Lattice Recurrent Unit

Chaitanya Ahuja

Language Technologies Institute, CMU

May 30, 2017

Motivation

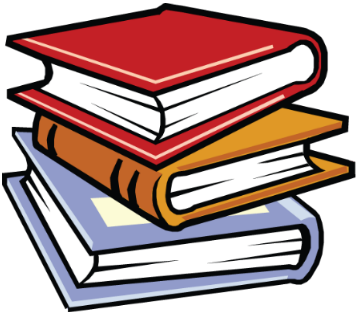
A plethora of data is **sequential** in nature

Motivation

A plethora of data is **sequential** in nature
$\Rightarrow$ The **order** is important

Motivation

A plethora of data is **sequential** in nature
$\Rightarrow$ The **order** is important



Motivation

A plethora of data is **sequential** in nature
$\Rightarrow$ The **order** is important

and because the permutations increase **factorially** ($n!$)
with the number of data points (n)

Motivation

A plethora of data is **sequential** in nature
$\Rightarrow$ The **order** is important

and because the permutations increase **factorially** $(n!)$
with the number of data points (n)

We might lose information if the order is not preserved

Motivation

Language Model

Motivation

Language Model

C H A R

Motivation

Language Model

C H A   R

$$P\left(R \mid C\,H\,A \right)$$

Language Model

C H A R

$$P\left(R \mid C\ H\ A \right)$$

$$P\left(x_t \mid x_{1:t-1} \right)$$

Rise of **Neural Networks**

- **Function Approximator:** A model that can approximate such distributions.

Rise of **Neural Networks**

- **Function Approximator:** A model that can approximate such distributions.

$\Rightarrow$ **Neural Networks**

Rise of **Neural Networks**

- **Function Approximator:** A model that can approximate such distributions.

  $\Rightarrow$ **Neural Networks**

- **Adaptable:** Generalizable over large datasets by increasing number of parameters.

Rise of **Neural Networks**

- **Function Approximator:** A model that can approximate such distributions.

  $\Rightarrow$ **Neural Networks**

- **Adaptable:** Generalizable over large datasets by increasing number of parameters.

  $\Rightarrow$ Increase **depth** of Neural Networks.

Rise of **Neural Networks**

- **Function Approximator:** A model that can approximate such distributions.

  $\Rightarrow$ **Neural Networks**

- **Adaptable:** Generalizable over large datasets by increasing number of parameters.

  $\Rightarrow$ Increase **depth** of Neural Networks.

- **Order Preserving:** Use **order-information** as well to predict.

Rise of **Neural Networks**

- **Function Approximator:** A model that can approximate such distributions.

  $\Rightarrow$ **Neural Networks**

- **Adaptable:** Generalizable over large datasets by increasing number of parameters.

  $\Rightarrow$ Increase **depth** of Neural Networks.

- **Order Preserving:** Use **order-information** as well to predict.

  $\Rightarrow$ Deep Recurrent Neural Networks (**Deep RNNs**)

Rise of **Neural Networks**

- **Function Approximator:** A model that can approximate such distributions.

  $\Rightarrow$ **Neural Networks**

- **Adaptable:** Generalizable over large datasets by increasing number of parameters.

  $\Rightarrow$ Increase **depth** of Neural Networks.

- **Order Preserving:** Use **order-information** as well to predict.

  $\Rightarrow$ Deep Recurrent Neural Networks (**Deep RNNs**)

- **Trainable:** Estimate *accurate* functions in a **practical** time-frame.

Rise of **Neural Networks**

- **Function Approximator:** A model that can approximate such distributions.

  $\Rightarrow$ **Neural Networks**

- **Adaptable:** Generalizable over large datasets by increasing number of parameters.

  $\Rightarrow$ Increase **depth** of Neural Networks.

- **Order Preserving:** Use **order-information** as well to predict.

  $\Rightarrow$ Deep Recurrent Neural Networks (**Deep RNNs**)

- **Trainable:** Estimate *accurate* functions in a **practical** time-frame.

  Use **gradient** based approaches.

Rise of **Neural Networks**

- **Function Approximator:** A model that can approximate such distributions.

  $\Rightarrow$ **Neural Networks**

- **Adaptable:** Generalizable over large datasets by increasing number of parameters.

  $\Rightarrow$ Increase **depth** of Neural Networks.

- **Order Preserving:** Use **order-information** as well to predict.

