Machine Learning

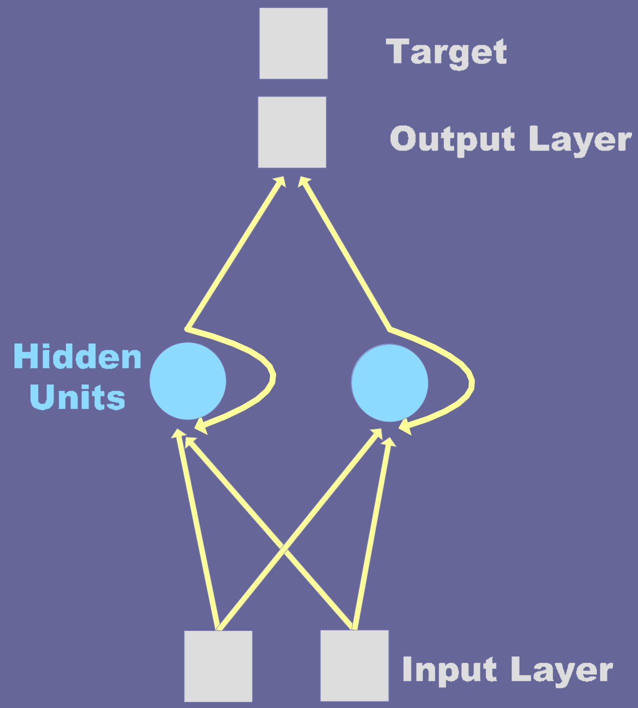
Lecture 17

Long-Short Term Memory (LSTM)

Overview of LSTM

- Replace sigmoidal unit with linear unit
- Error flows through linear unit does not decay
- But error flow would very often diverge
- Surround linear unit with multiplicative gates to allow greater control over flow of information.
- Using gates, possible to have stable constant error flow

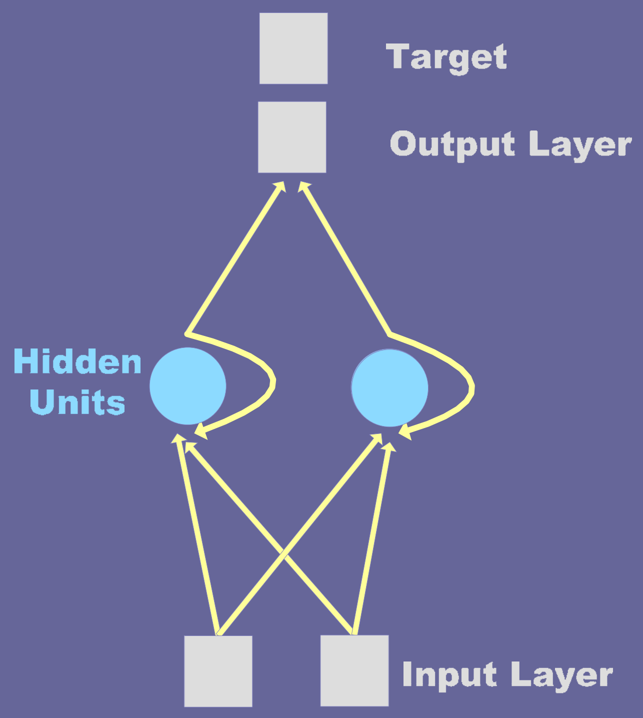
Start with standard recurrent neural network



Buiding LSTM... other units in network blah b

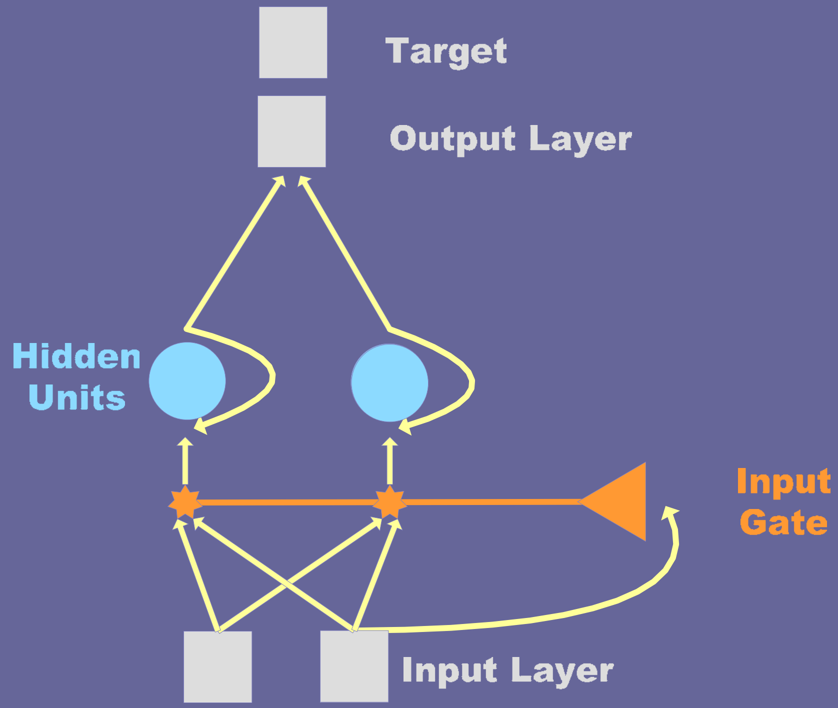
Start with standard recurrent neural network

