Supervised Sequence Labelling with Recurrent Neural Networks

Chapter 4 to Chapter 7

Alex Graves, Ph.D Thesis (2008)

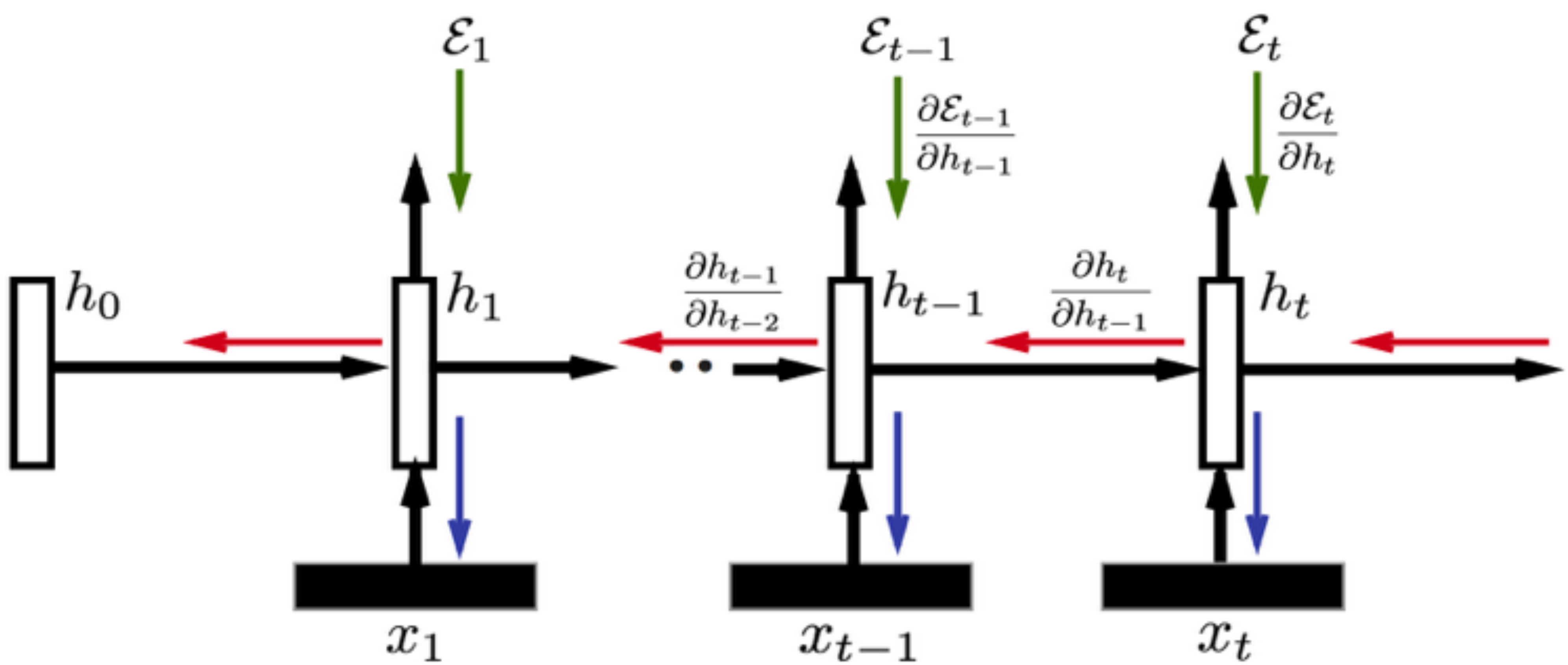
Kazuya Kawakami

Recurrent Neural Networks

- Benefit of Recurrent Networks
  - Learning representations of contextual information of sequence.

- Limitations of RNNs
  - The range of accessible context is limited. (**Vanishing** / **Exploding** Gradient)

$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial \theta} \right) \qquad \frac{\partial h_t}{\partial h_k} = \prod_{k < i \le t} \frac{\partial h_i}{\partial h_{i-1}}$$



Previous work addressing Vanishing Gradient Problem

Old works listed in the paper

- Simulated Annealing and Discrete Error Propagation (Bengio et al 1994)
- Explicit time delay modeling (Lang et al, 1990; Lin et al, 1996; Plate, 1993)
- Time constants (Mozer, 1992)
- Hierarchical sequence compression (Schmidhuber, 1992)

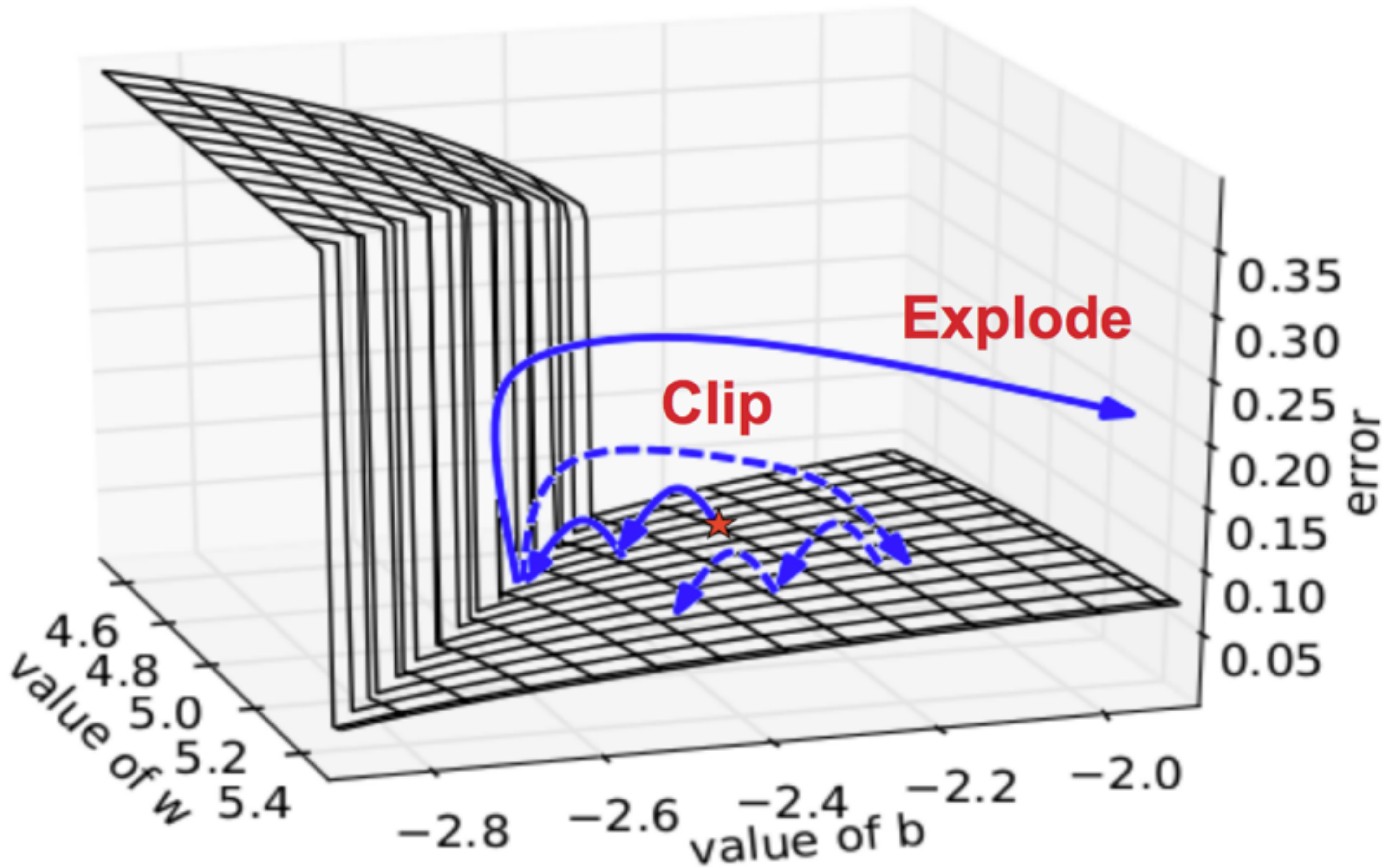
Recent work

- L1 / L2 regularization on the recurrent weights
- Hessian-Free Optimization (Sutskever et al, 2011)
- Gradient Clip (Pascanu et al, 2013)

