ELEC/COMP 576:
Gated Recurrent Neural Network Architectures & Applications

Ankit B. Patel
Baylor College of Medicine (Neuroscience Dept.)
Rice University (ECE Dept.)

RNN Training

Why Training is Unstable

$$x^{(l)} = W^{(l-1)} y^{(l-1)} + b^{(l-1)}$$

$$y^{(l)} = f(x^{(l)})$$

Let the activation function $f(x) = \alpha x + \beta$,

$$Var\left(y^{(l)}\right) = \alpha^2 n_{l-1} \sigma_{l-1}^2 \left(Var\left(y^{(l-1)}\right) + \beta^2 I_{n_l}\right).$$

$$Var\left(\frac{\partial cost}{\partial y^{(l-1)}}\right) = \alpha^2 n_l \sigma_{l-1}^2 Var\left(\frac{\partial cost}{\partial y^{(l)}}\right).$$

Variance of activations/gradients grows multiplicatively

[Xu, Huang, Li]

Interesting Question

- Are there modifications to an RNN such that it can combat these gradient problems?

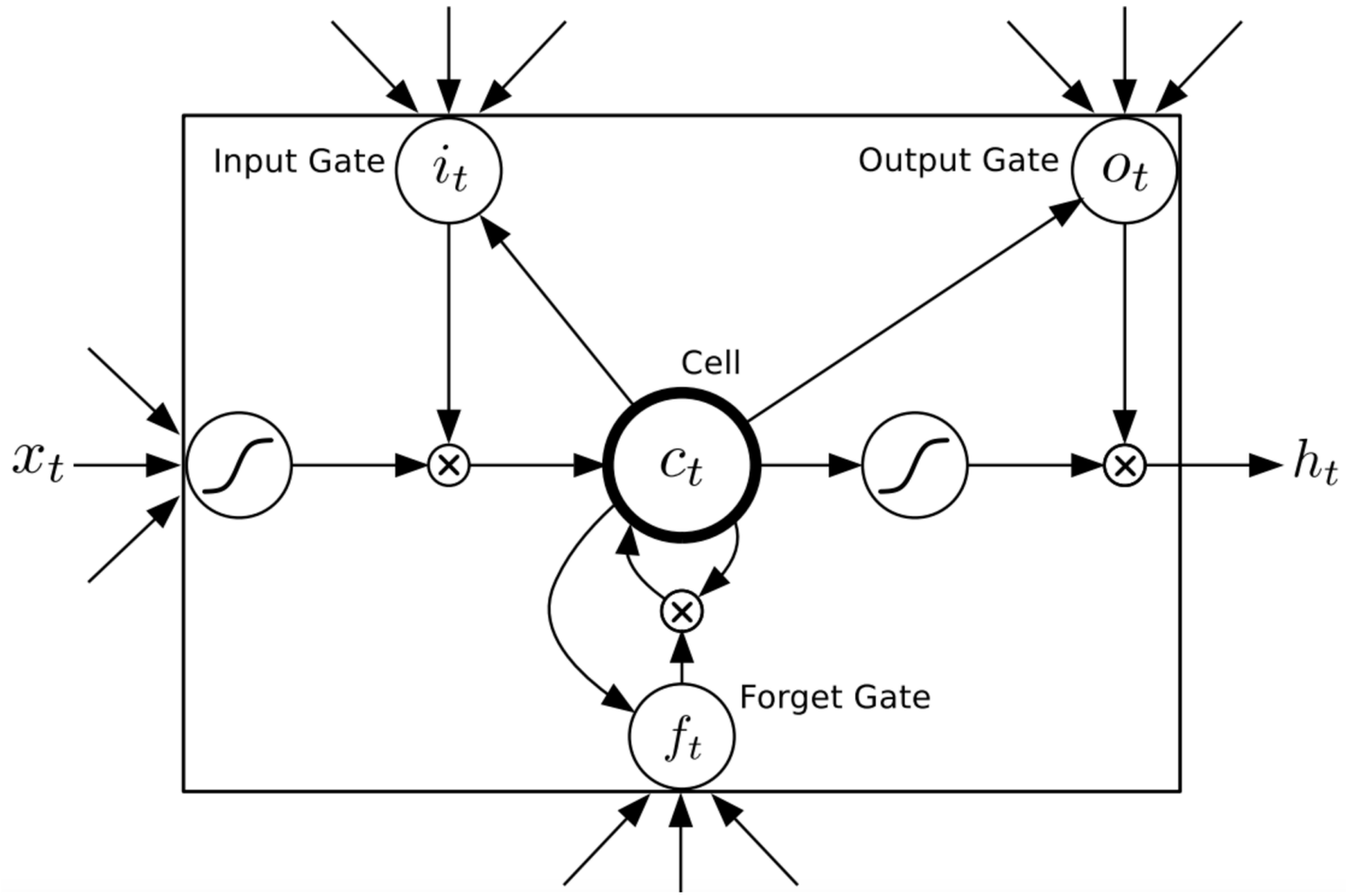
RNNs with Longer Term Memory

Motivation

- The need to remember certain events for arbitrarily long periods of time (Non-Markovian)
- The need to forget certain events

Long Short Term Memory

- 3 gates
- Input
- Forget
- Output



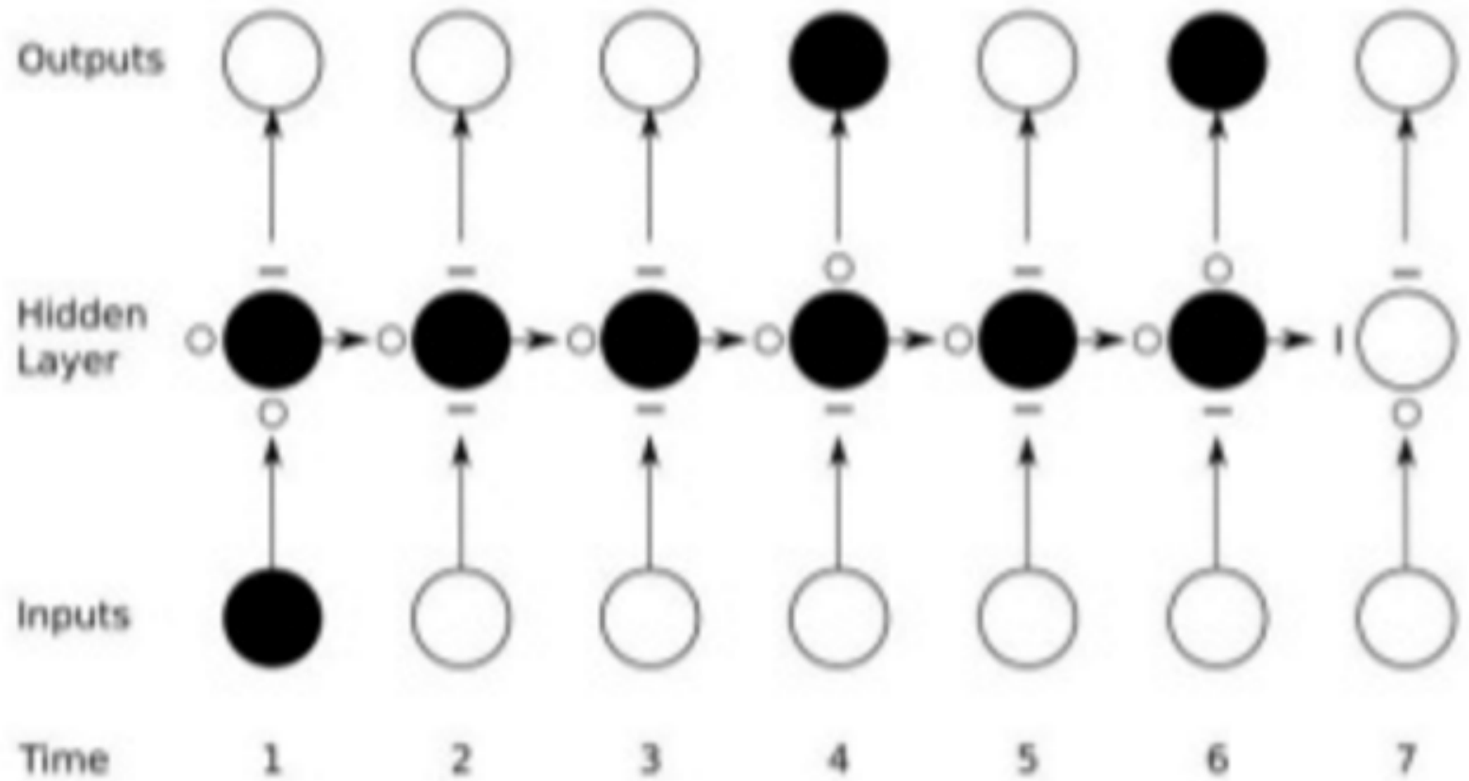
[Zygmunt Z.]

LSTM Formulation

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

[Alex Graves, Navdeep Jaitly]

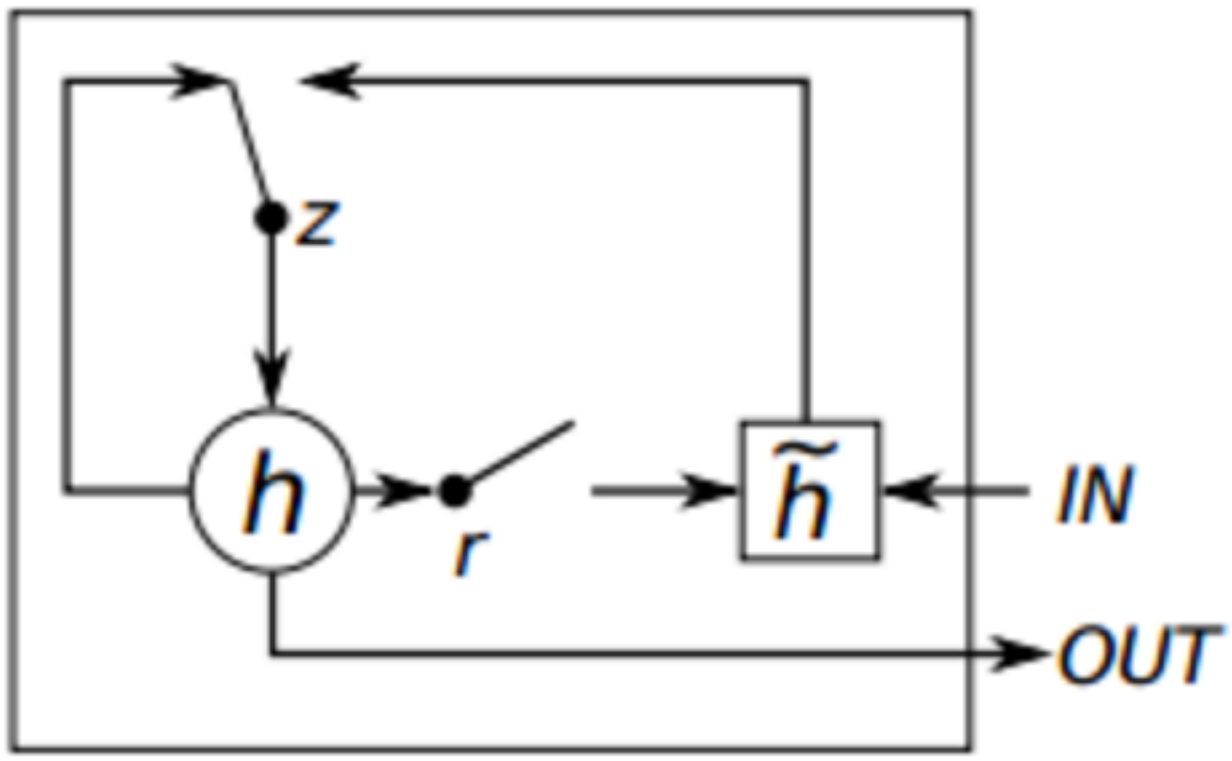
Preserving Gradients



[Hochreiter, Schmidhuber]

Gated Recurrent Unit

- 2 gates
  - Reset
    - Combine new input with previous memory
  - Update
    - How long the previous memory should stay



[Zygmunt Z.]