An Input Gate lets LSTM selectively process incoming information

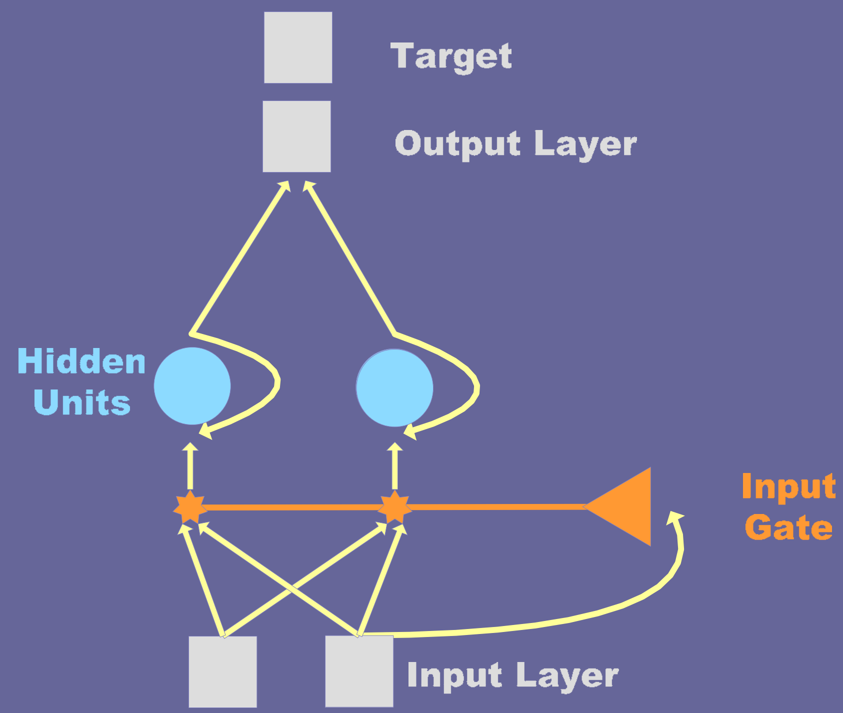


Start with standard recurrent neural network

An Input Gate lets LSTM selectively process incoming information



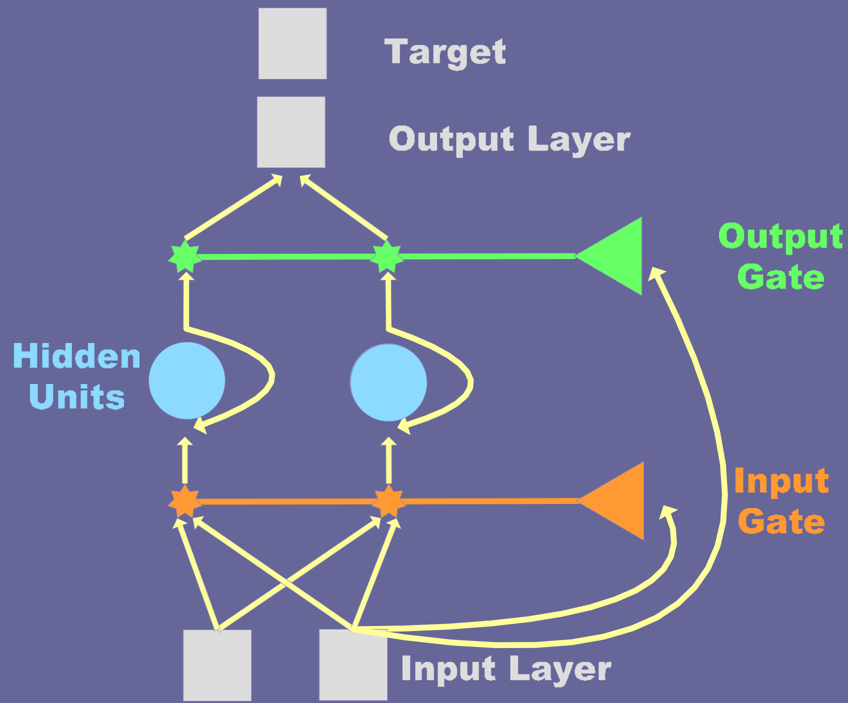
An Output Gate allows units to selectively take themselves offline



An Input Gate lets LSTM selectively process incoming information

An Output Gate allows units to selectively take themselves offline

Forget Gate enables a block to empty it's own memory contents

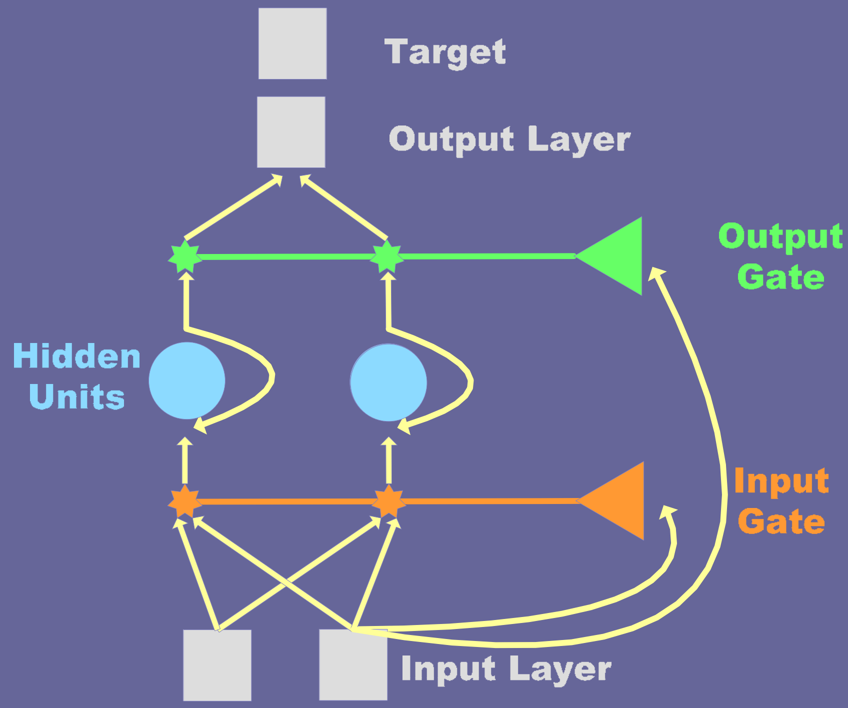


An Input Gate lets LSTM selectively process incoming information

An Output Gate allows units to selectively take themselves offline

A Forget Gate enables units to empty their own memory contents

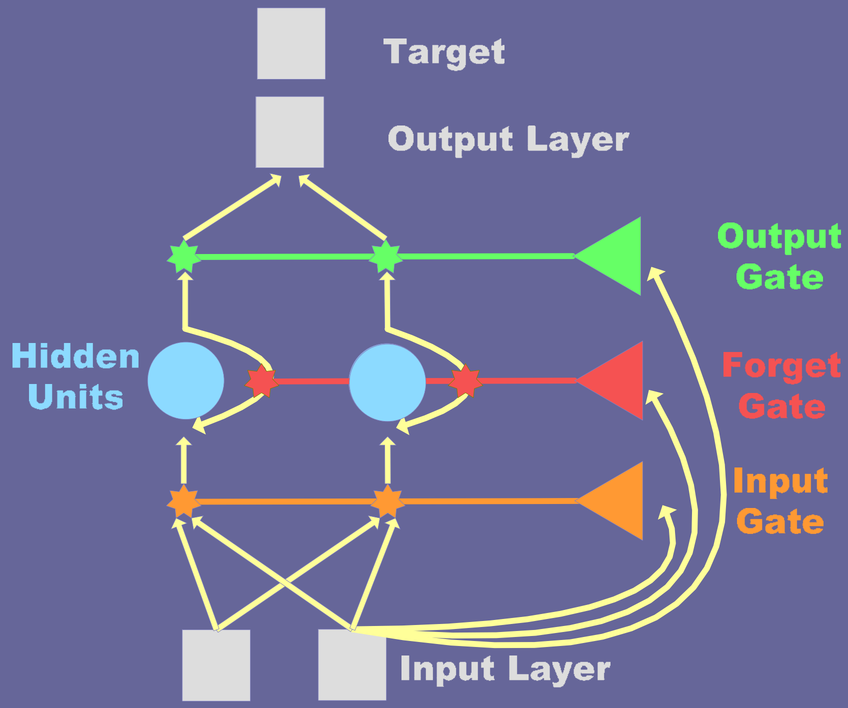
An Input Gate lets LSTM selectively process incoming information