Technique explained this paper

- **Long Short Term Memory** (Hochreiter and Schmidhuebr, 1997)

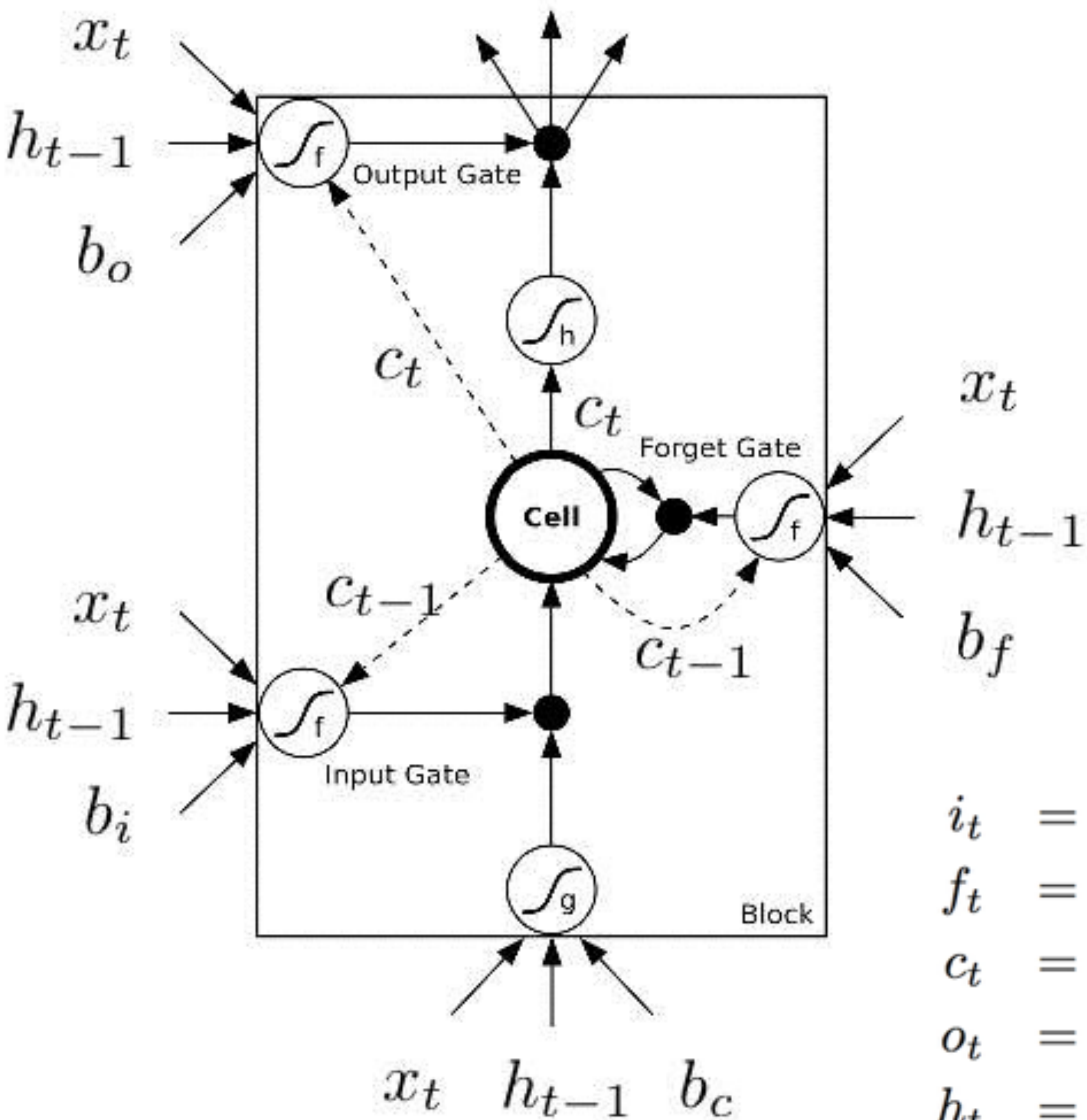
Gradient Clipping



[Pascanu et al. 2013]

Long Short Term Memory - Network Architecture

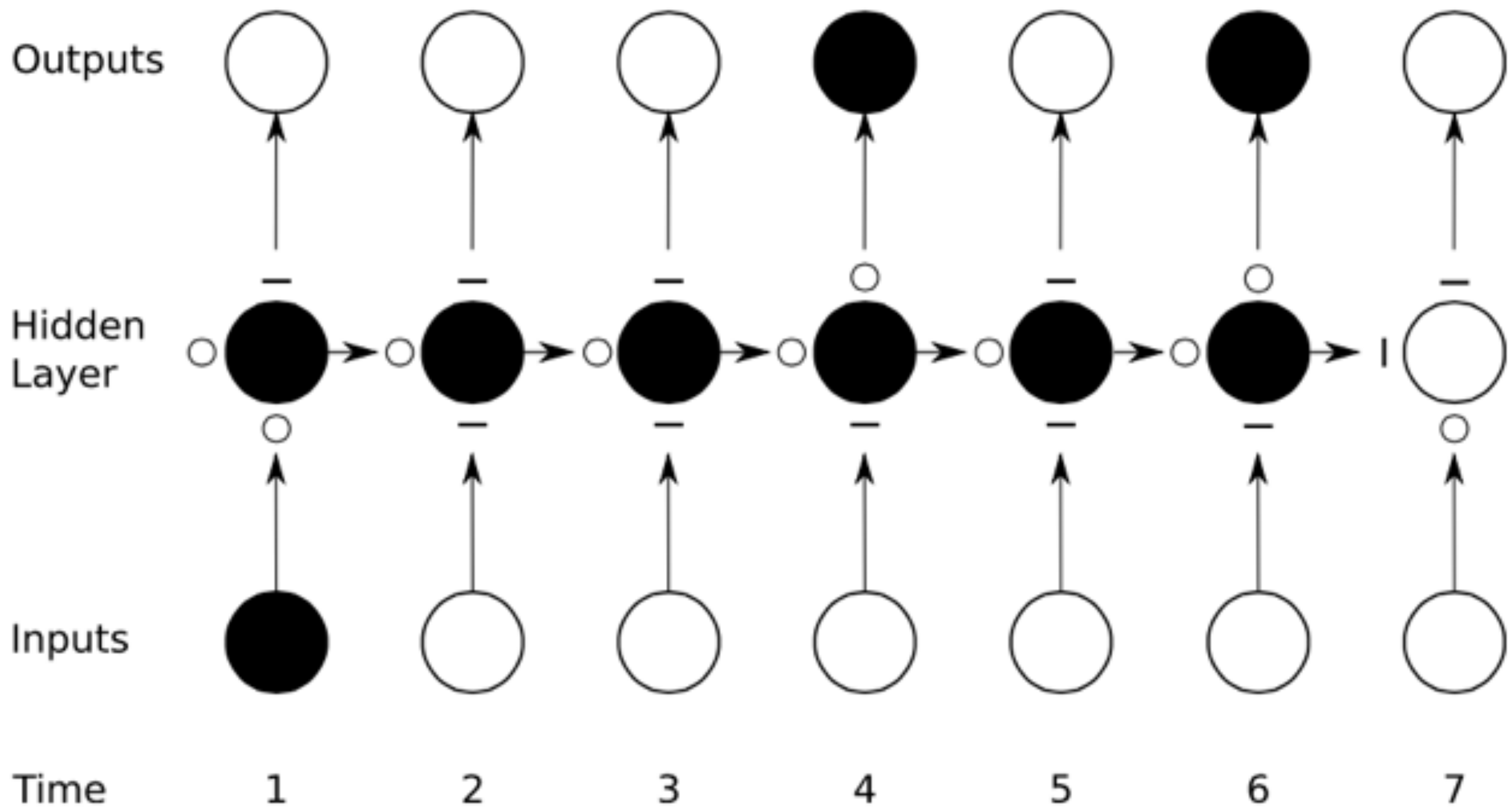
- LSTM Computation per cell
  - Training with BPTT (or Real Time Recurrent Learning: RTRL)