GRU Formulation

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$
$$r = \sigma(x_t U^r + s_{t-1} W^r)$$
$$h = tanh(x_t U^h + (s_{t-1} \circ r) W^h)$$
$$s_t = (1 - z) \circ h + z \circ s_{t-1}$$

[Danny Britz]

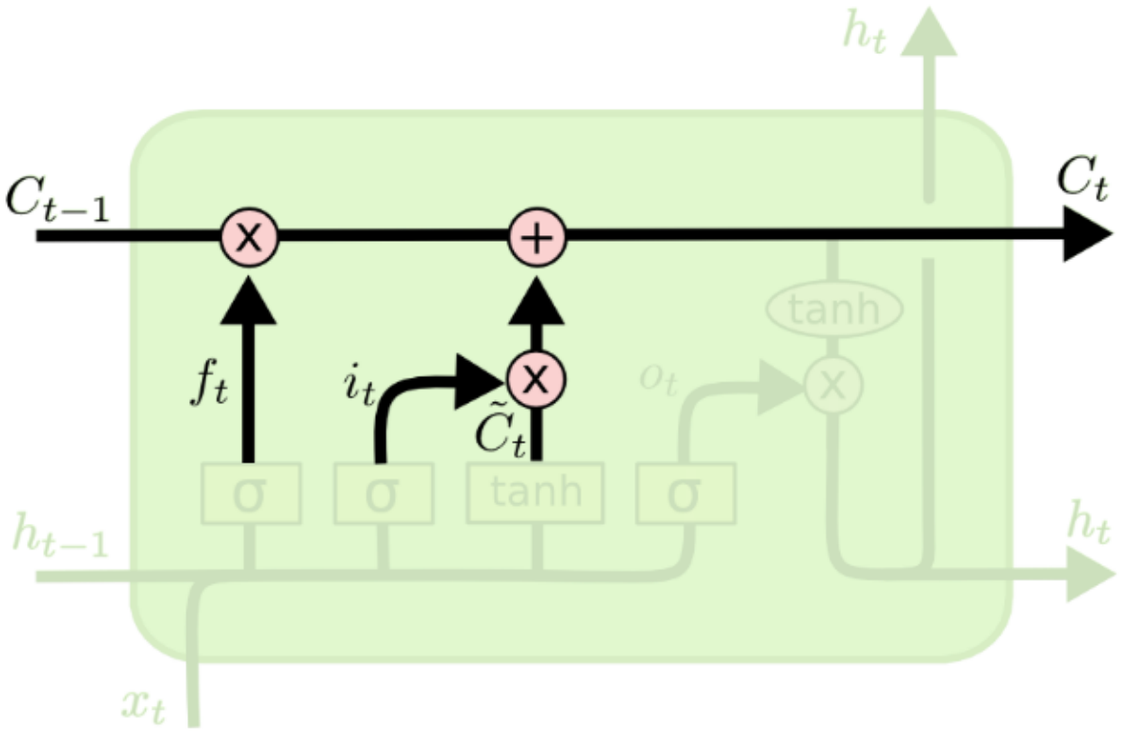
LSTM & GRU Benefits

- Remember for longer temporal durations
  - RNN has issues for remembering longer durations
- Able to have feedback flow at different strengths depending on inputs

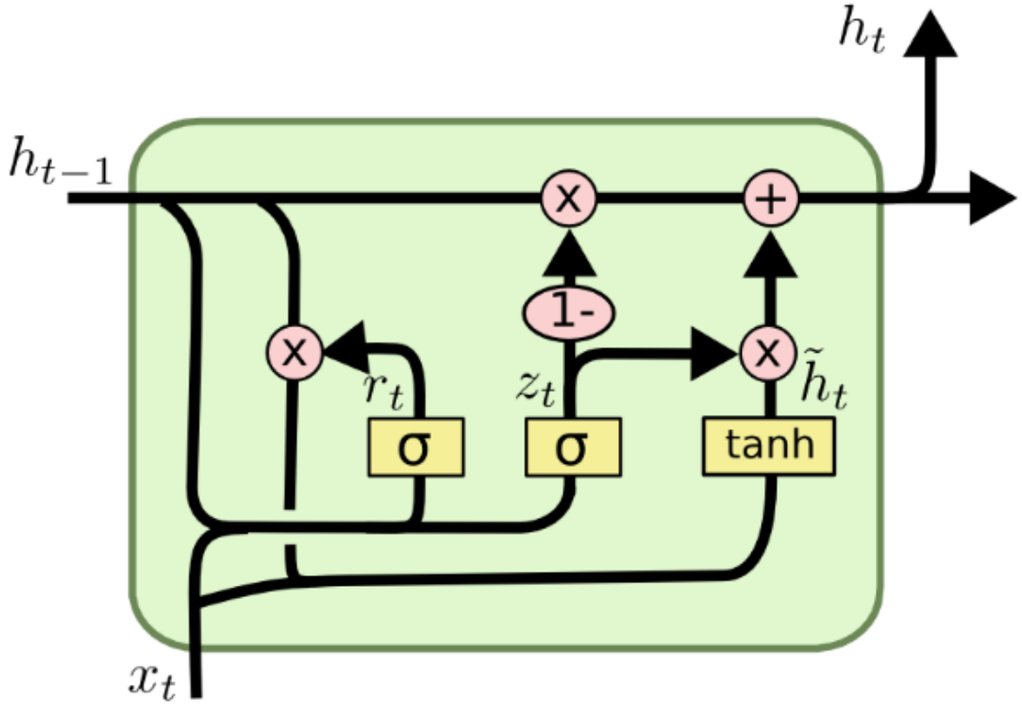
Differences between LSTM & GRU

- GRU has two gates, while LSTM has three gates
- GRU does not have internal memory
- GRU does not use a second nonlinearity for computing the output

Visual Difference of LSTM & GRU



LSTM



GRU

[Chris Olah]

LSTM vs GRU Results

			tanh	GRU	LSTM
Music Datasets	Nottingham	train	3.22	2.79	3.08
		test	**3.13**	3.23	3.20
	JSB Chorales	train	8.82	6.94	8.15
		test	9.10	**8.54**	8.67
	MuseData	train	5.64	5.06	5.18
		test	6.23	**5.99**	6.23
	Piano-midi	train	5.64	4.93	6.49
		test	9.03	**8.82**	9.03
Ubisoft Datasets	Ubisoft dataset A	train	6.29	2.31	1.44
		test	6.44	3.59	**2.70**
	Ubisoft dataset B	train	7.61	0.38	0.80
		test	7.62	**0.88**	1.26

[Chung, Gulcehre, Cho, Bengio]

Other Methods for Stabilizing RNN Training

Why Training is Unstable

$$x^{(l)} = W^{(l-1)} y^{(l-1)} + b^{(l-1)}$$

$$y^{(l)} = f(x^{(l)})$$

Let the activation function $f(x) = \alpha x + \beta$,

$$Var\left(y^{(l)}\right) = \alpha^2 n_{l-1} \sigma_{l-1}^2 \left(Var\left(y^{(l-1)}\right) + \beta^2 I_{n_l}\right).$$

$$Var\left(\frac{\partial cost}{\partial y^{(l-1)}}\right) = \alpha^2 n_l \sigma_{l-1}^2 Var\left(\frac{\partial cost}{\partial y^{(l)}}\right).$$

Variance of activations/gradients grows multiplicatively

[Xu, Huang, Li]

Stabilizing Activations & Gradients

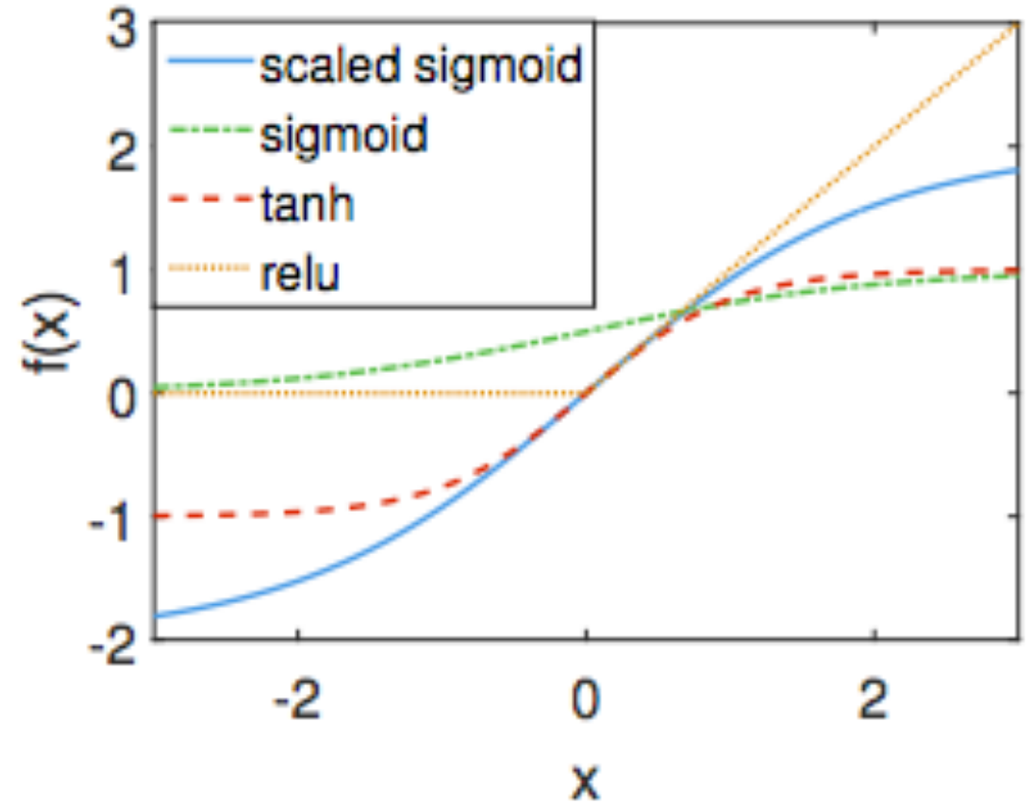
$$\mathbf{Var}\left(y^{(l)}\right) = \mathbf{Var}\left(y^{(l-1)}\right) \quad \textbf{and} \quad \mathbf{Var}\left(\frac{\partial \text{cost}}{\partial y^{(l)}}\right) = \mathbf{Var}\left(\frac{\partial \text{cost}}{\partial y^{(l-1)}}\right);$$

$$n_l \sigma_{l-1}^2 \approxeq 1 \quad \textbf{and} \quad n_{l-1} \sigma_{l-1}^2 \approxeq 1;$$

We want $\alpha = 1$ **and** $\beta = 0$.