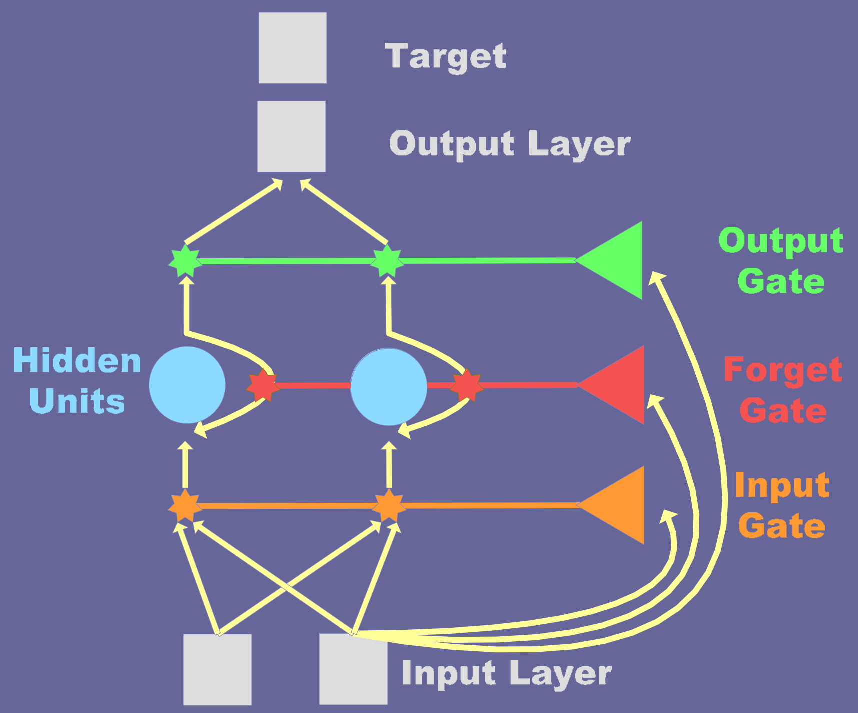
An Output Gate allows
units to selectively
take themselves offline

A Forget Gate enables
units to empty their
own memory contents

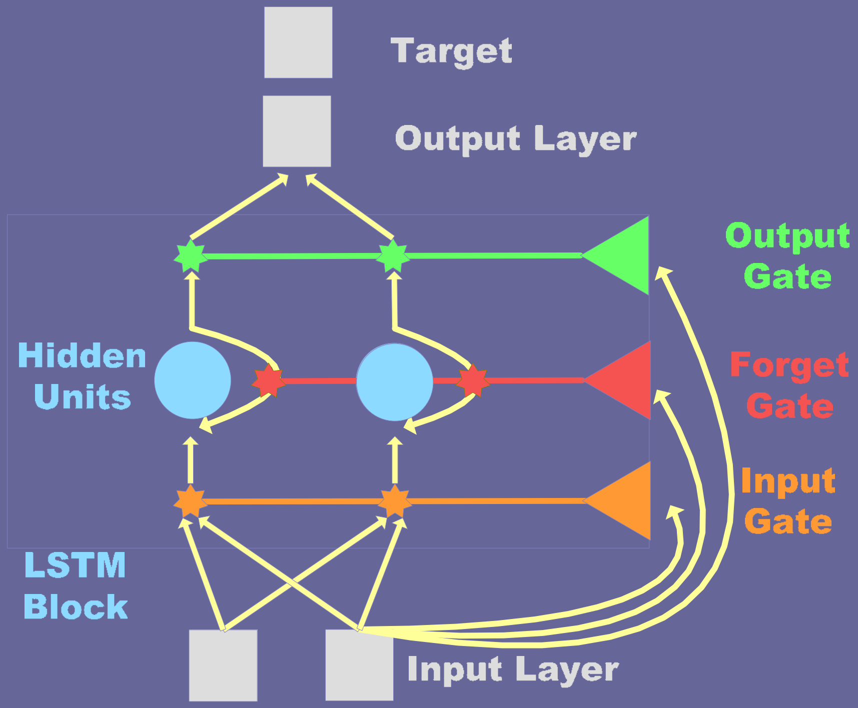
An Input Gate lets LSTM
selectively process
incoming information



The resulting structure is called an LSTM Block

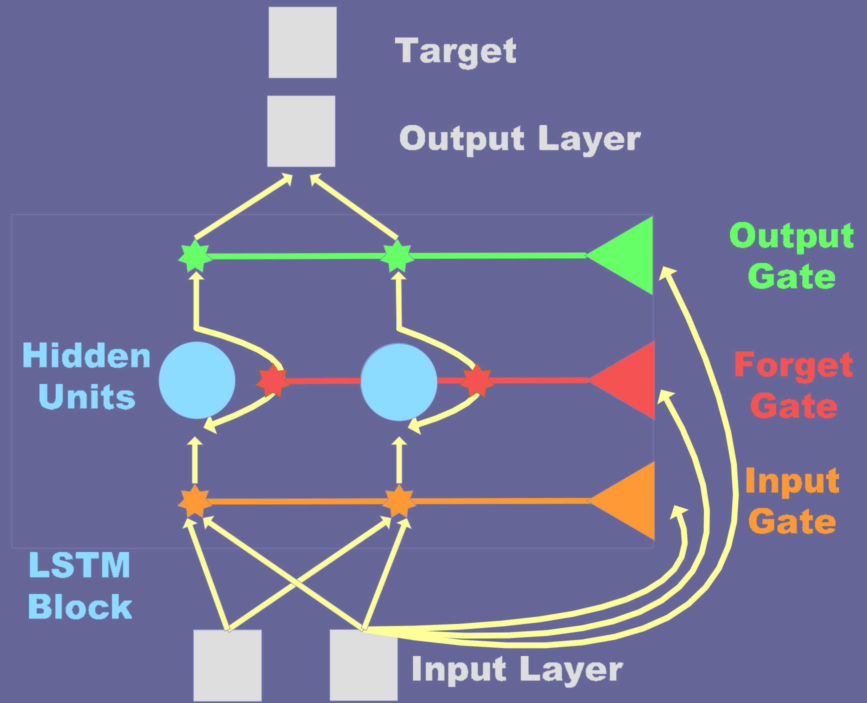


The resulting structure is called an LSTM Block



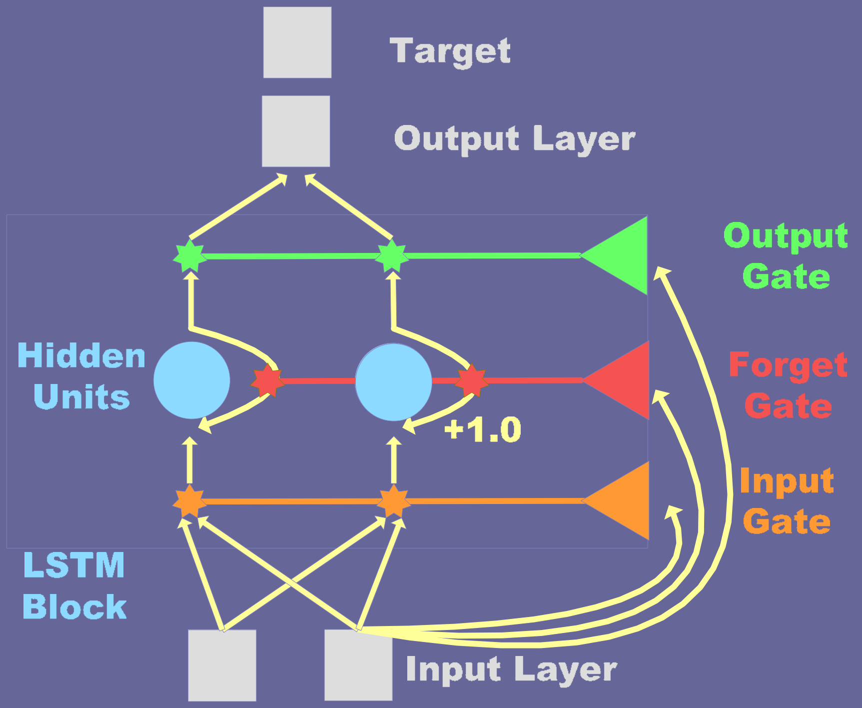
The resulting structure is called an LSTM Block