  $\Rightarrow$ Deep Recurrent Neural Networks (**Deep RNNs**)

- **Trainable:** Estimate *accurate* functions in a **practical** time-frame.

  Use **gradient** based approaches.

  Are they **easy to train**?

Rise of **Recurrent** Neural Networks

$$x_t = \mathsf{RNN}\left(x_{t-1}, c_{t-1}\right)$$

Rise of **Recurrent** Neural Networks

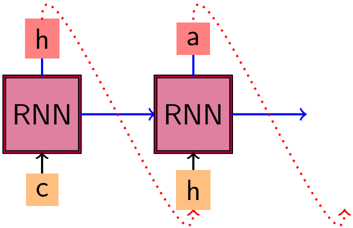
$$
\begin{aligned}
x_t &= \mathsf{RNN}\left(x_{t-1}, c_{t-1}\right) \\
&= \mathsf{RNN}\left(x_{t-1}, x_{t-2} \ldots, x_1\right) \\
&= \mathsf{RNN}\left(x_{1:t-1}\right)
\end{aligned}
$$

Rise of **Recurrent** Neural Networks

$$
\begin{aligned}
x_t &= \mathsf{RNN}\left(x_{t-1}, c_{t-1}\right) \\
&= \mathsf{RNN}\left(x_{t-1}, x_{t-2} \ldots, x_1\right) \\
&= \mathsf{RNN}\left(x_{1:t-1}\right) \\
\Rightarrow \mathsf{RNN} &\sim \mathsf{P}\left(x_t \,\middle|\, x_{1:t-1}\right)
\end{aligned}
$$

Rise of **Recurrent** Neural Networks

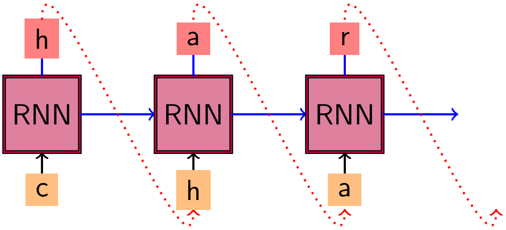
Language Modeling



- Note: Blue Arrows correspond to **hidden states** (c_{t-1}).

Rise of **Recurrent** Neural Networks

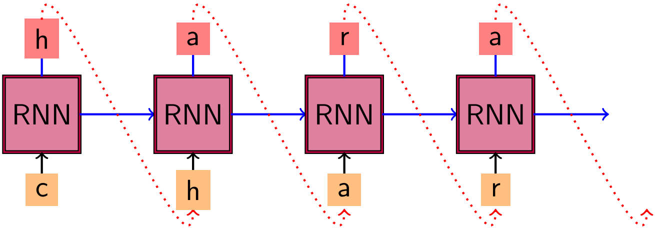
Language Modeling



- Note: Blue Arrows correspond to **hidden states** **(**c_{t-1}**)**.

Rise of **Recurrent** Neural Networks

Language Modeling



- Note: Blue Arrows correspond to **hidden states** **(**c_{t-1}**)**.

Rise of **Recurrent** Neural Networks

- **Trainability:** Vanilla RNNs suffer from the **Vanishing Gradient** problem,

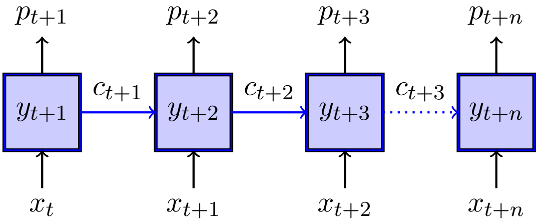


Figure 1: Forward Pass

Rise of **Recurrent** Neural Networks

- **Trainability:** Vanilla RNNs suffer from the **Vanishing Gradient** problem,

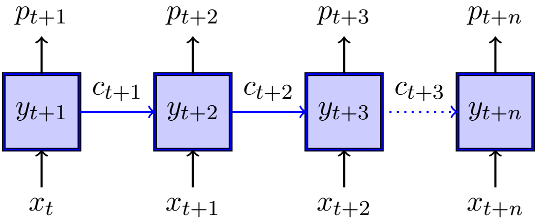


Figure 1: Forward Pass

$$y_{t+1} \quad = \quad \mathbf{X} x_t + \mathbf{C} c_t$$

Rise of **Recurrent** Neural Networks

- **Trainability:** Vanilla RNNs suffer from the **Vanishing Gradient** problem,

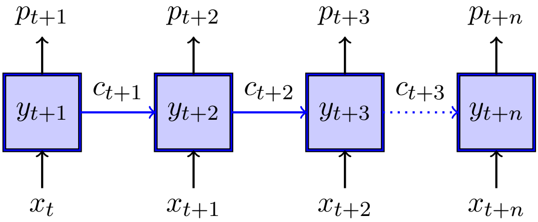


Figure 1: Forward Pass

$$
\begin{aligned}
y_{t+1} &= \mathbf{X}x_t + \mathbf{C}c_t \\
c_{t+1} &= \sigma\left(y_{t+1}\right)
\end{aligned}
$$