[Xu, Huang, Li]

Taylor Expansions of Different Activation Functions



$$\text{sigmoid}(x) = \frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + O(x^5)$$

$$\tanh(x) = 0 + x - \frac{x^3}{3} + O(x^5)$$

$$\text{relu}(x) = 0 + x \quad \text{for } x \geq 0.$$

[Xu, Huang, Li]

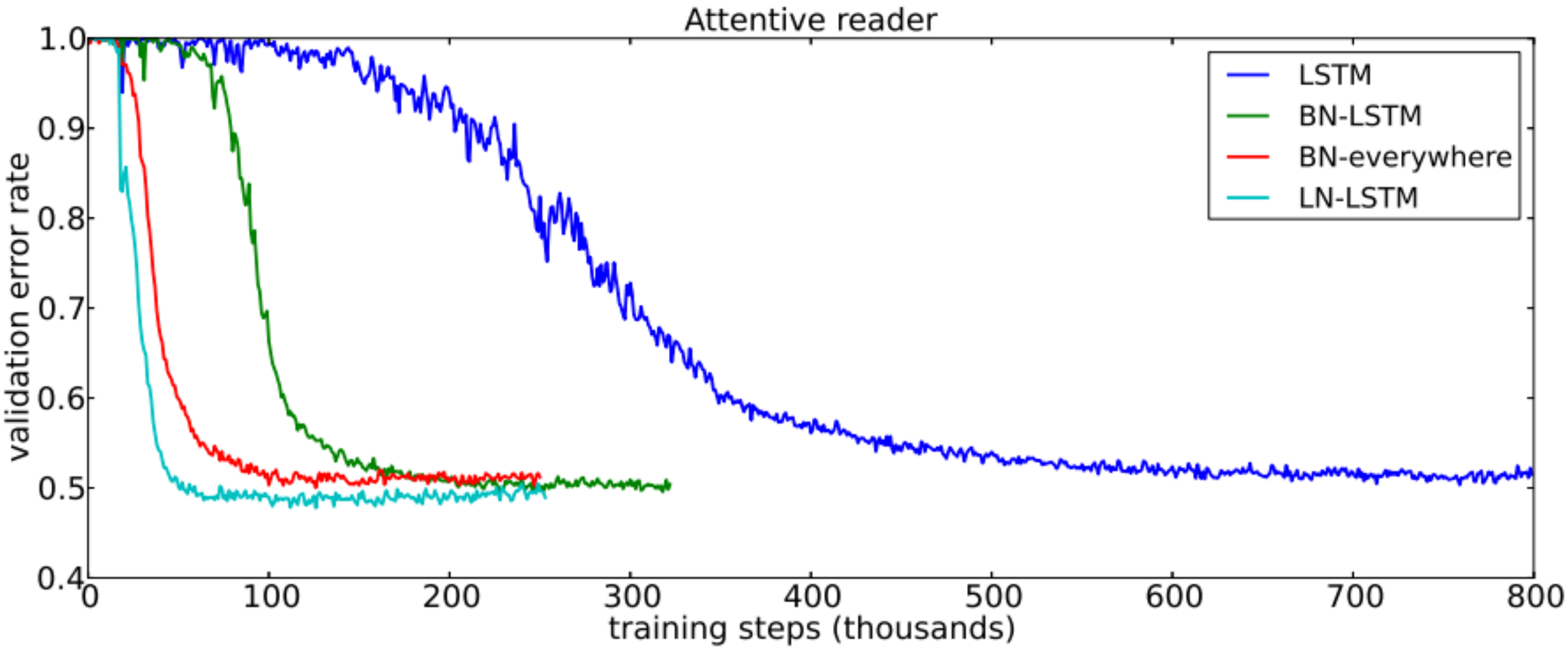
Layer Normalization

- Similar to batch normalization
  - Apply it to RNNs to stabilize the hidden state dynamics

$$\mathbf{h}^t = f\left[\frac{\mathbf{g}}{\sigma^t} \odot \left(\mathbf{a}^t - \mu^t\right) + \mathbf{b}\right] \qquad \mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \qquad \sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^t - \mu^t\right)^2}$$

[Ba, Kiros, Hinton]

Layer Normalization Results



[Ba, Kiros, Hinton]

Streamlining Normalization

Algorithm 5 Streaming Normalization Layer: Forward

Require: layer input $\mathbf{x}$ (a mini-batch), NormOP N(.,.), function S(.) to compute NormStats for every element of $\mathbf{x}$, running estimates of NormStats and/or related information packed in a structure/table H_1, function F to update H_1 and generate current estimate of NormStats $\hat{\mathrm{s}}$.

Ensure: layer output $\mathbf{y}$ (a mini-batch) and H_1 (it is stored in this layer, instead of feeding to other layers), always maintain the latest $\hat{\mathrm{s}}$ in case of testing

if *training* **then**

$\mathrm{s} = \mathrm{S}(\mathbf{x})$

$\{H_1, \hat{\mathrm{s}}\} = \mathrm{F}(H_1, \mathrm{s})$

$\mathrm{y} = \mathrm{N}(\mathbf{x}, \hat{\mathrm{s}})$

else {*testing*}

$\mathrm{y} = \mathrm{N}(\mathbf{x}, \hat{\mathrm{s}})$

end if

[Liao, Kawaguchi, Poggio]

Streamlining Normalization

Algorithm 6 Streaming Normalization Layer: Backpropagation

Require: $\frac{\partial E}{\partial y}$ (a mini-batch) where E is objective, layer input $\mathbf{x}$ (a mini-batch), NormOP N(.,.), function S(.) to compute NormStats for every element of $\mathbf{x}$, running estimates of NormStats $\hat{\mathrm{s}}$, running estimates of gradients and/or related information packed in a structure/table H_2, function G to update H_2 and generate the current estimates of gradients of NormStats $\frac{\partial E}{\partial \hat{s}}$.

Ensure: $\frac{\partial E}{\partial x}$ (a mini-batch) and H_2 (it is stored in this layer, instead of feeding to other layers)

$\frac{\partial E}{\partial \hat{s}}$ is calculated using chain rule.

$\{H_2, \widehat{\frac{\partial E}{\partial \hat{s}}}\} = \mathrm{G}(H_2, \frac{\partial E}{\partial \hat{s}})$

Use $\widehat{\frac{\partial E}{\partial \hat{s}}}$ for further backpropagation, instead of $\frac{\partial E}{\partial \hat{s}}$

$\frac{\partial E}{\partial x}$ is calculated using chain rule.

[Liao, Kawaguchi, Poggio]

Streamlining Normalization for RNNs

Normalized RNN

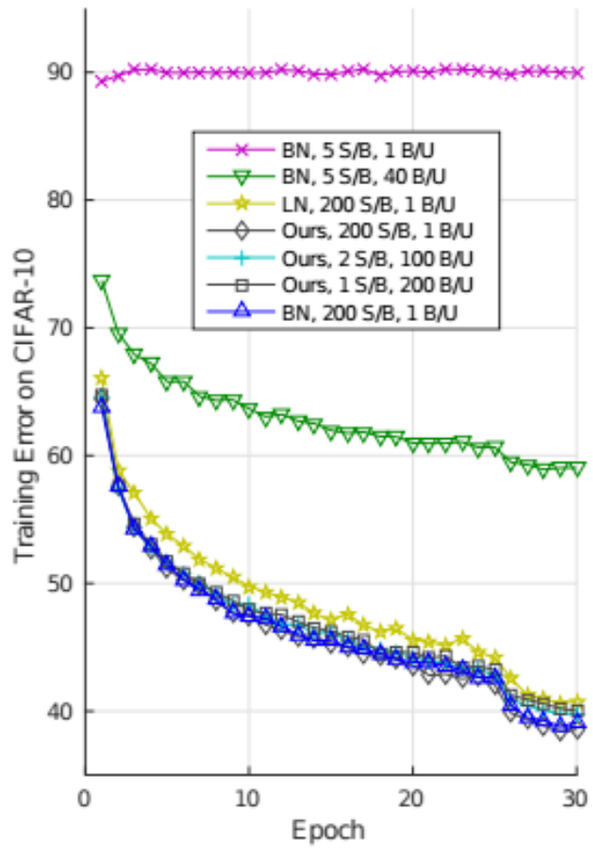
$$h_t = NonLinear(Norm(W_x * x_t) + Norm(W_h * h_{t-1}))$$