Self-recurrent connections for hidden units fixed at +1.0



The resulting structure is called an LSTM Block

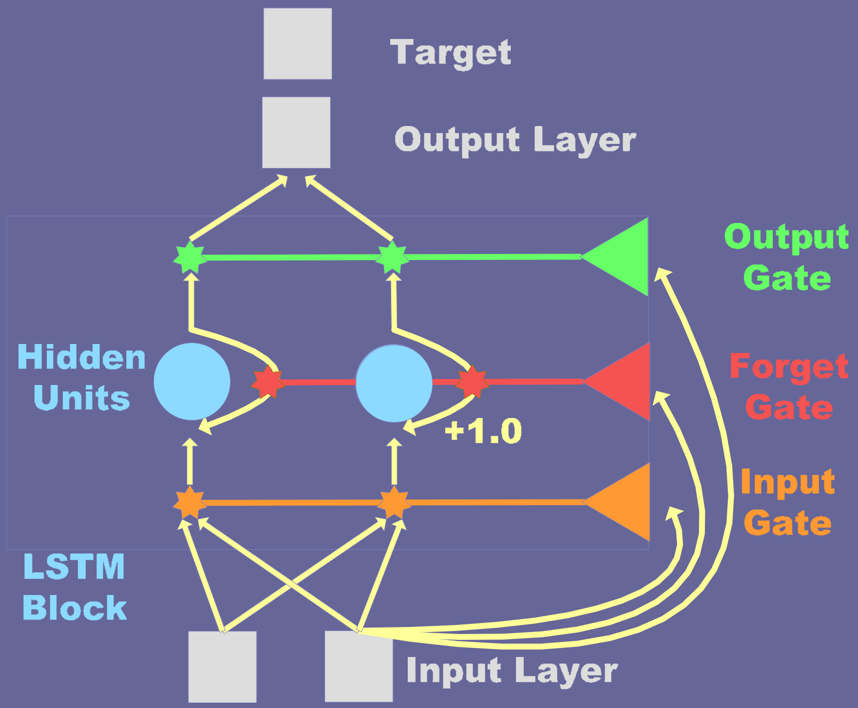
Self-recurrent connections for hidden units fixed at +1.0



The resulting structure is called an LSTM Block

Self-recurrent connections for hidden units fixed at +1.0

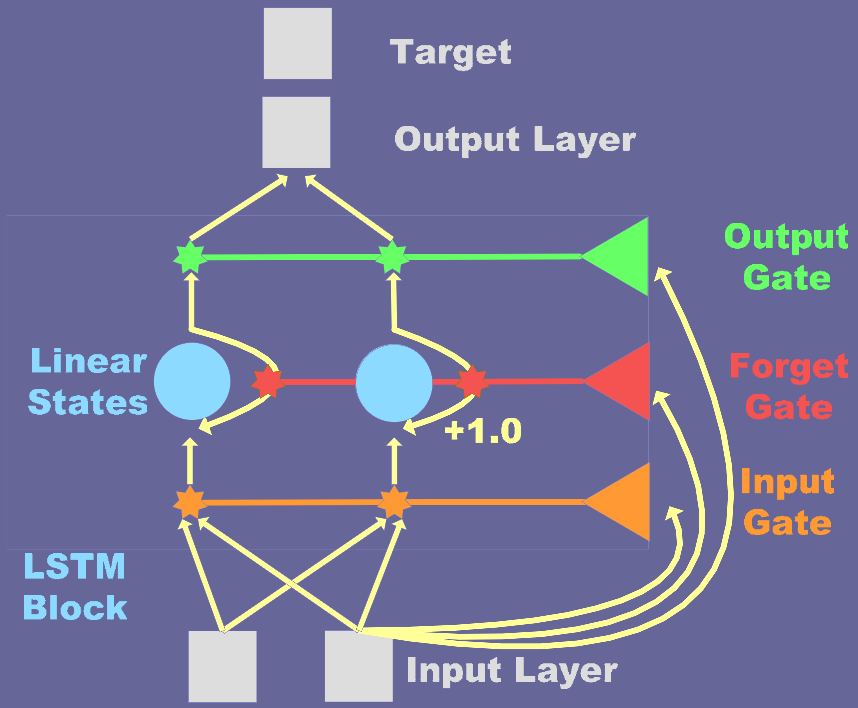
Units are not squashed ("Linear States")



The resulting structure is called an LSTM Block

Self-recurrent connections for hidden units fixed at +1.0

Units are not squashed ("Linear States")