$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}
$$

Preservation of Gradient with LSTM

- If the input gate = 0 and forget gate = 1, gradient pass through the cell



$$c_t = f_t c_{t-1} + i_t tanh(W_{xc} x_t + W_{hc} h_{t-1} + b_c)$$

Architectural Variants

- Original LSTM model only have inputs and output gates.
- Forget gates 'reset' themselves to forget previous information.

$$c_t = \underbrace{0}_{\text{reset}} c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

- Peephole weights provide precise timing.
  Caution:: Output gate take c_t

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + \underline{W_{ci}}c_{t-1} + b_i)$$

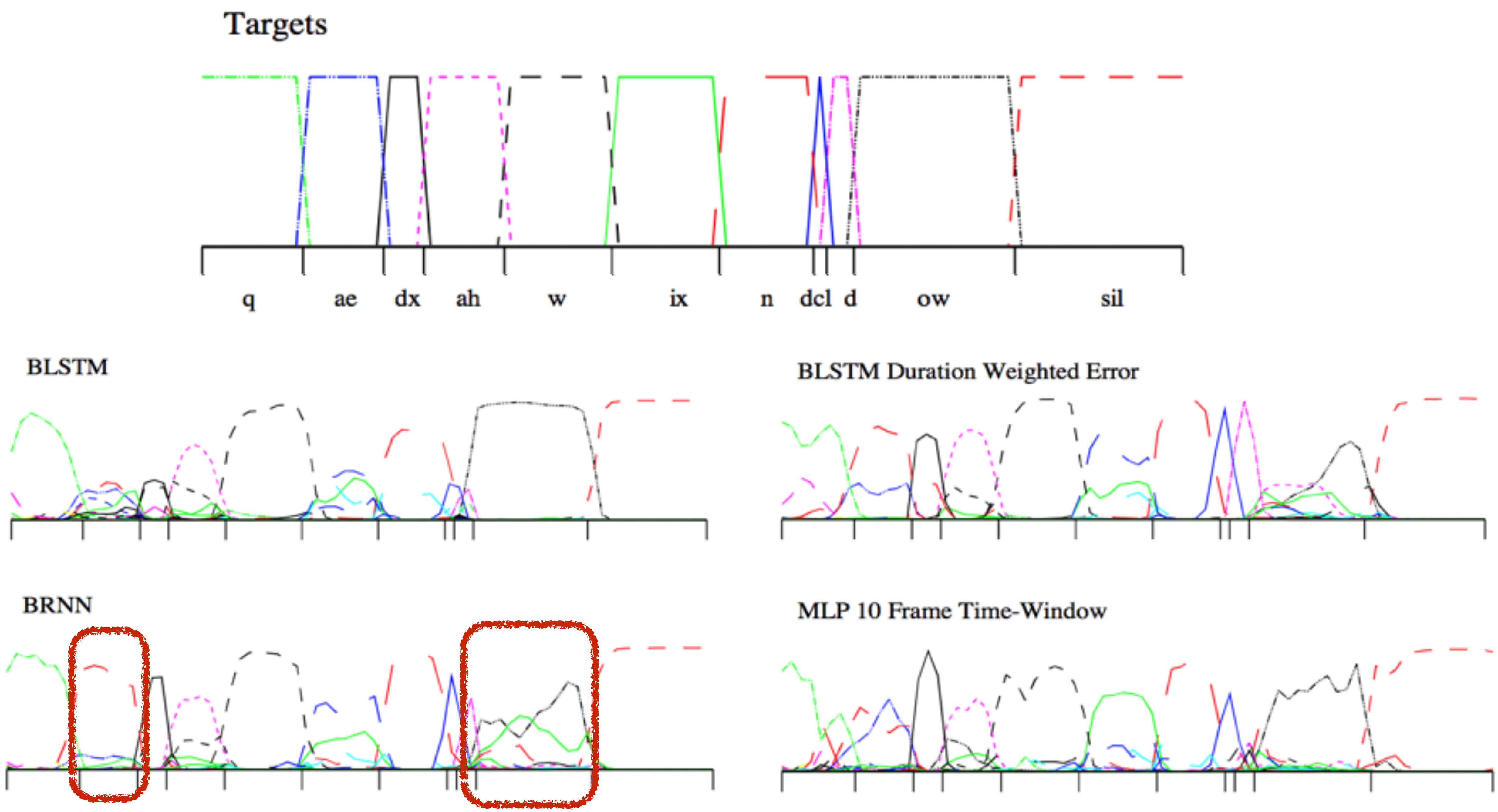
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + \underline{W_{cf}}c_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + \underbrace{W_{co}}_{\text{Peephole weights}}c_t + b_o)$$