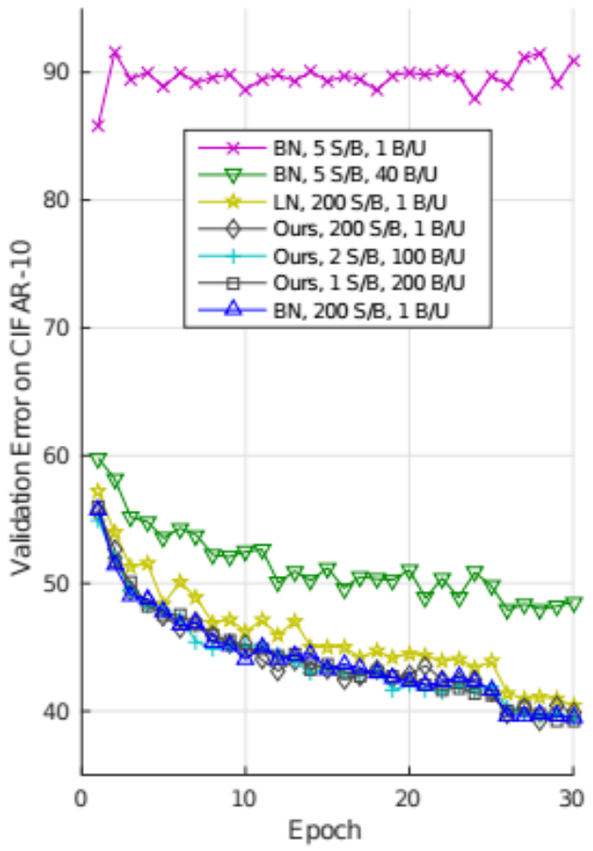
Normalized GRU

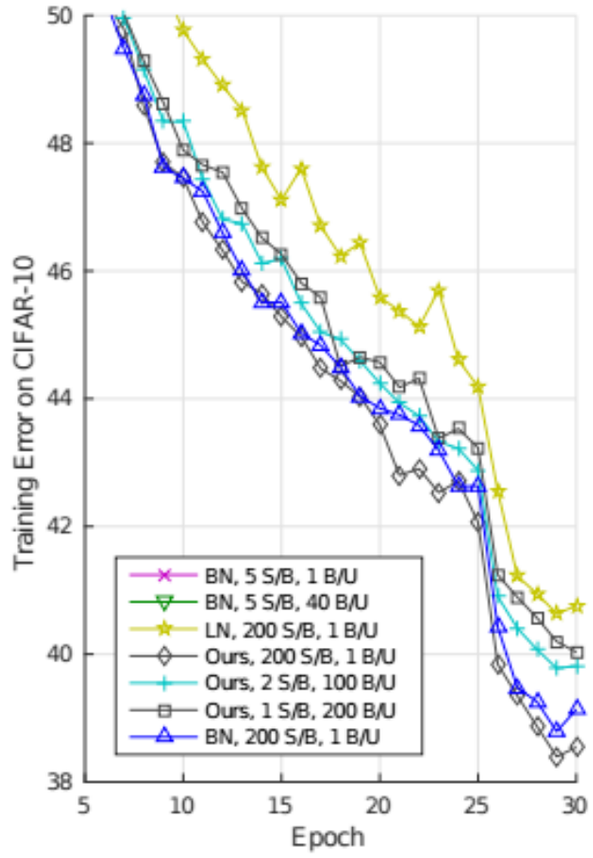
$$g_r = Sigmoid(Norm(W_{xr} * x_t) + Norm(W_{hr} * h_{t-1}))$$
$$g_z = Sigmoid(Norm(W_{xz} * x_t) + Norm(W_{hz} * h_{t-1}))$$
$$h_{new} = NonLinear(Norm(W_{xh} * x_t) + Norm(W_{hh} * (h_{t-1} \odot g_r)))$$
$$h_t = g_z \odot h_{new} + (1 - g_z) \odot h_{t-1}$$

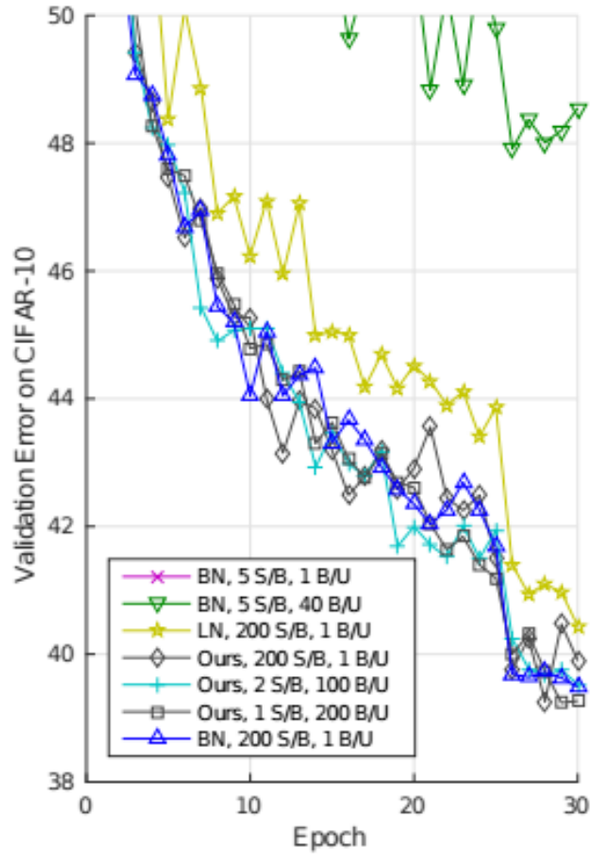
[Liao, Kawaguchi, Poggio]