LSTM Forward Pass

Cell Activation

$$act_{cell} = act_{out} s$$

Output Gate

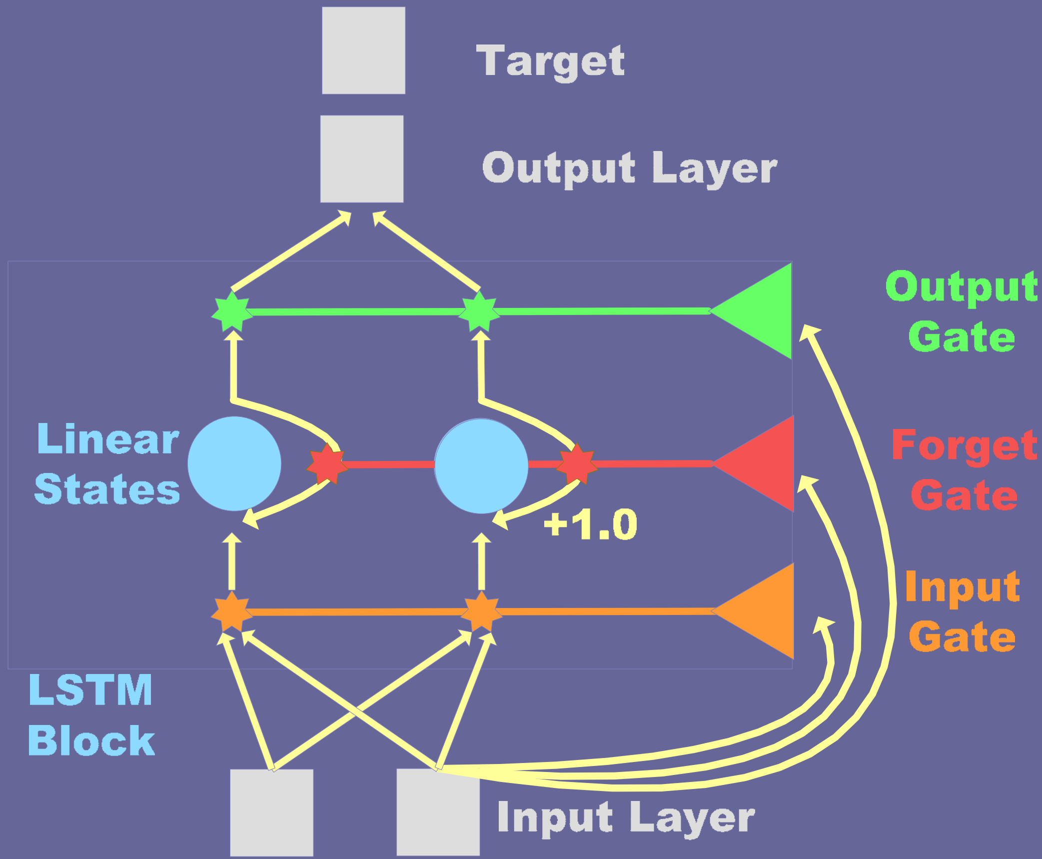
$$net_{out} = \sum_m w_{out,m} act_m$$

$$act_{out} = f(net_{out})$$

Cell State

$$net_{cell} = \sum_m w_{cell,m} act_m$$

$$s = act_{fgt} \hat{s} + act_{in} f(net_{cell})$$



Forget Gate

$$net_{fgt} = \sum_m w_{fgt,m} act_m$$

$$act_{fgt} = f(net_{fgt})$$

Input Gate

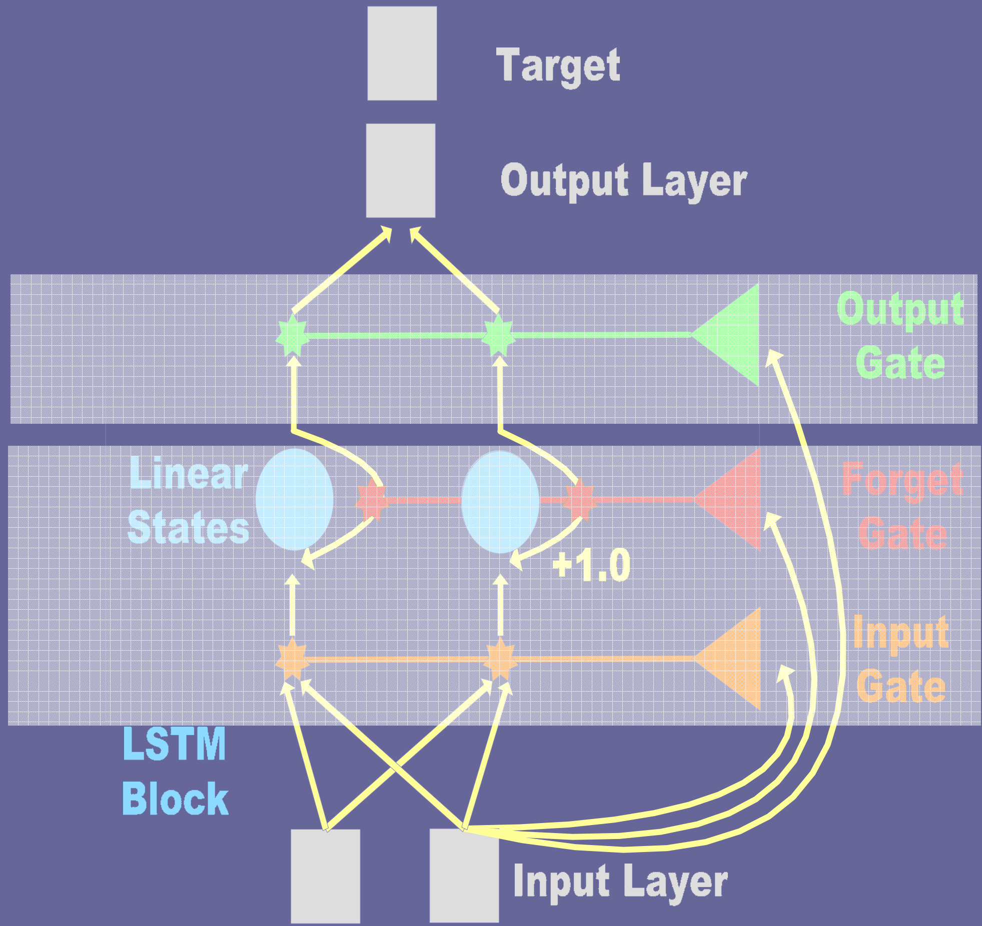
$$net_{in} = \sum_m w_{in,m} act_m$$

$$act_{in} = f(net_{in})$$

LSTM Backward Pass

Output gate uses standard backprop

Cell states, forget gates and input gates use Real Time Recurrent Learning (RTRL)-style partial derivative tables



LSTM Backward Pass

Compute errors and partials

Output Gate Error (BP)

$$\delta_{out_j} = f'_{out_j}(z_{out_j})\left(\sum_{v=1}^{S_j} s_{c_j^v} \sum_k w_{kc_j^v}\delta_k\right)$$

Cell State Error (BP)

$$e_{s_{c_j^v}} = y_{out_j}\left(\sum_k w_{kc_j^v}\delta_k\right)$$

Cell State Partials (RTRL)

$$dS_{cm}^{jv} = dS_{cm}^{jv} y_{fgt_j} + g'(z_{c_j^v}) y_{in_j} \hat{y}_m$$

Input Gate Partials (RTRL)

$$dS_{in,m}^{jv} = dS_{in,m}^{jv} y_{fgt_j} + g(z_{c_j^v}) f'_{in_j}(z_{in_j}) \hat{y}_m$$

Forget Gate Partials (RTRL)

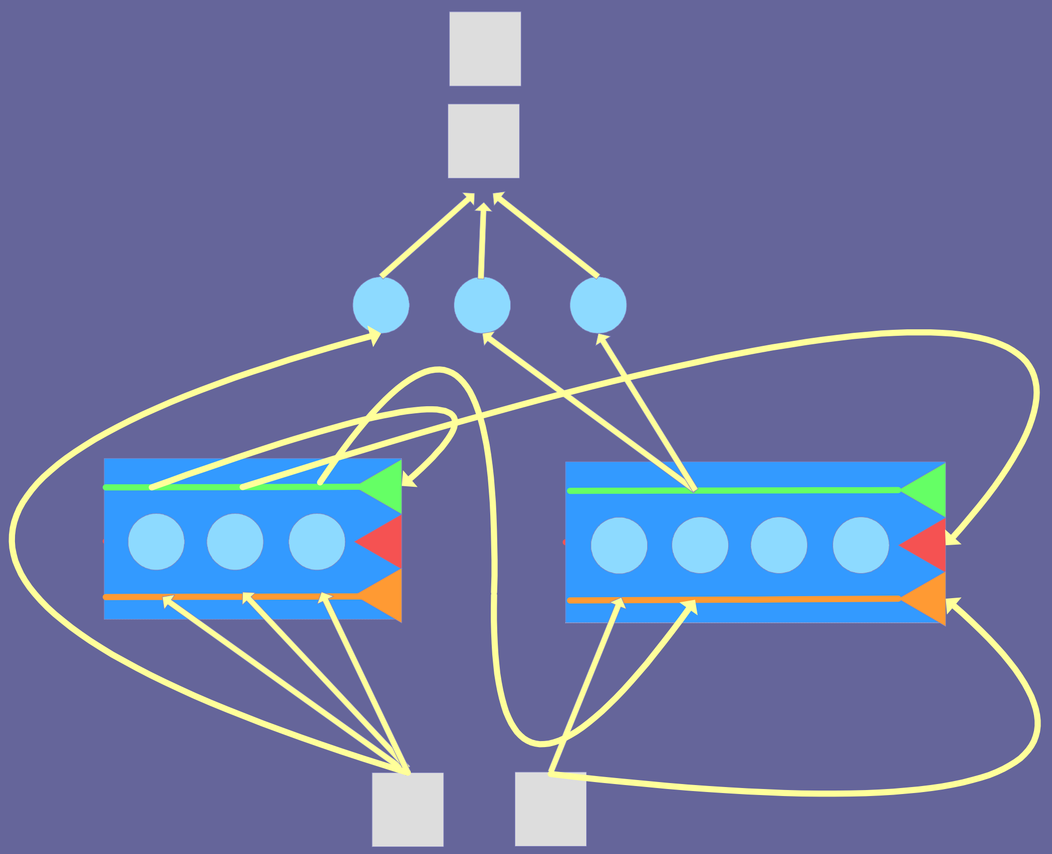
$$dS_{fgt,m}^{jv} = dS_{fgt,m}^{jv} y_{fgt_j} + \hat{s}_{c_j^v} f'_{fgt_j}(z_{fgt_j}) \hat{y}_m$$

