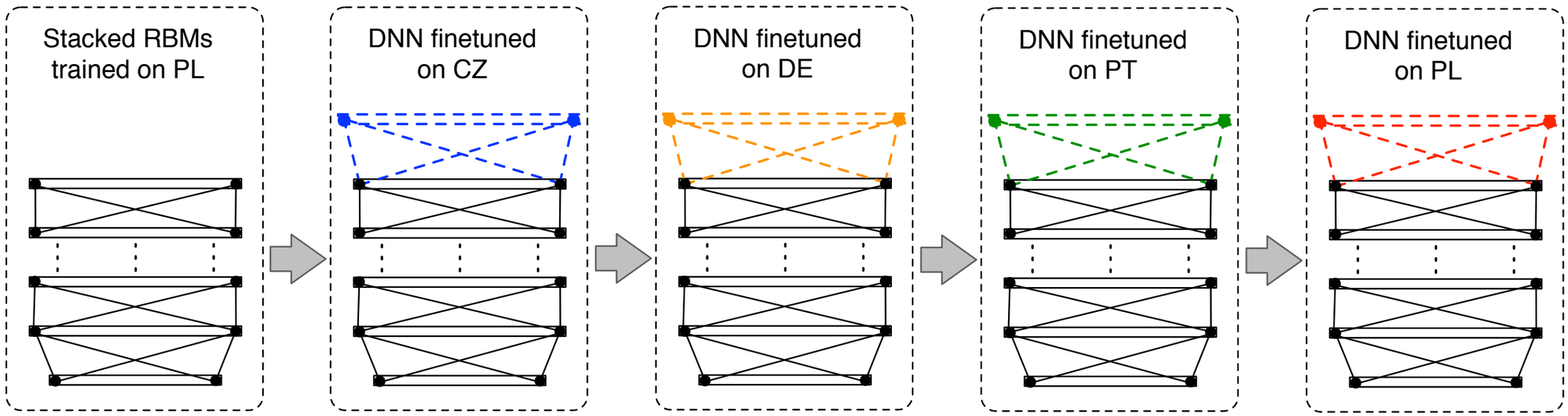
[TABLE 3] A COMPARISON OF THE PERCENTAGE WERs USING DNN-HMMs AND GMM-HMMs ON FIVE DIFFERENT LARGE VOCABULARY TASKS.

TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	

Hybrid DNN-HMM systems consistently outperform GMM-HMM systems (sometimes even when the latter is trained with lots more data)

Table copied from G. Hinton, et al., “Deep Neural Networks for Acoustic Modeling in Speech Recognition”, IEEE Signal Processing Magazine, 2012.

Multilingual Training (Hybrid DNN/HMM System)

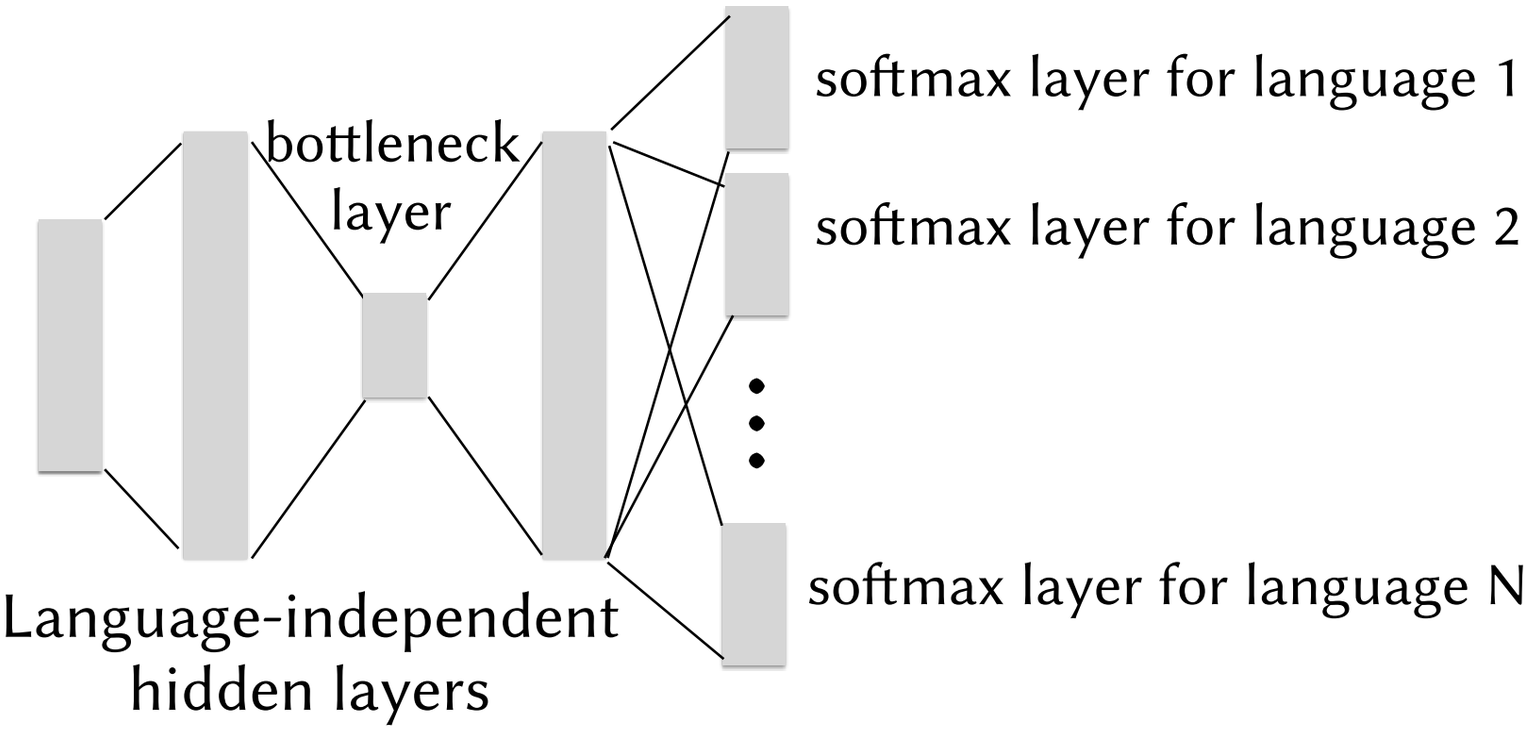


Languages	Dev	Eval
RU	27.5	24.3
CZ →RU	27.5	24.6
CZ →DE →FR →SP →RU	26.6	23.8
CZ →DE →FR →SP →PT →RU	26.3	23.6

Monolingual and multilingual DNN results on Russian

Image/Table from Ghoshal et al., “Multilingual training of deep neural networks”, ICASSP, 2013.

Multilingual Training (Tandem System)



Language	Czech	English	German	Portugese	Spanish	Russian	Turkish	Vietnamese
HMM	22.6	16.8	26.6	27.0	23.0	33.5	32.0	27.3
mono-BN	19.7	15.9	25.5	27.2	23.2	32.5	30.4	23.4
1-Softmax	19.4	15.5	24.8	25.6	23.2	32.5	30.3	25.9
8-Softmax	**19.3**	**14.7**	**24.0**	**25.2**	**22.6**	**31.5**	**29.4**	**24.3**

Monolingual/multilingual BN feature-based results

Vesely et al., “The language-independent bottleneck features”, SLT, 2012.