Streamlining Normalization Results



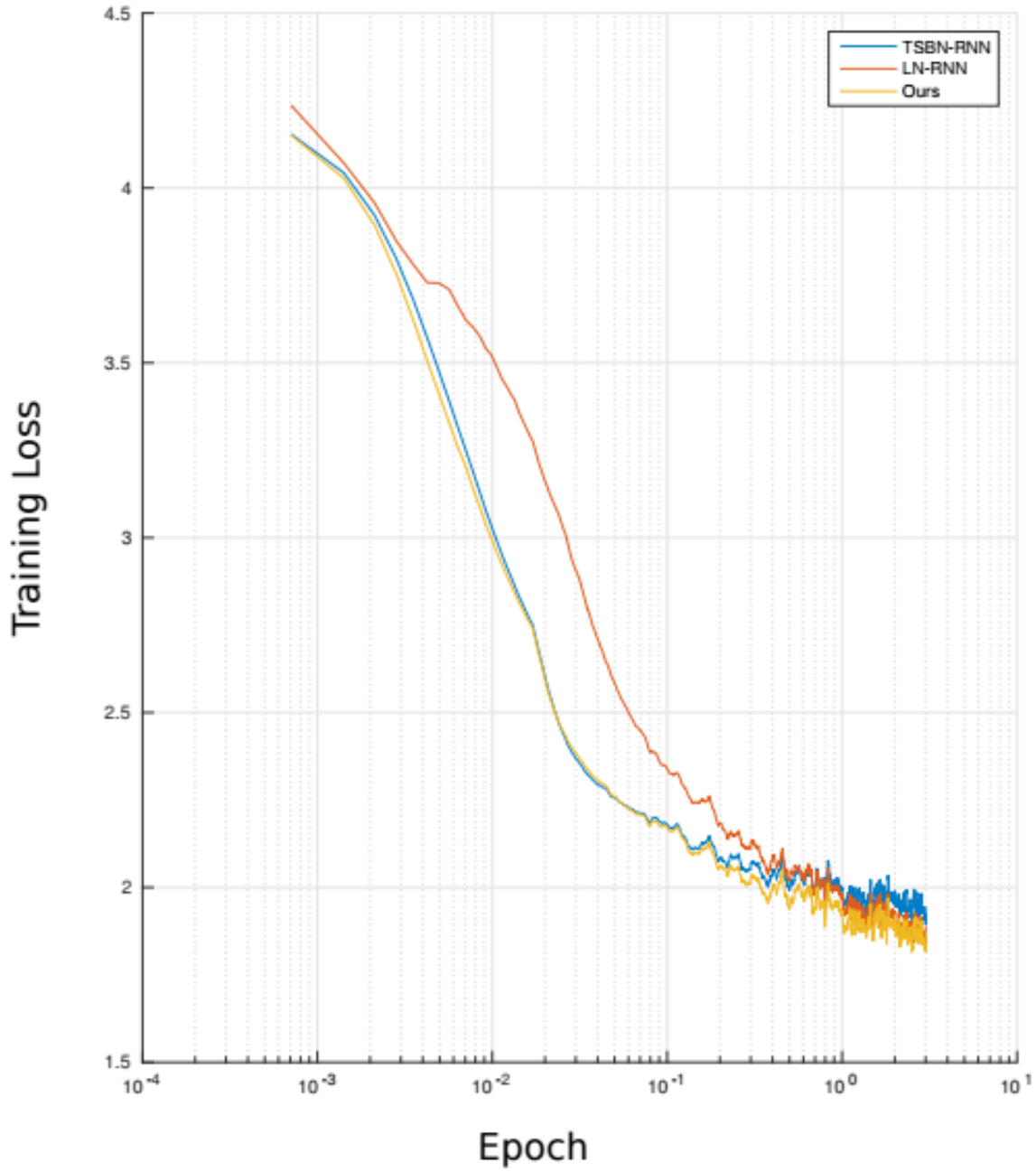


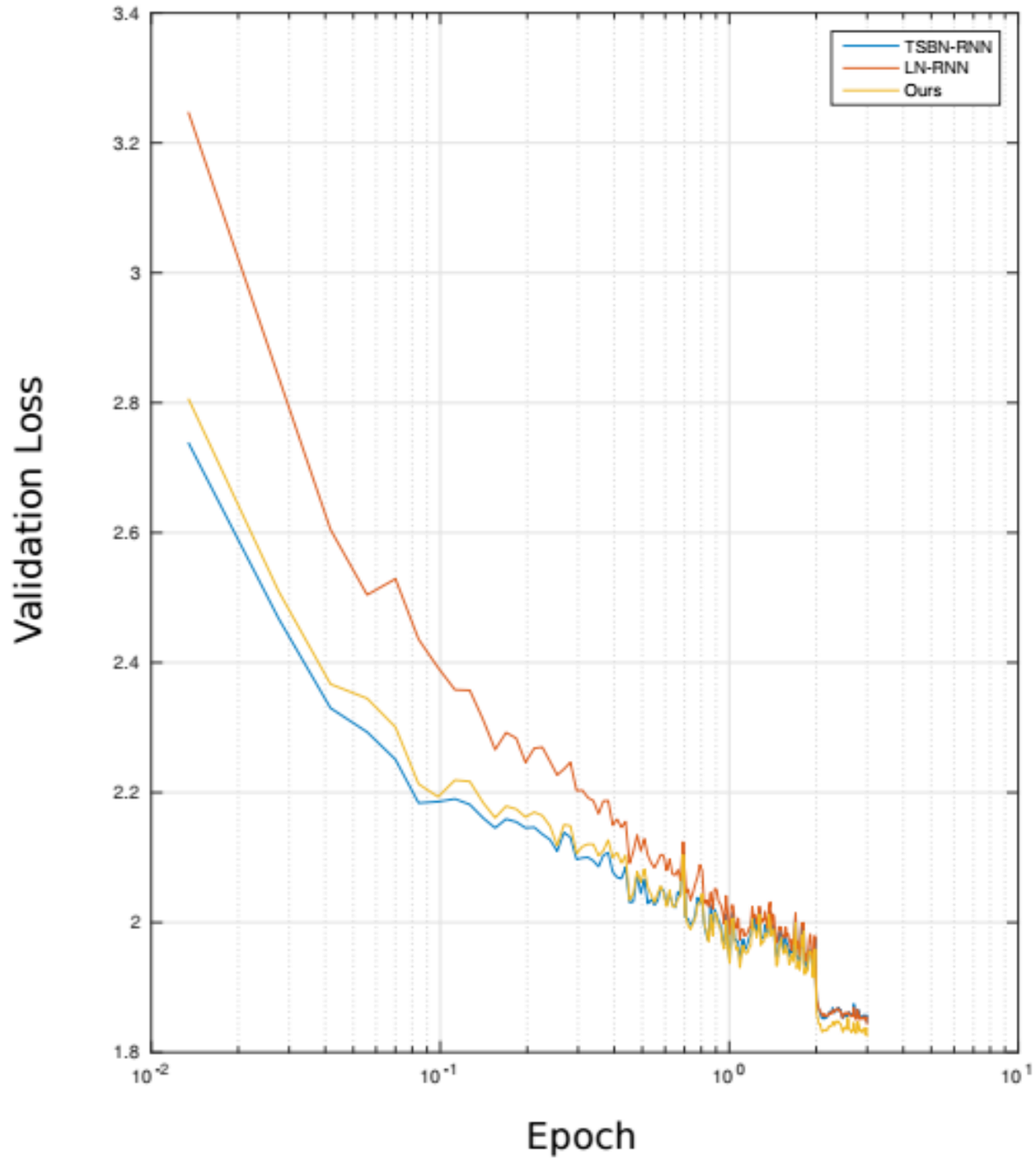




[Liao, Kawaguchi, Poggio]

Streamlining Normalization RNN Results





[Liao, Kawaguchi, Poggio]

Normalization Techniques

Approach	FF & FC	FF & Conv	Rec & FC	Rec & Conv	Online Learning	Small Batch	All Combined
Original Batch Normalization(BN)	✓	✓	✗	✗	✗	Suboptimal	✗
Time-specific BN	✓	✓	Limited	Limited	✗	Suboptimal	✗
Layer Normalization	✓	✗*	✓	✗*	✓	✓	✗*
Streaming Normalization	✓	✓	✓	✓	✓	✓	✓

Table 1: An overview of normalization techiques for different tasks. ✓: works well. ✗: does not work well. FF: Feedforward. Rec: Recurrent. FC: Fully-connected. Conv: convolutional. Limited: time-specific BN requires recording normalization statistics for each timestep and thus may not generalize to novel sequence length. *Layer normalization does not fail on these tasks but perform significantly worse than the best approaches.

[Liao, Kawaguchi, Poggio]

Applications

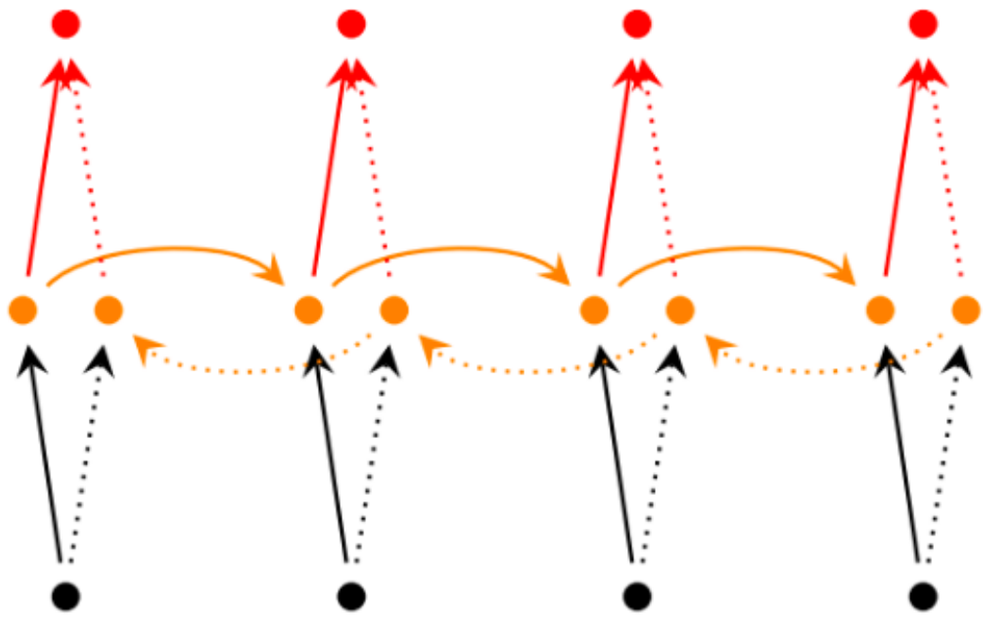
RNN Applications

- Speech Recognition
- Natural Language Processing
- Action Recognition
- Machine Translation
- Many more to come

Bidirectional RNNs

- The output at time *t* does not depend on previous time steps but also the future

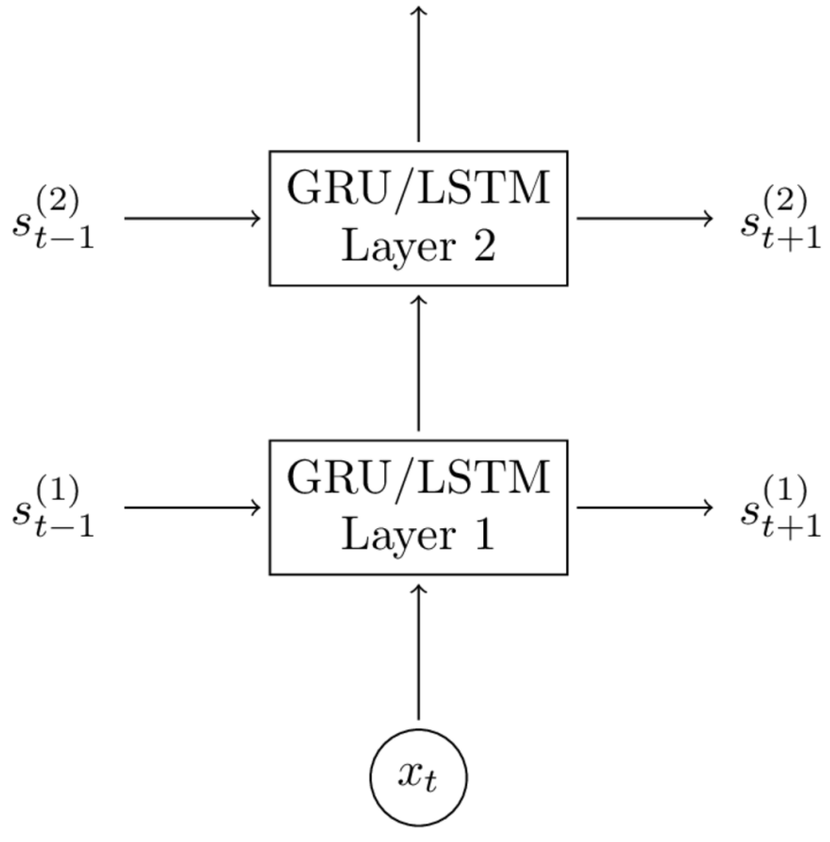
  - Two RNNs stacked on top of each other



[Danny Britz]

Deep RNNs

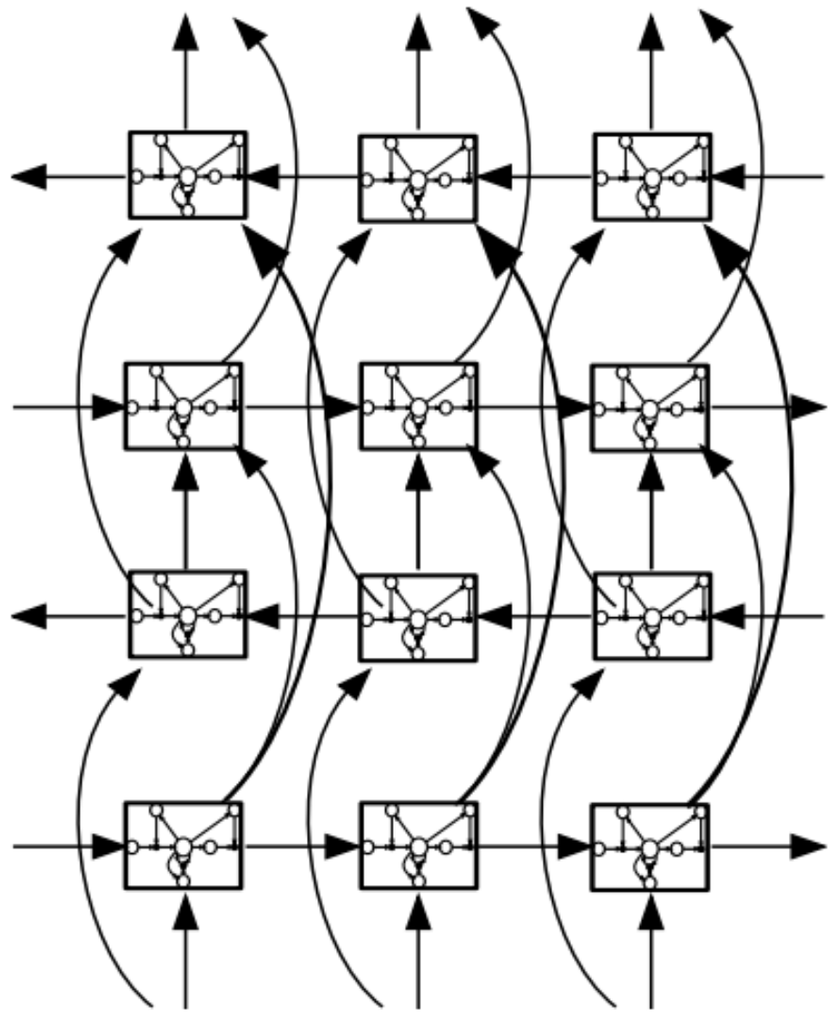
- Stack them on top of each other
  - The output of the previous RNN is the input to the next one



[Danny Britz]