Compute weight changes

- Weight changes into output gate proportional to output gate delta
- Weight changes into cell, input gate, forget gate proportional to partials

Connectivity

- Recurrent connections back into gates and cells very useful
- Additional feed-forward layers are useful
- Additional LSTM layers probably not useful: gradient truncated

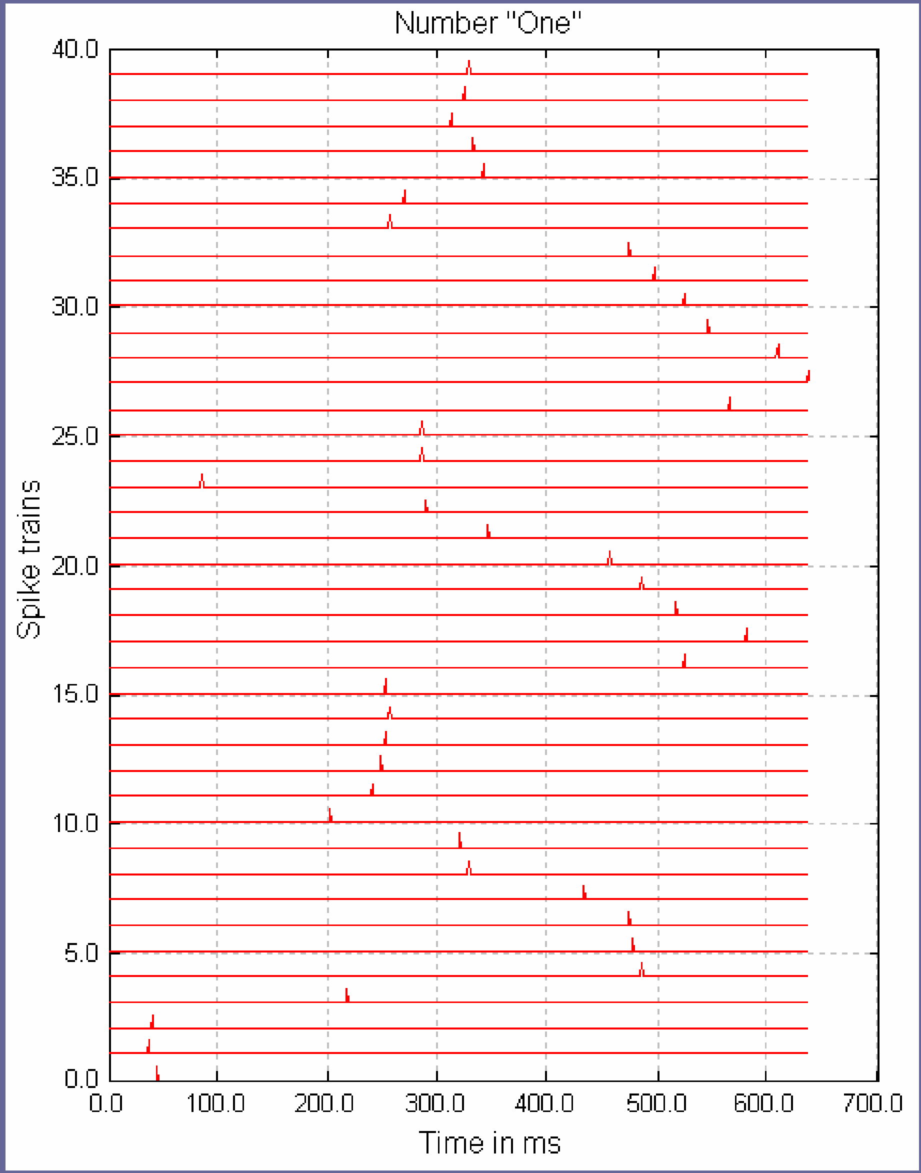


Speech Recognition

- NNs already show promise (Boulard, Robinson, Bengio)
- LSTM may offer a better solution by finding long-timescale structure in speech
- At least two areas where this may help:
  - Time warping (rate invariance)
  - Dynamic, learned model of phoneme segmentation (with little apriori knowledge)

Speech Set 1: Spoken Digits

- Mus Silicum Competition (Brody and Hopfield)
- 500 input files, each a spoken digit “one” through “ten”
- Very compressed representation:
  - 40 spike trains having either one or zero spikes per train
  - Spikes mark onsets, peaks or offsets for 40 different frequencies (100Hz to 5kHz)



Mus Silicium Task A: Identification of digits

- Learn synchrony-based model of digit prediction
- Perform predictions online
- Training set n=300, testing=200
  - Error=false negs/n_{pos} + false pos/n_{neg}
- Maass et.al. SNN-type Generic Neural Microcircuits mean 0.14, best 0.013
- LSTM mean 0.03, best 0.0 (over 25 runs)
- LSTM synchronizes internal states to spike onsets

Task B: Identification of "one" from single example

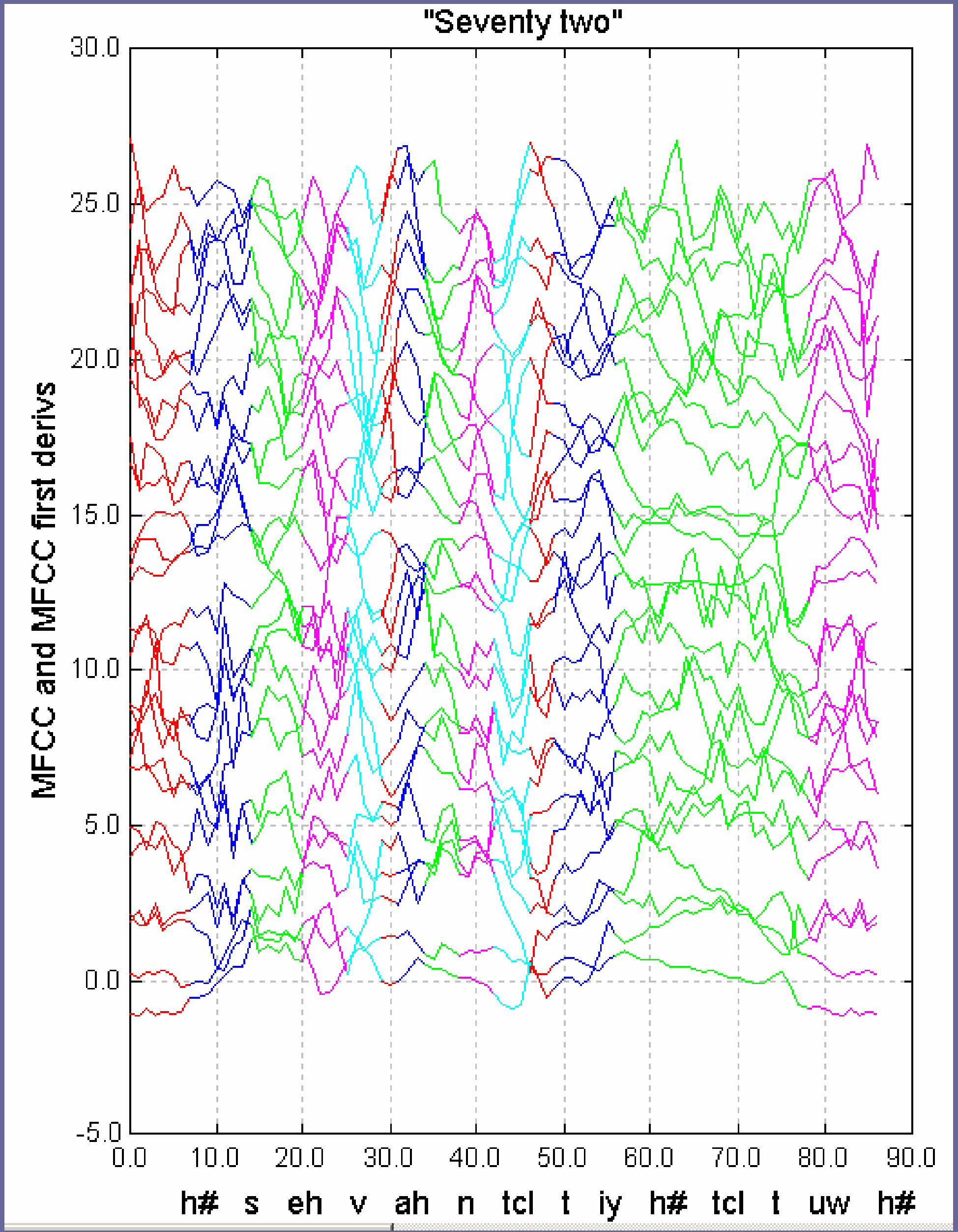
- Competition task (Hopfield and Brody)
  - 1 positive example of "one"
  - 9 randomly generated negative examples
  - Predict "one" or "not one" for dataset of size 500
- Best in competition err=0.23
- Hopfield and Brody err=0.14
- LSTM best error=0.14
  (mean over 15 runs=0.26)