- Bi-directional LSTM

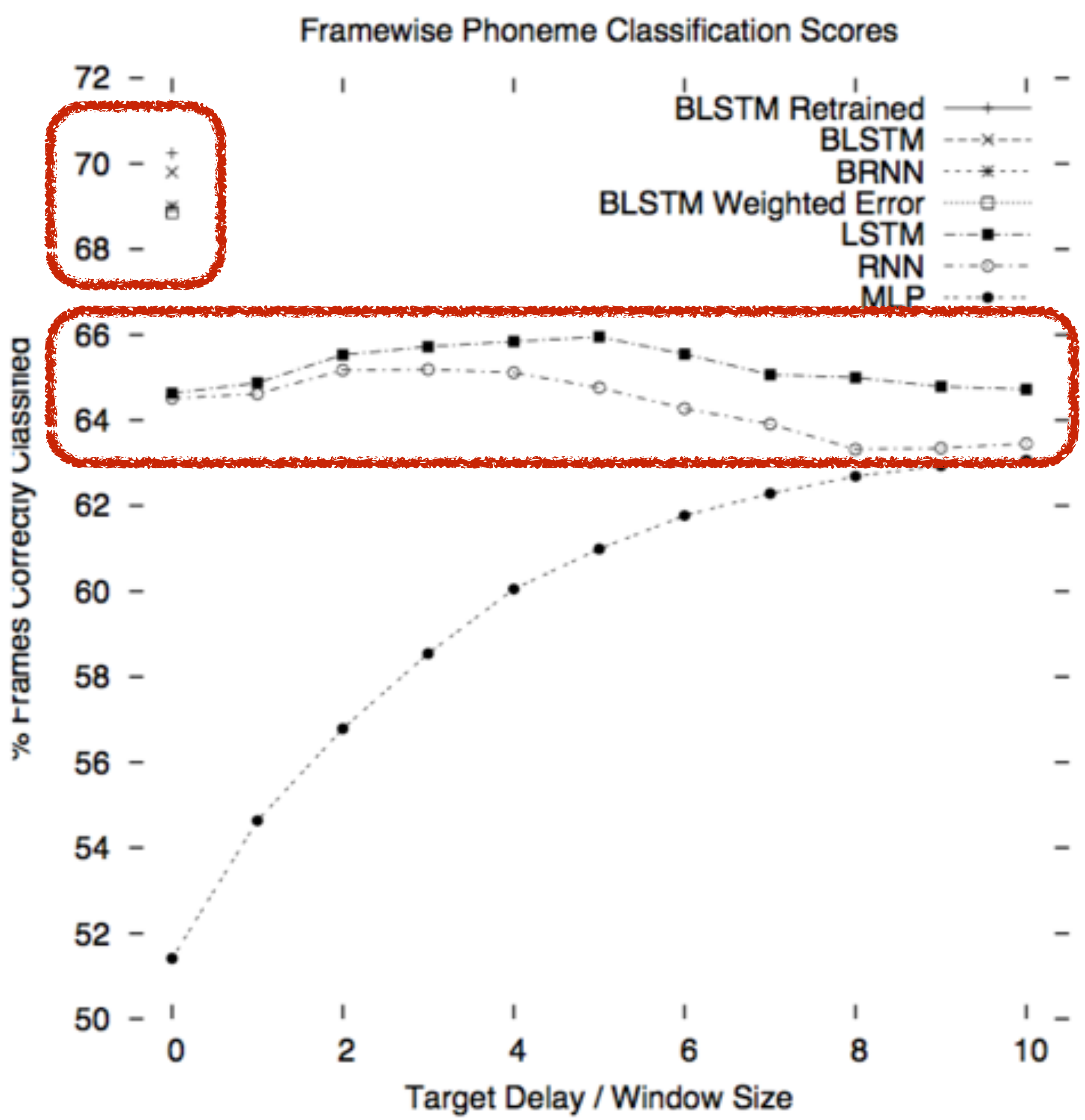
Comparison of Network Architecture

- TIMIT Dataset
  - All networks have approximately the same number of weights for fair comparison



Comparison of Network Architecture

- BLSTM achieve best result without Target Delay
  - Single way RNN / LSTM do not have full contextual information



Comparison of Network Architecture

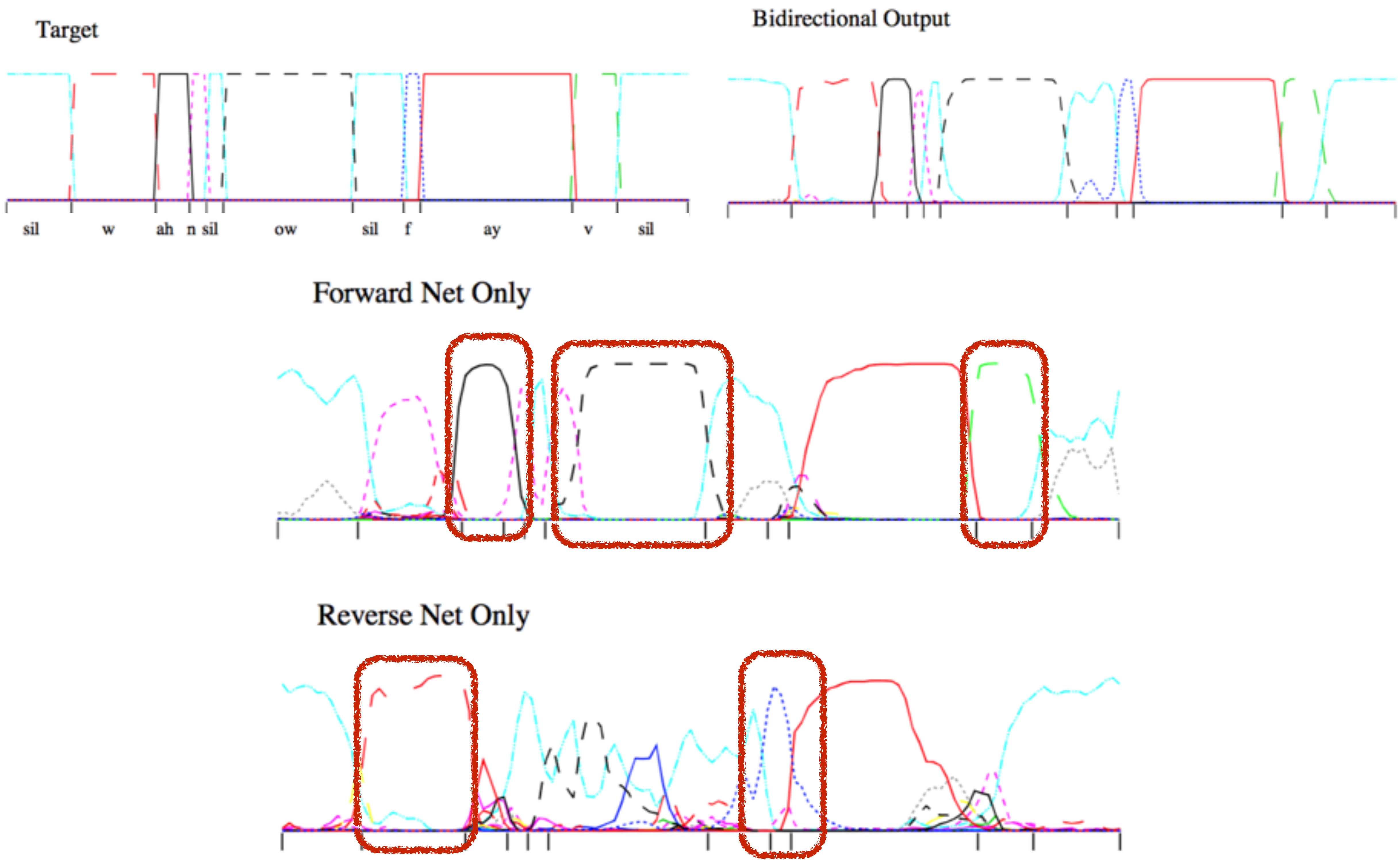
- BLSTM achieve best performance with no-time delay and fewer epochs

Table 5.1: **Framewise phoneme classification results on TIMIT.** The error measure is the frame error rate (percentage of misclassified frames). BLSTM results are means over seven runs $\pm$ standard error.