Speech Recognition

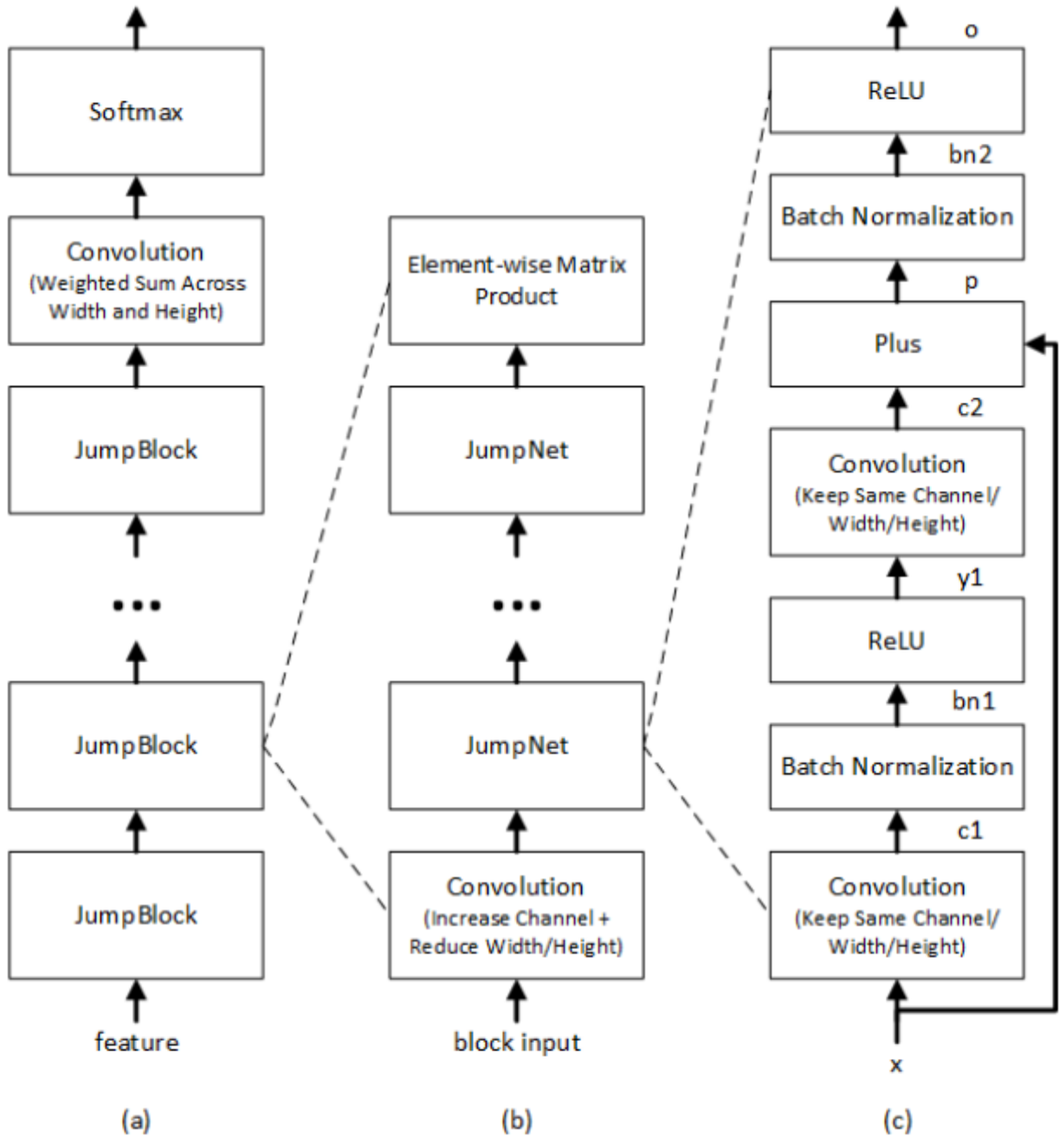
- Infer phonemes from the past and the future
  - Bidirectional Stacked LSTM



[Alex Graves, Navdeep Jaitly, Abdel-rahman Mohamed]

Conversational Speech Recognition

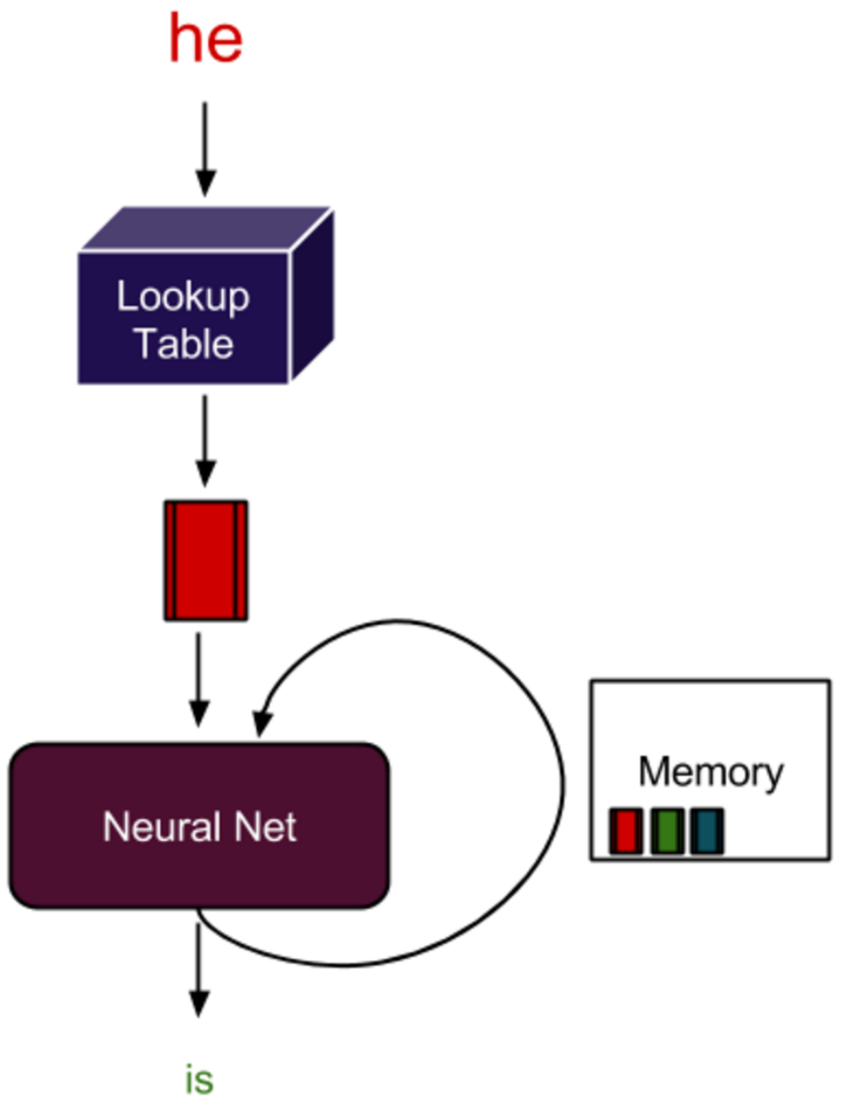
- Achieving human parity
- Combining RNN and CNN



[Xiong et al.]

Natural Language Processing

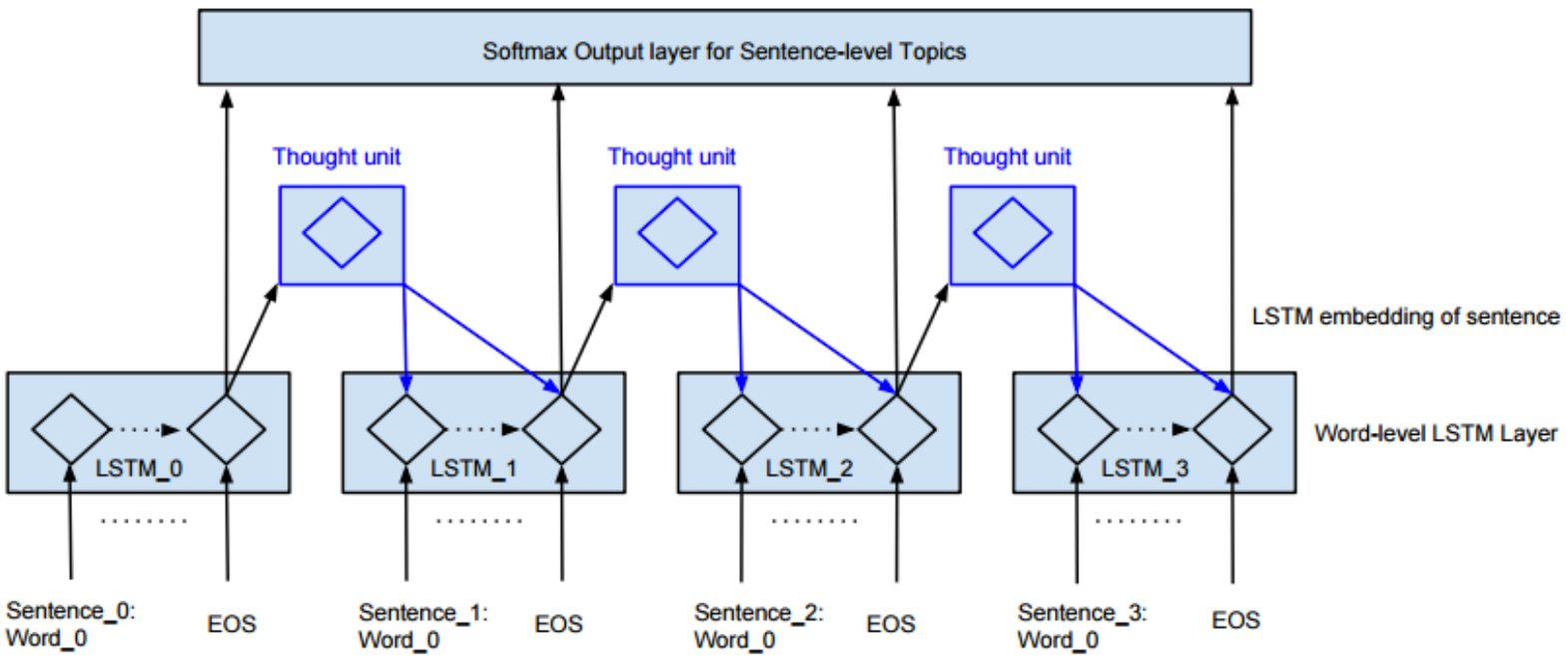
- Infer character distribution from past characters in the sequence
- Remember for longer durations



[Soumith Chantala]