Discussion

- LSTM networks much smaller
  - Hopfield and Brody: ~=5600 units
  - Maass: 135 units
  - LSTM: 50 units (10 gated blocks with 2 cells each yielding 30 gating units and 20 states)
- LSTM exhibited desired “online prediction”
- LSTM outperformed contest entrants and matched performance of Hopfield and Brody.
- By using synchrony-like mechanism, LSTM generalizes well and copes with timewarping

Speech Set 2: Phoneme Identification

- “Numbers 95” database. Numeric street addresses and zip codes (collaborator: Bengio)
- 13 MFCC values plus first derivative = 26 inputs
- 27 possible phonemes
- ~=4500 sentences
  ~=77000 phonemes
  ~= 666,000 10ms frames


"Seventy two"
MFCC and MFCC first derivs
30.0
25.0
20.0
15.0
10.0
5.0
0.0
-5.0
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
h# s eh v ah n tcl t iy h# tcl t uw h#

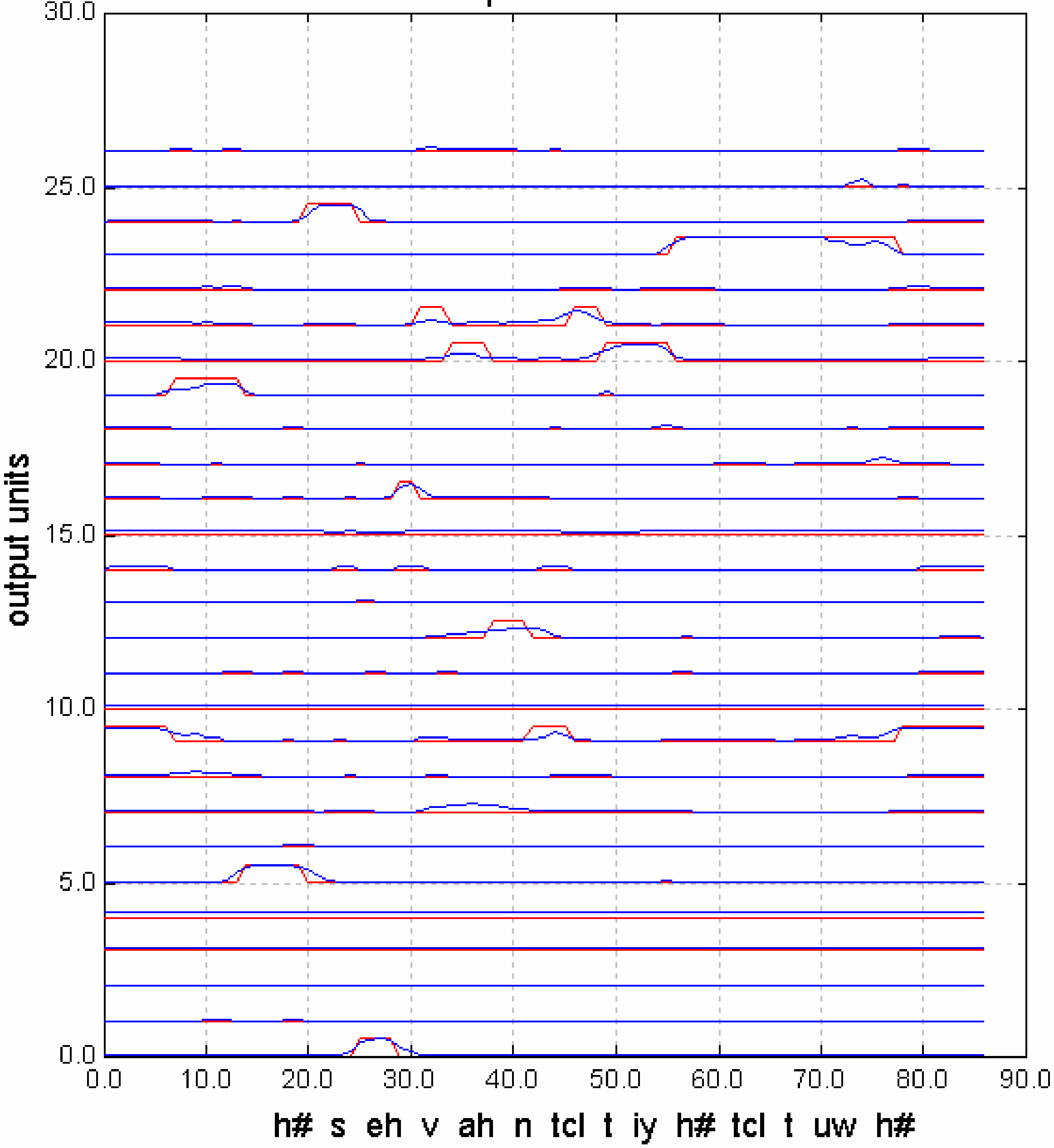
Task A: Single phoneme identification

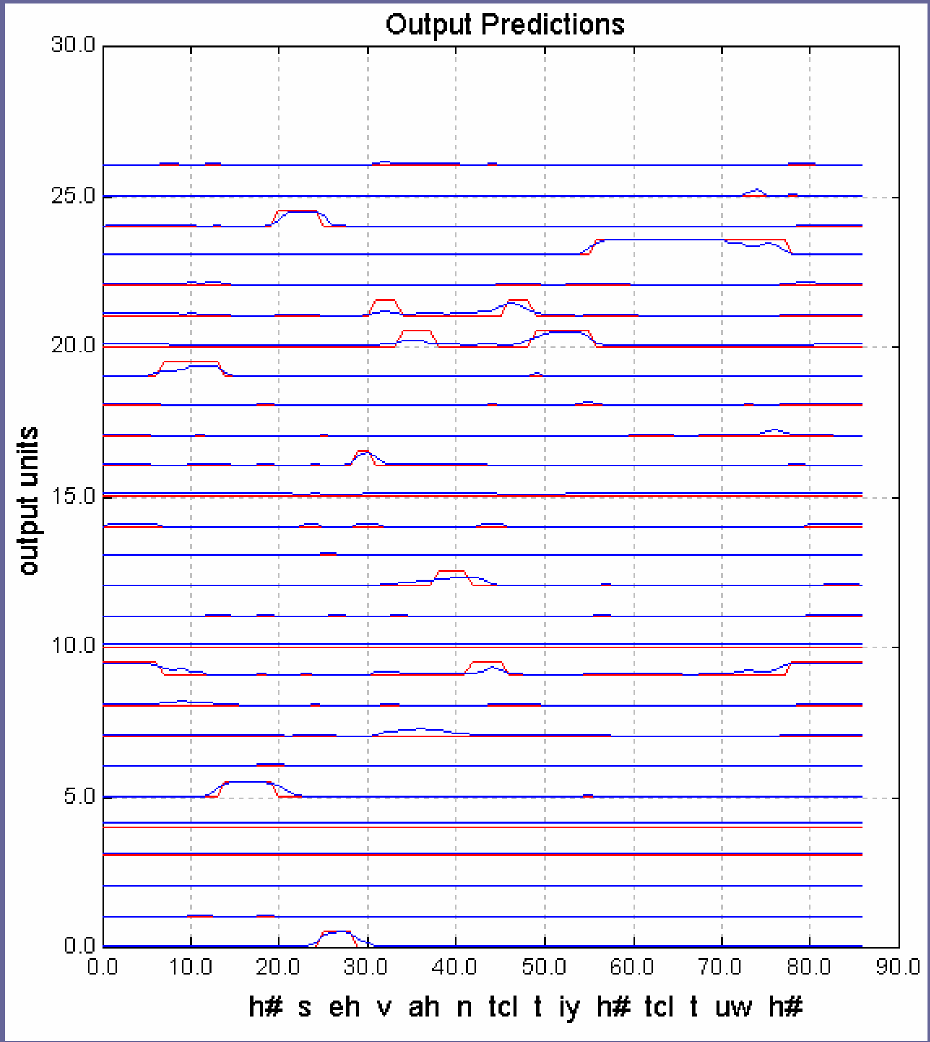
- Categorize phonemes in isolation.
- Prediction made only at last time step
- LSTM has no advantage because no history
- Benchmark ~=92% correct (S. Bengio)
- LSTM ~= 85%*

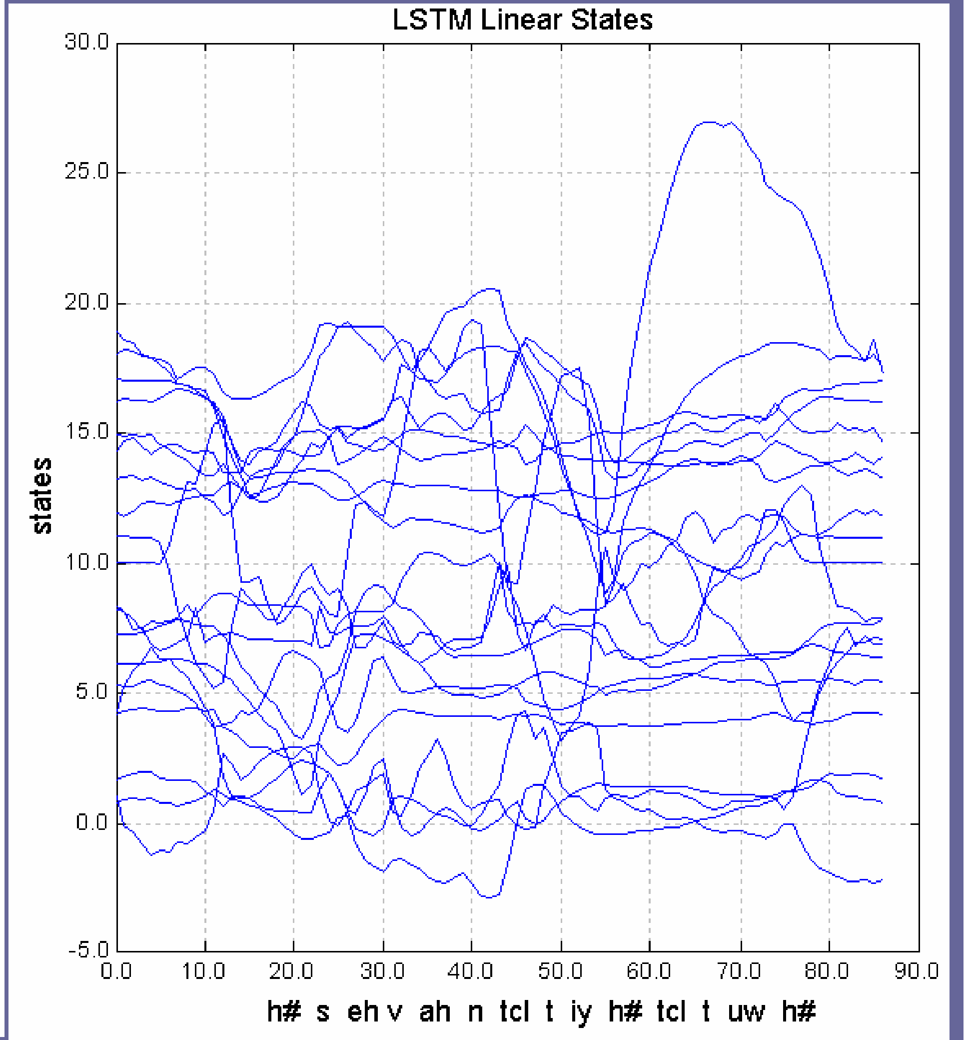
Task B: frame-level phoneme recognition

- Assign all frames to one of 27 phonemes.
- Use entire sentence
- For later phonemes, history can be exploited
- Benchmark ~= 80%
- LSTM ~= 78%*

Output Predictions