Network	Train Error (%)	Test Error (%)	Epochs
MLP (no window)	46.4	48.6	835
MLP (10 frame window)	32.4	36.9	990
RNN (delay 0)	30.1	35.5	120
LSTM (delay 0)	29.1	35.4	15
LSTM (backwards, delay 0)	29.9	35.3	15
RNN (delay 3)	29.0	34.8	140
LSTM (delay 5)	22.4	34.0	35
BLSTM (Weighted Error)	24.3	31.1	15
BRNN	24.0	31.0	170
BLSTM	22.6$\pm$0.2	30.2$\pm$0.1	20.1$\pm$0.5
BLSTM (retrained)	21.4	29.8	17

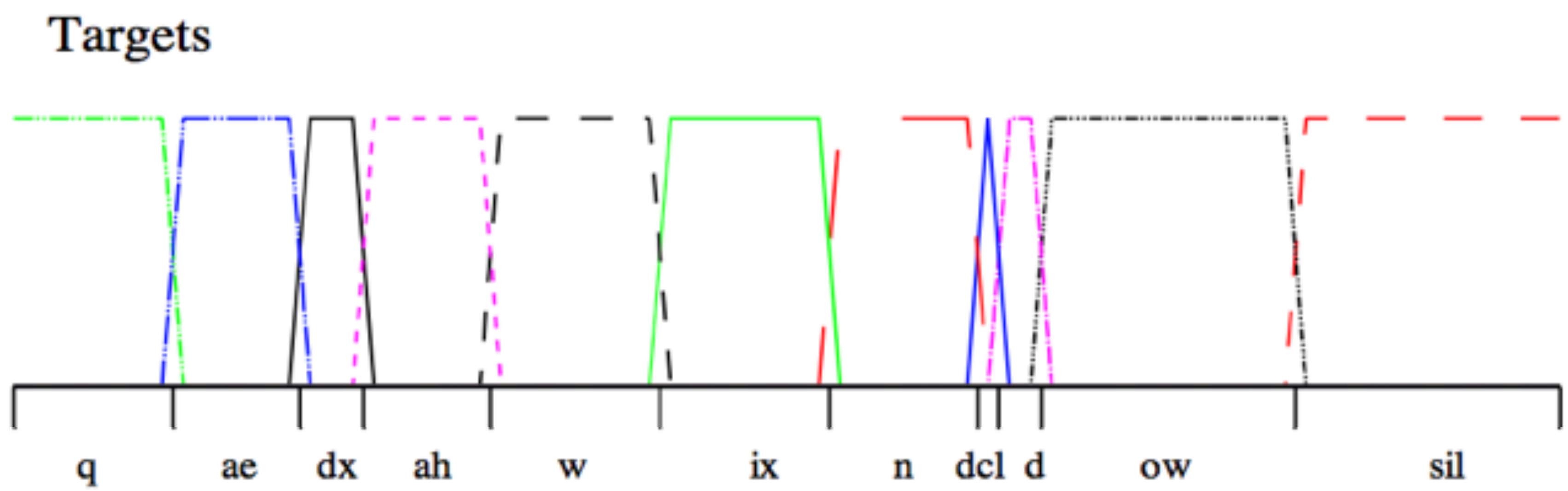
Comparison of Network Architecture - LSTM v.s. BLSTM

- Forward and Backward have different properties.



HMM Hybrids

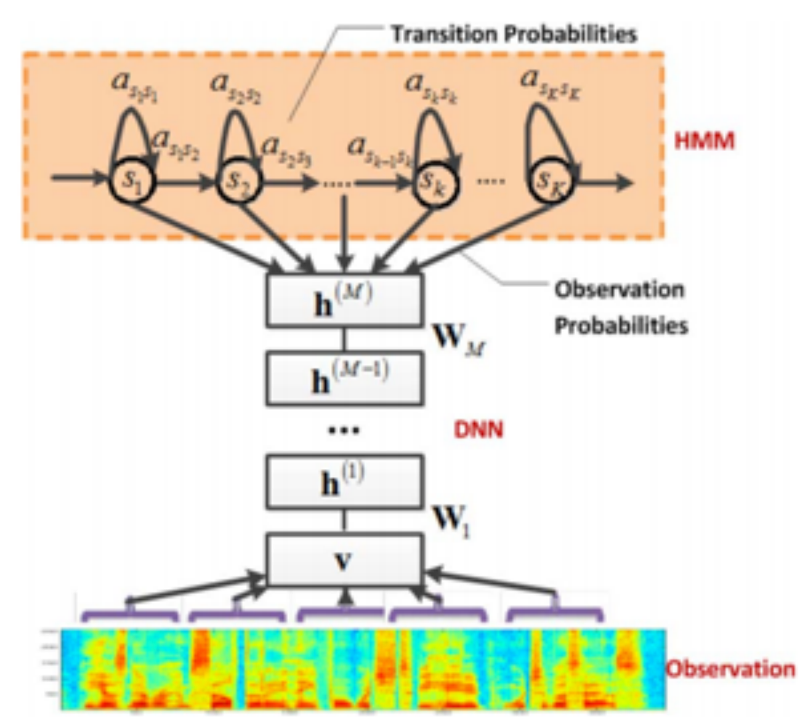
- Problem of RNN
  - RNN map N-length sequence to N-length output => Redundant



Labels we want

Output of RNN q q q q ae ae dx ah ah w w w w .....