Contextual LSTM for NLP Tasks

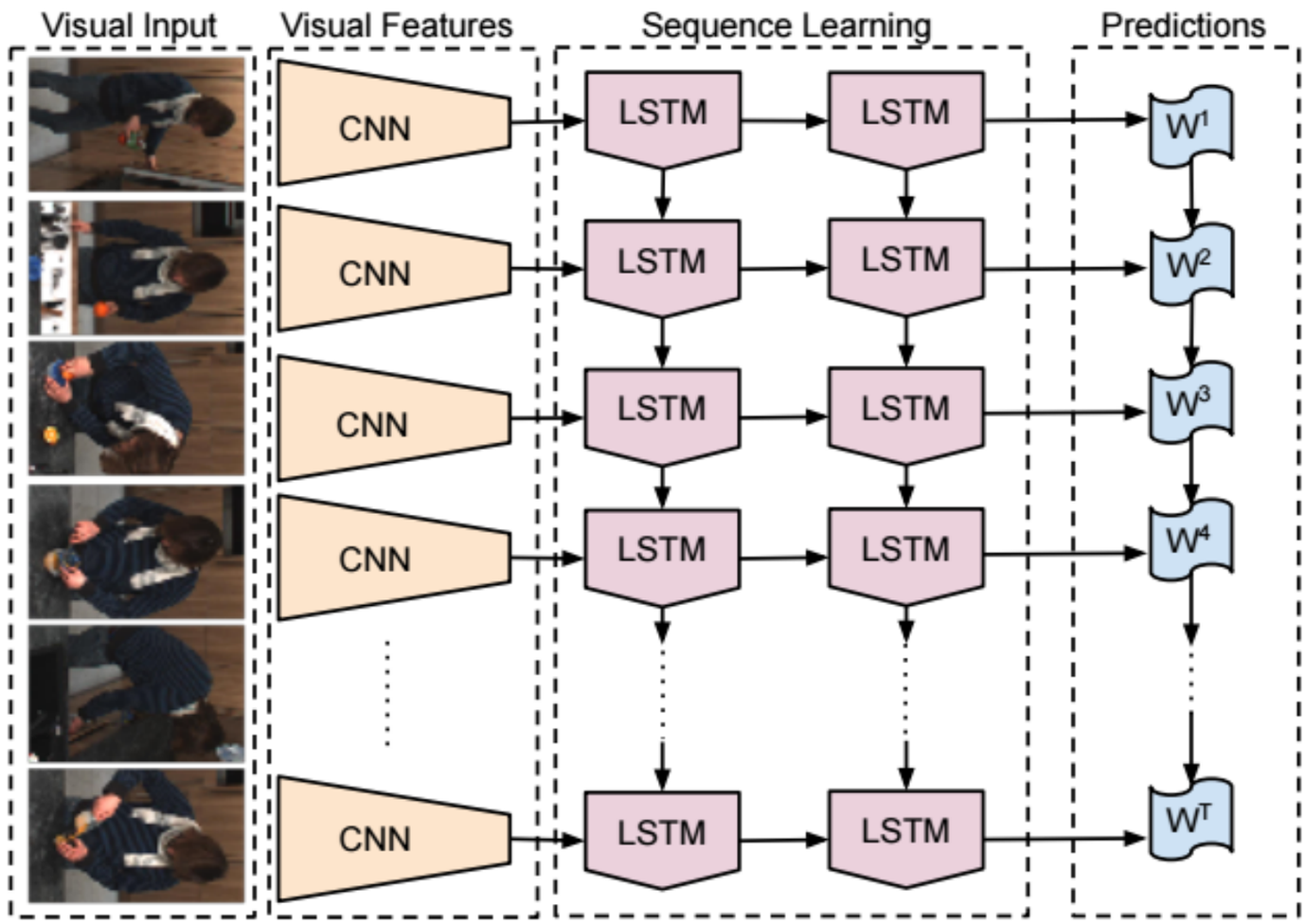
- Using a word embedding with LSTM layers to learn sentiments
- Incorporating a thought unit



[Ghost et al.]

Action Recognition

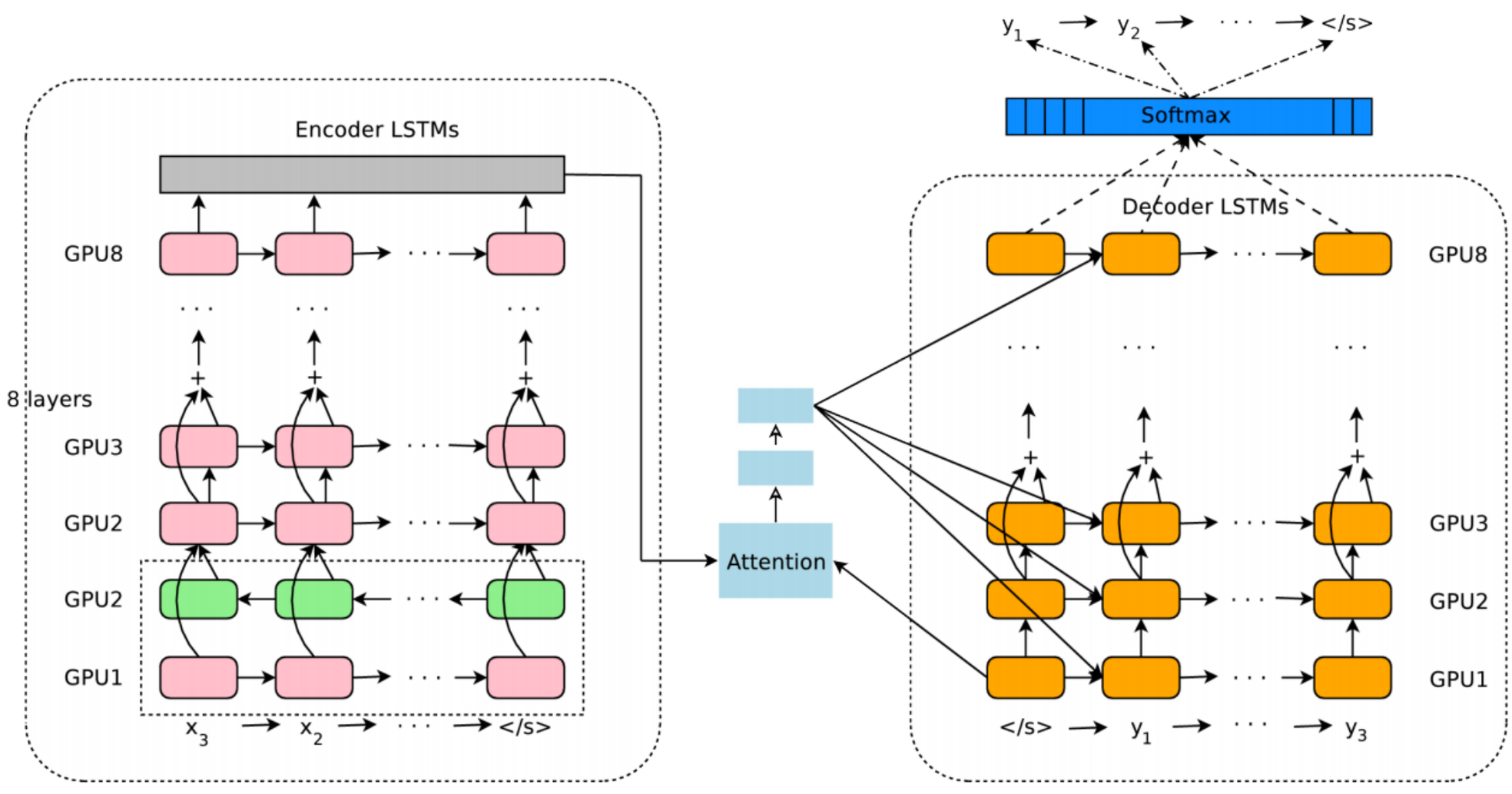
- Using LSTMs and CNN for videos
- CNN creates a feature vector that is feed into LSTM



[Donahue et al.]

Google's Neural Machine Translation System

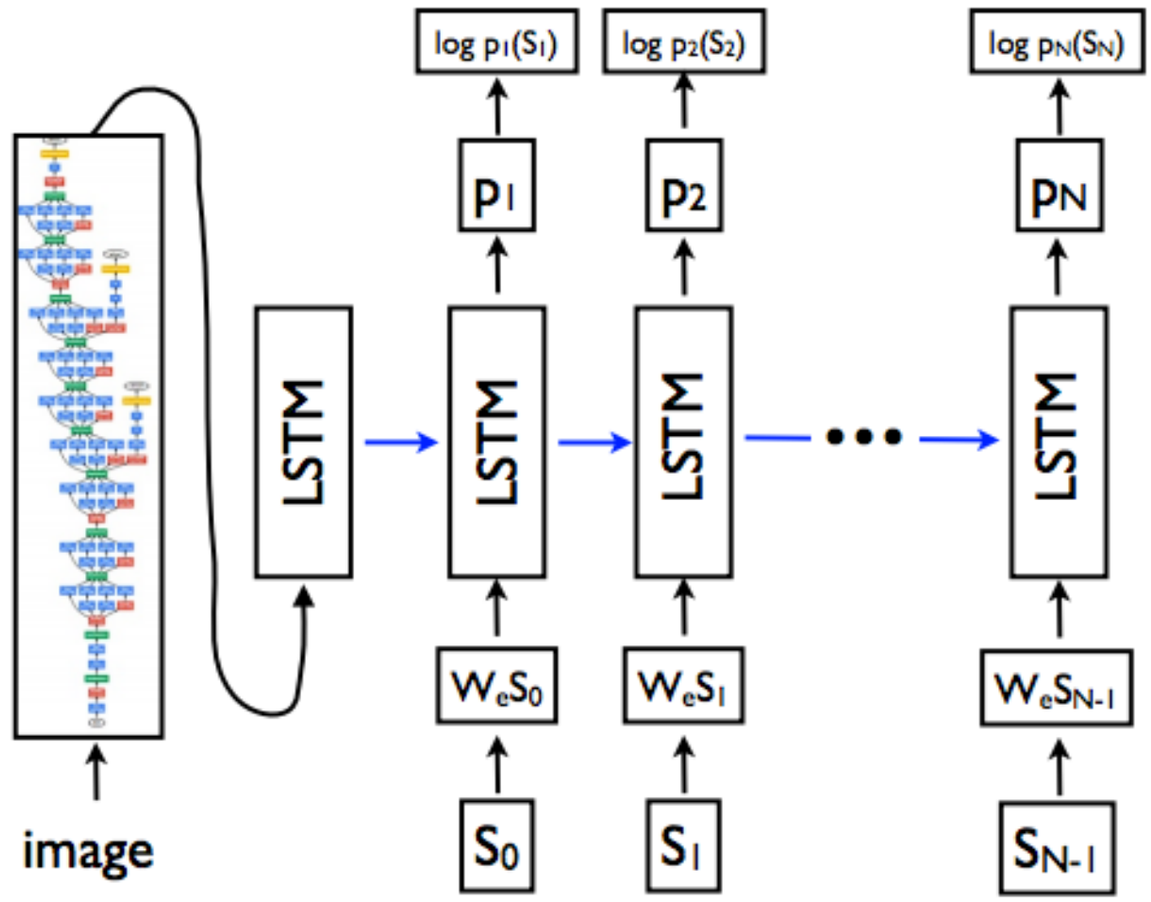
- Encoder and Decoder LSTMs
- Attention model



[Yonghui Wu et al.]