State trajectories suggest a use of history.

Alternatives to standard RNNs

- **Architectual Changes to RNNs:**
  - **Time windows:** Moving buffer over input time series
  - **TDNNs:** Delayed flow of information through net with cascaded internal delays (Haffner & Waibel)
  - **Focused Backprop:** Delay update of activations (Mozer, also deVries & Principe)
  - **NARX:** Multiple input time windows ("embedded memories") shortcut error flow (Lin et al)
  - **Reuse activations:** Update using scaled sum of old act new input (Sun)
  - **Hierarchical RNNs**: Organization time delays hierarchically: (El Hihi et. al.)
  - **Dynamic allocation** : When a unit receives conflicting error signals, add new unit (Ring)
- **Alternative Search Methods for RNNs**:
  - **Weight guessing**: Works only for easy problems (Hochreiter)
  - **Search without gradients**: Propagate discrete error (Bengio et. al.)
  - **Simulated annealing** : Controlled use of noise (Bengio for this specific issue)
  - **Genetic approaches**: (Angeline et. al.)
  - **Second-order methods**: Pseudo-Newton methods, Kalman Filters, Particle Filters
- **Non-RNN Approaches:**
  - **IO-HMMs**: Discrete networks trained with EM (Bengio & Frasconi).
  - **Spiking Neural Networks**: Use synchrony as latching mechanism.
  - **Hidden Markov Models** : Non-ergodic transition diagrams allow, e.g, left-to-right flow.