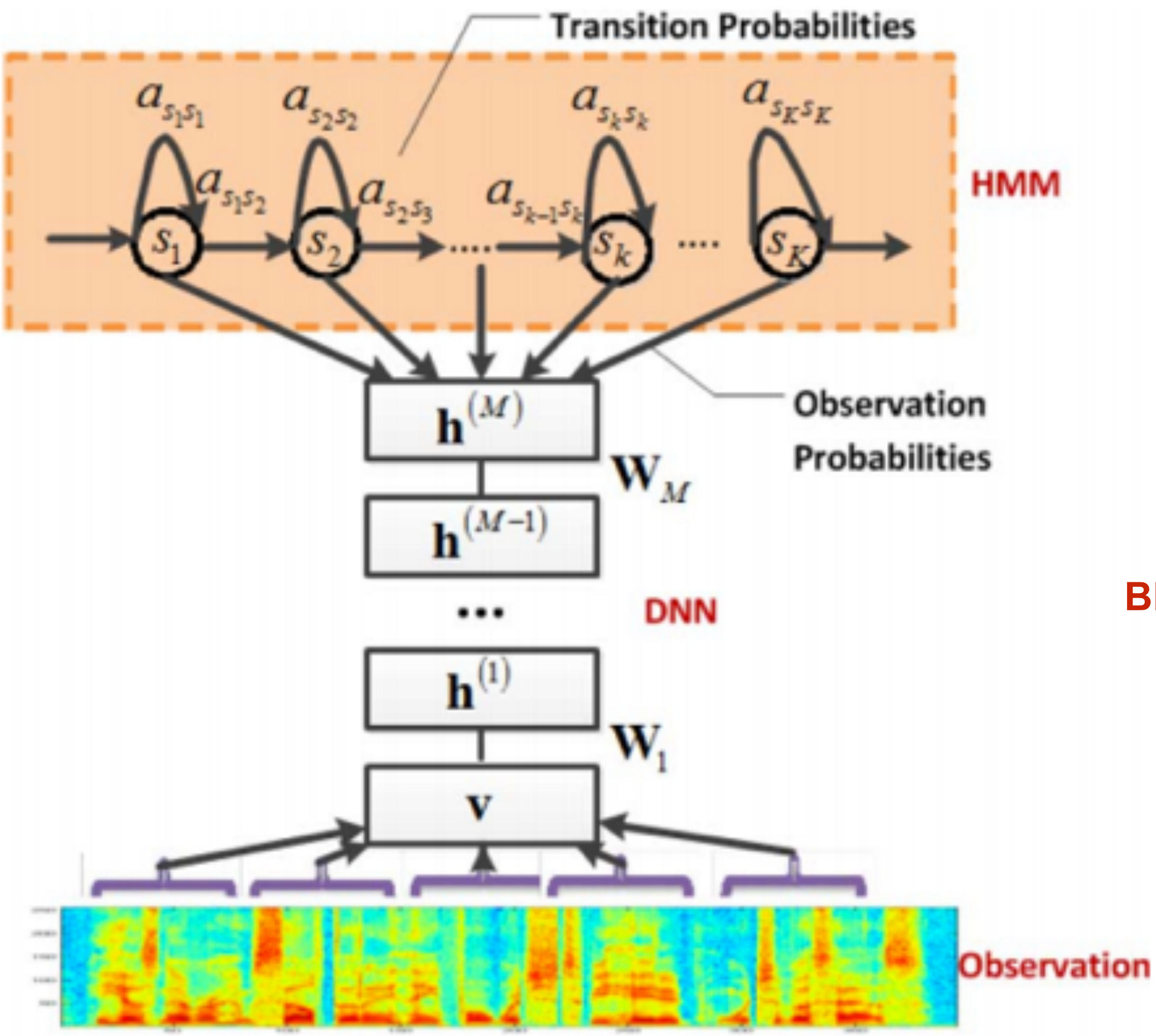
- GMM-HMM => DNN-HMM
  - Traditionally we use hand crafted feature + GMM
  - Instead of using handcrafted feature, use DNN
    - More contextual information can be included.
  - Training is done by iterative fashion
    - Train HMM and Re-train DNN



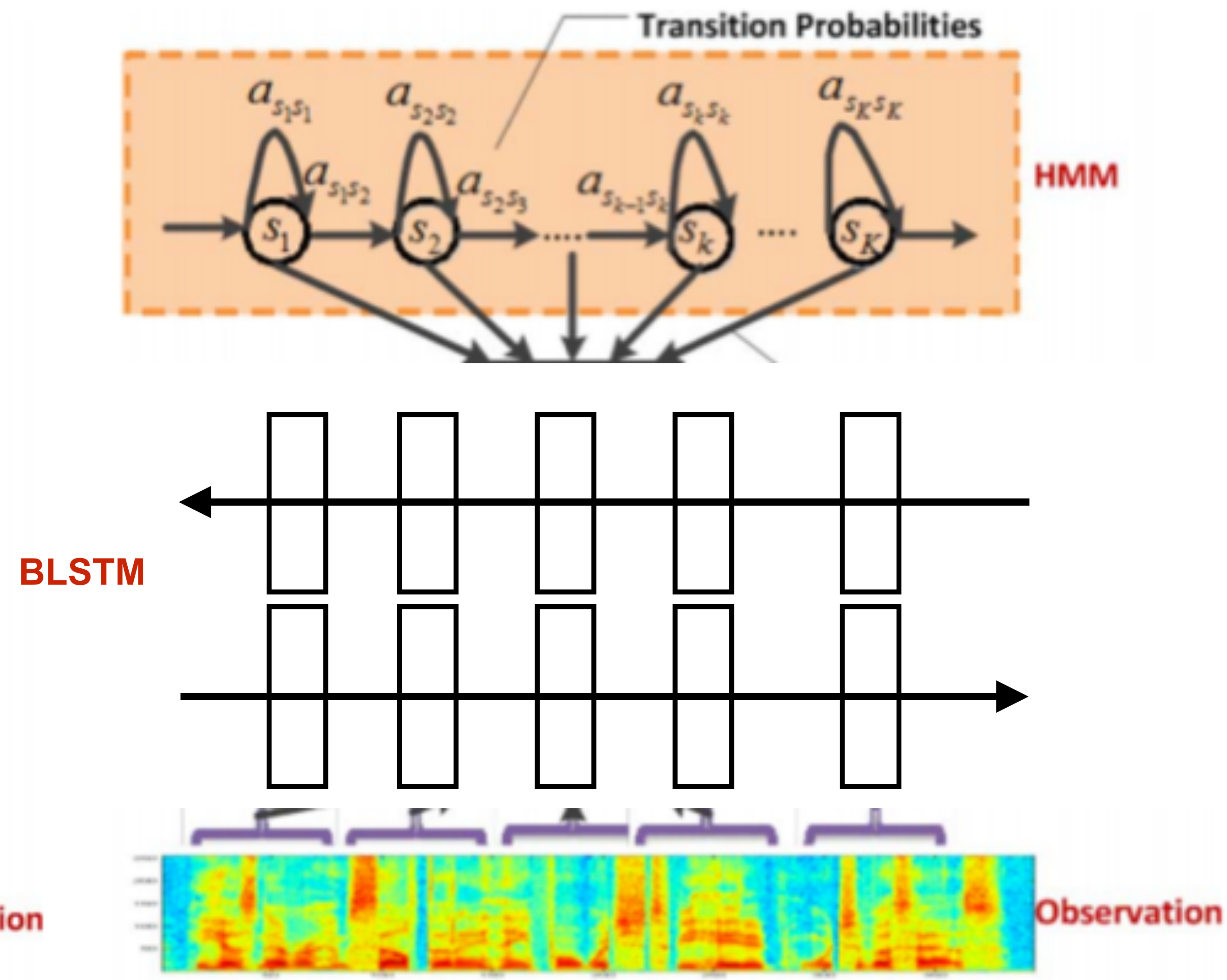
GE Dhal et al. Context-Dependent Pre-Trained Deep NeuralNetworks for Large-Vocabulary Speech Recognition

HMM Hybrids + BLSTM

- Replace DNN part with BRNN



Context Dependent HMM



HMM-BLSTM

HMM Hybrids

- HMM-BLSTM hybrids outperformed both context-dependent/ -independent HMMs.
  - The number of parameters is much smaller than context dependent HMM