Image Captioning Pt 1

- Combination of CNN and LSTM to caption images
  - Using a pretrained CNN for visual features



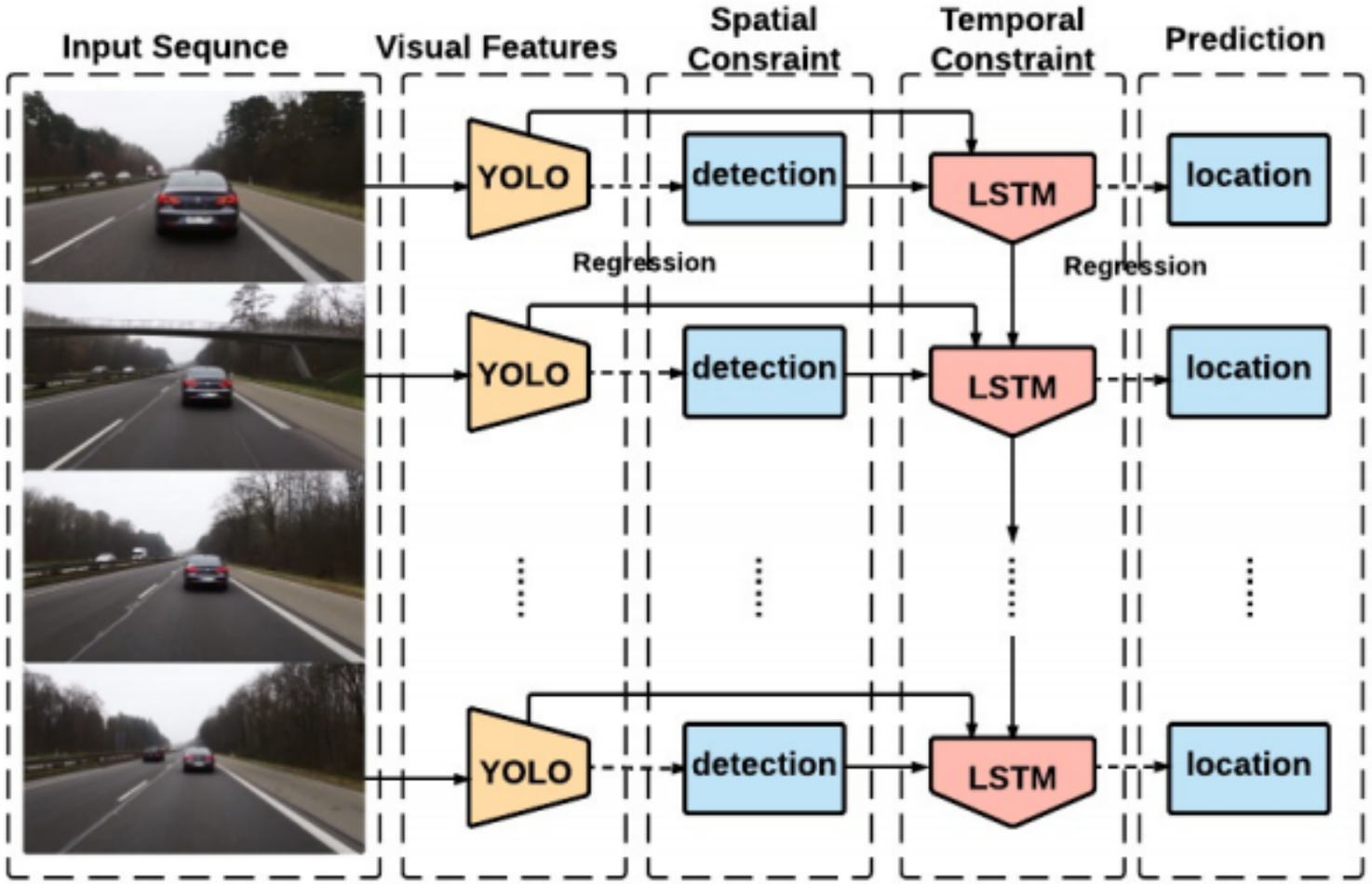
[Vinyals, Toshev, Bengio, Erhan]

Image Captioning Pt 2

[Vinyals, Toshev, Bengio, Erhan]

Object Tracking

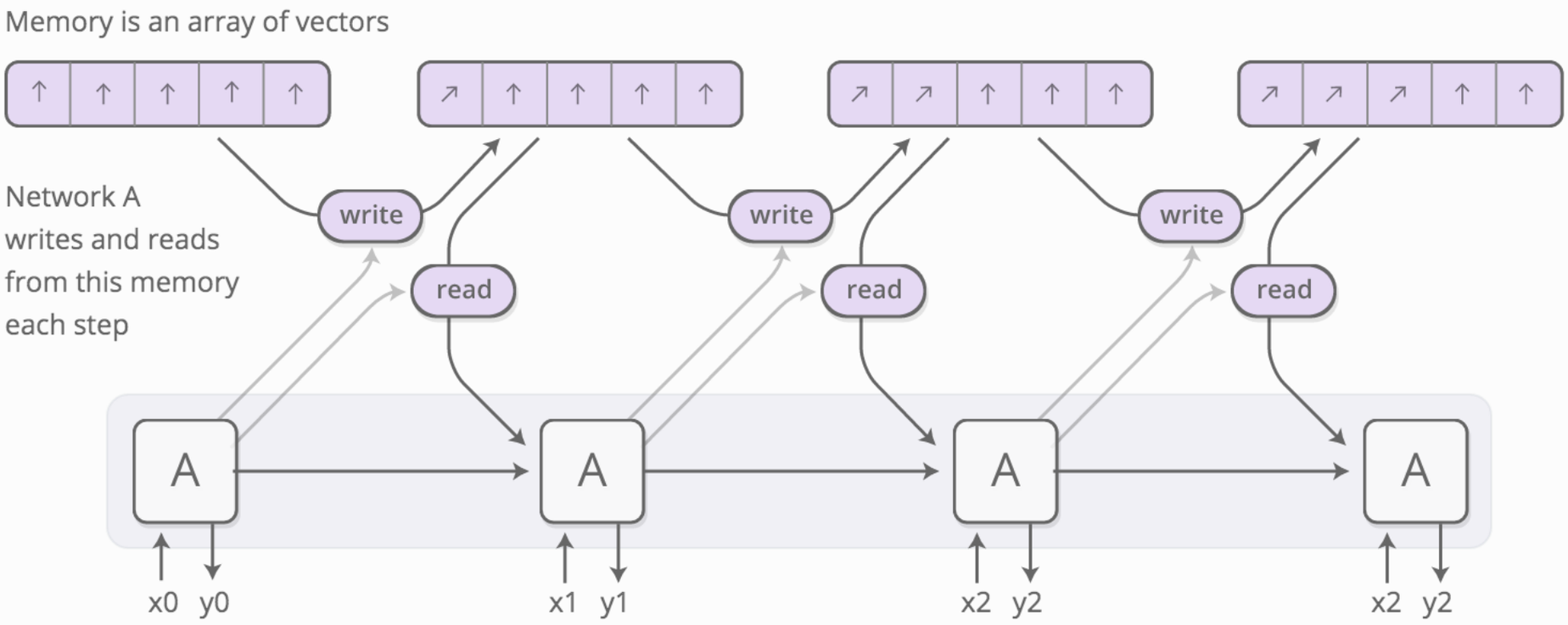
- Using an object detector with an LSTM to track objects
- Model the dynamics of video



[Ning, Zhang, Huang, He, Wang

Neural Turing Machines

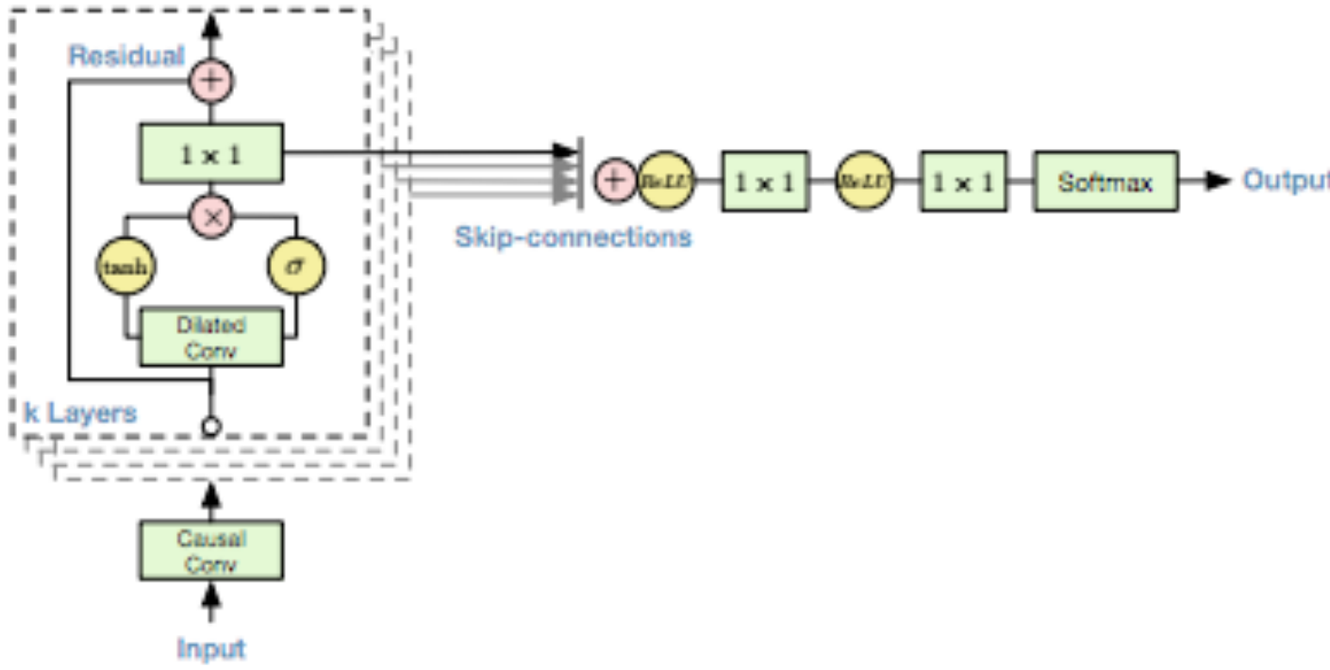
- LSTM with external memory
- Analogous to a Turing Machine



[Chris Olah]

WaveNet

- Using stack of diluted layers
- To generate next sample, it models conditional probability given previous samples



[van den Oord et al.]

DoomBot

- Doom Competition
  - Facebook won 1st place (F1)
  - https://www.youtube.com/watch?v=94EPSjQH38Y