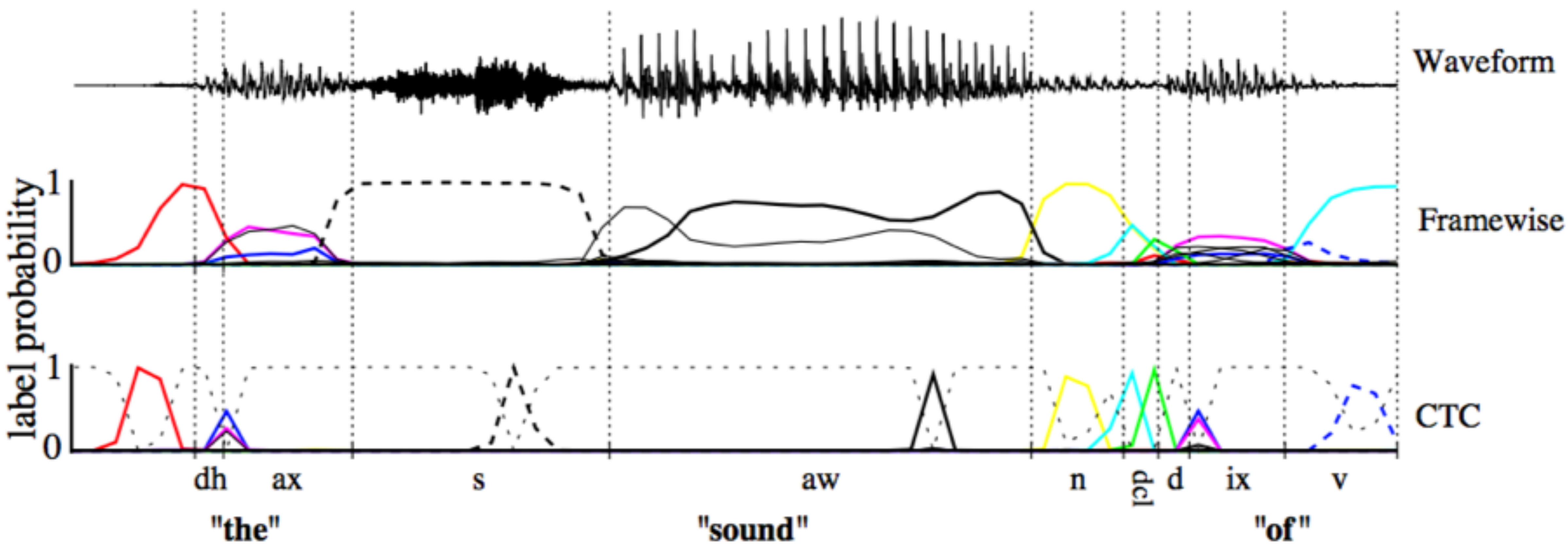
Table 6.1: **Phoneme recognition results on TIMIT.** The error measure is the phoneme error rate. Hybrid results are means over 5 runs, ± standard error. All differences are significant ($p < 0.01$).

System	Parameters	Error (%)
Context-independent HMM	80 K	38.85
Context-dependent HMM	>600 K	35.21
HMM-LSTM	100 K	39.6 ± 0.08
HMM-BLSTM	100 K	33.84 ± 0.06
HMM-BLSTM (weighted error)	100 K	31.57 ± 0.06

Connectionist Temporal Classification (CTC)

- Special Output Layer for Sequence Labeling Task
  - where alignments with input and labels are unknown.
  - replace HMM with CTC Layer

- Benefit of CTC
  - No need for have pre-segmented training data
  - No need for external post-processing to extract the label sequence



Formulation of CTC

- Suppose our labels are alphabet A
- CTC consists of softmax output layer with one more than $|A|$.
- The extra output correspond to ‘ blank ’ label. $A' = A \cup \{blank\}$
- If we have T length sequence x over A'
- The conditional probability over each sequence $\pi \in A'^T$

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y^t_{\pi_t}$$

y^t_k : activation of network output k at time t

- There are many mapping from redundant sequence to label sequence $\mathbf{l} \in A^{\leq T}$
  - ex. a - b - - b = a b - - - b => abb
  - The mapping function from redundant sqeuence to label sequence is noted as

$$\mathcal{F} : A'^T \mapsto A^{\leq T}$$

- The probability of label sequence is calculated by summing over all possible path

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{F}^{-1}(\mathbf{l})} p(\pi|\mathbf{x})$$

Learning with CTC

- Computation of $p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{F}^{-1}(\mathbf{l})} p(\pi|\mathbf{x})$ grows exponentially with length of input

  - Use Forward Backward algorithm as we do in HMM

- Suppose we have T length input U length label l .
  - Suppose all the paths at time t pass through u , $V(t,u)$
  - We can define sum of all probabilities at (t,u) as

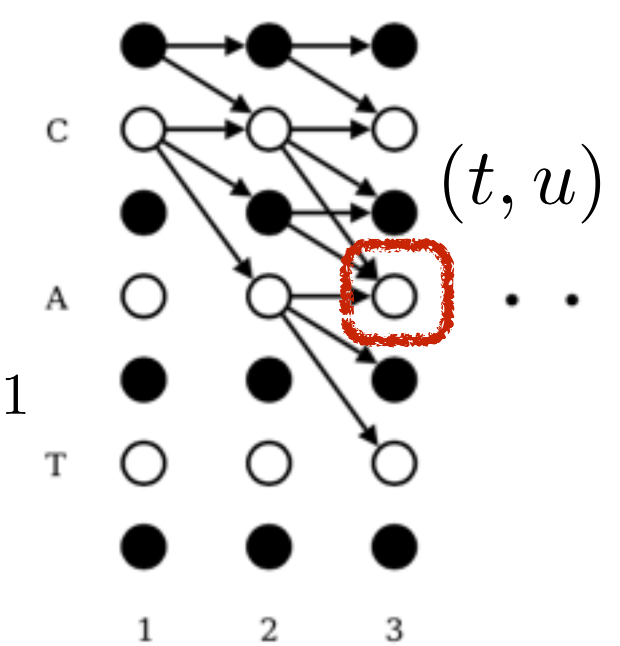
$$\alpha(t,u) = \sum_{\pi \in V(t,u)} \prod_{i=1}^{t} y^{i}_{\pi_i}$$

$$\alpha(1,1) = y^1_b$$
$$\alpha(1,2) = y^1_{l_1}$$
$$\alpha(1,u) = 0, \ \forall u > 2$$

  - The paths after (t,u) denoted as $W(t,u)$
  - We can also define sum of all probability after (t,u)

$$\beta(t,u) = \sum_{\pi \in W(t,u)} \prod_{i=1}^{T-t} y^{t+i}_{\pi_i}$$

Length
$U' = 2U + 1$



CTC forward backward algorithm

Learning with CTC

- Recursive operation over graph

$$\alpha(t,u) = y^t_{l'_u} \sum_{i=f(u)}^{u} \alpha(t-1,i)$$

- where

$$f(u) = \begin{cases} u-1 & \text{if } l'_u = \text{blank or } l'_{u-2} = l'_u \\ u-2 & \text{otherwise} \end{cases}$$

- $\beta(t,u)$ is computed as $\alpha(t,u)$, but in backward.
- In practice, we will use log-scale probabilities

Loss Function, Prediction

- Negative log probability of correctly labelling over all training examples

$$\mathcal{L}(S) = -\ln \prod_{(\mathbf{x},\mathbf{z})\in S} p(\mathbf{z}|\mathbf{x}) = -\sum_{(\mathbf{x},\mathbf{z})\in S} \ln p(\mathbf{z}|\mathbf{x})$$

- As we defined forward and backward probability at (t, u) ,
  - Probabilities over all possible labels is

$$p(\mathbf{z}|\mathbf{x}) = \sum_{u=1}^{|\mathbf{z}'|} \alpha(t,u)\beta(t,u)$$

  - Thus we get negative log likelihood over sequence

$$\mathcal{L}(\mathbf{x},\mathbf{z}) = -\ln \sum_{u=1}^{|\mathbf{z}'|} \alpha(t,u)\beta(t,u)$$

- Learning is done by BPTT, and for prediction we solve the following decoding
  - Best path search, Prefix Search Decoding, Constrained Decoding

$$\hat{l} = \underset{l}{\operatorname{argmax}}\, p(l, X)$$

Experiments - Phoneme Recognition

- TIMIT phoneme recognition problem, BLSTM-CTC get close to State-of-the-art.

Table 7.3: **Phoneme recognition results on TIMIT with 39 phonemes.** The error measure is the phoneme error rate. Results for BLSTM-CTC are averages $\pm$ standard error over 10 runs. The average number of training epochs was 112.5 ± 6.4

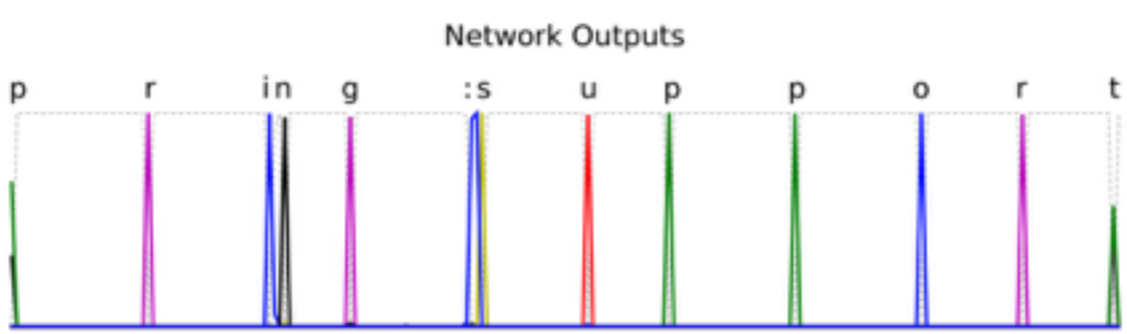
System	Error (%)
Conditional Random Fields (Morris and Lussier, 2006)	34.8
Large Margin HMM (Sha and Saul, 2006)	28.7
Baseline HMM (Yu et al., 2006)	28.6
Triphone Continuous Density HMM (Lamel and Gauvain, 1993)	27.1
Augmented Conditional Random Fields (Hifny and Renals, 2009)	26.7
RNN-HMM Hybrid (Robinson, 1994)	26.1
Bayesian Triphone HMM (Ming and Smith, 1998)	25.6
Near-miss Probabilistic Segmentation (Chang, 1998)	25.5
BLSTM-CTC (best path decoding)	25.2 ± 0.2
Monophone HTMs (Yu et al., 2006)	24.8
BLSTM-CTC (prefix search decoding)	24.6 ± 0.2
Heterogeneous Classifiers (Glass, 2003)	24.4
Discriminative BMMI Triphone HMMs (Sainath et al., 2009)	22.7
Monophone Deep Belief Network (Mohamed et al., 2011)	20.7

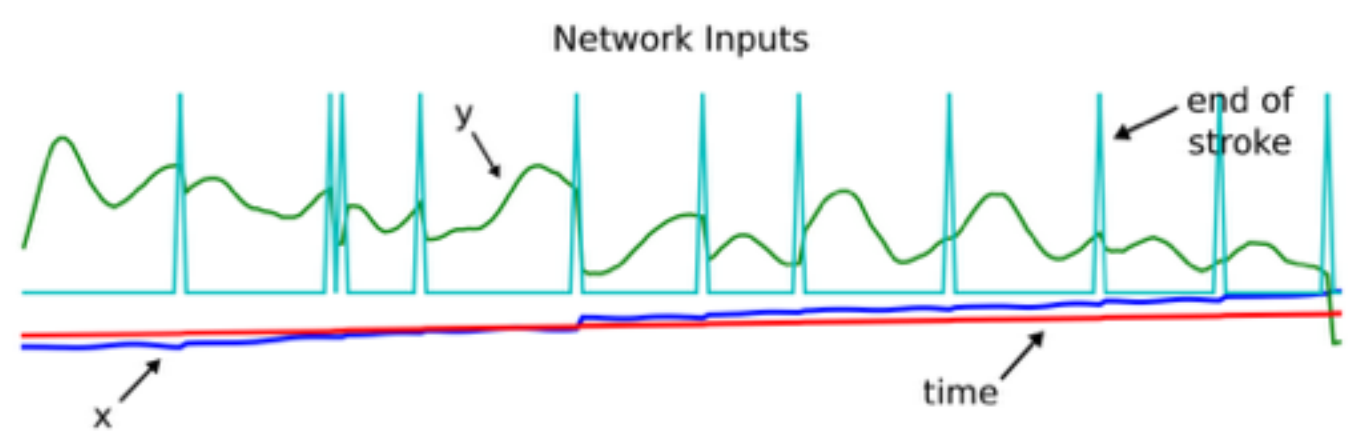
Experiment - Online Hand Writing Recognition

- BLSTM-CTC result significantly improve performance over HMM.
  - Bidirectional RNN did not converge on this task

Table 7.6: **Word recognition on IAM-OnDB.** The error measure is the word error rate. LM = language model. BLSTM-CTC results are a mean over 4 runs, ± standard error. All differences were significant ($p < 0.01$).

System	Input	LM	Error (%)
HMM	Preprocessed	✓	35.5
BLSTM-CTC	Raw	✗	30.1 ± 0.5
BLSTM-CTC	Preprocessed	✗	26.0 ± 0.3
BLSTM-CTC	Raw	✓	22.8 ± 0.2
BLSTM-CTC	Preprocessed	✓	20.4 ± 0.3





Discussion - HMM v.s. CTC

- HMM is **generative** model, LSTM-CTC is **discriminative** model
  - Discriminative Model achieve better performance on Labeling task.

- GMM-HMM with diagonal covariance matrix assume features are not correlated.
  - RNN, **LSTM do not assume features came from particular distribution**.
  - RNN, LSTM can model non-linear relationship among features.

- States of HMM are discrete and single valued
  - RNN, **LSTM activations are continuous** and multivariate.
  - HMM with N states can model only O(logN) bit information
  - For RNN, the amount of information grows linearly with the number of hidden units

- RNN-CTC can **deal with continuous input** without segments

- HMM define probabilities of each observation to depend only on current state
  - HMM cannot model data where each observation depend on observation around.
  - Modeling longer range of context is difficult with HMM.
  - HMM can model rich context with more parameters, but get unstable.

Summary

- RNN v.s. LSTM
  - RNN suffer Vanishing / Exploding gradient
  - RNN sometimes don’t even converge where LSTM converge.

- Advantage of LSTM
  - LSTM let use capture longer context with input/forget gates by passing gradients

- Forward / Backward LSTM => BLSTM
  - Phoneme recognition task, forward and backward LSTM capture data differently
  - Combining both (BLSTM) yields stable, and better performance

- HMM can be used for problems where we do not know input-output alignment
  - Context-dependent Multilayer Perceptron
  - DNN-HMM
  - BRNN-HMM, BLSTM-HMM

- CTC output layer replace HMM
  - CTC define forward-backward probability over possible sequence
  - Learning can be done with BPTT, and Prediction can be done with Decoding.