Rise of **Recurrent** Neural Networks

- **Trainability:** Vanilla RNNs suffer from the **Vanishing Gradient** problem,

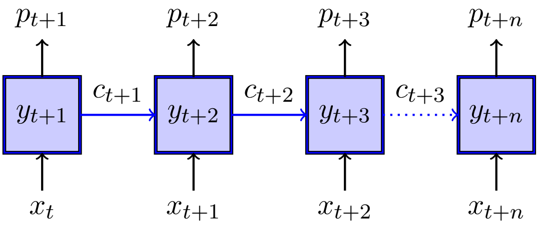


Figure 1: Forward Pass

$$
\begin{aligned}
y_{t+1} &= \mathbf{X} x_t + \mathbf{C} c_t \\
c_{t+1} &= \sigma\left(y_{t+1}\right) \\
p_{t+1} &= \text{softmax}\left(\mathbf{W} c_{t+1}\right)
\end{aligned}
$$

Rise of **Recurrent** Neural Networks

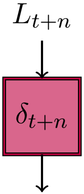


Figure 2: Gradients in Vanilla RNN

Rise of **Recurrent** Neural Networks

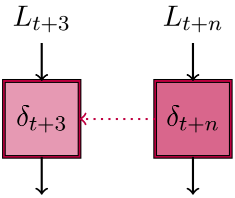


Figure 2: Gradients in Vanilla RNN

Rise of **Recurrent** Neural Networks

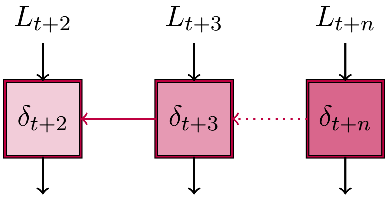


Figure 2: Gradients in Vanilla RNN

Rise of **Recurrent** Neural Networks

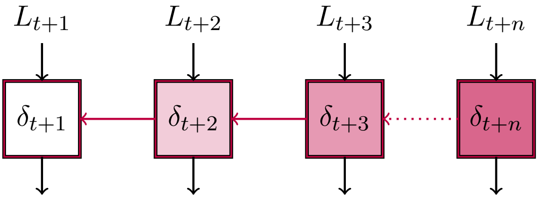


Figure 2: Gradients in Vanilla RNN

Rise of **Recurrent** Neural Networks

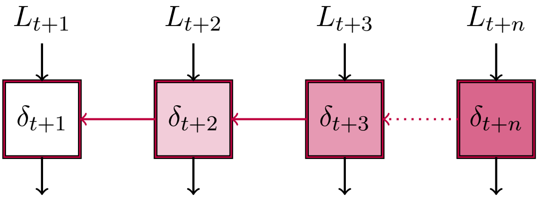


Figure 2: Gradients in Vanilla RNN

$$L_j = (p_j^{gt} - p_j)^2$$

Rise of **Recurrent** Neural Networks

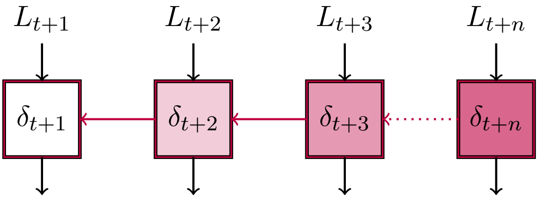


Figure 2: Gradients in Vanilla RNN

$$
\begin{aligned}
L_j &= (p_j^{gt} - p_j)^2 \\
\frac{\partial L_{t+n}}{\partial y_{t+1}} &= \frac{\partial L_{t+n}}{\partial y_{t+n}} \cdot \frac{\partial y_{t+n}}{\partial y_{t+n-1}} \cdot \ldots \frac{\partial y_{t+2}}{\partial y_{t+1}}
\end{aligned}
$$

Rise of **Recurrent** Neural Networks

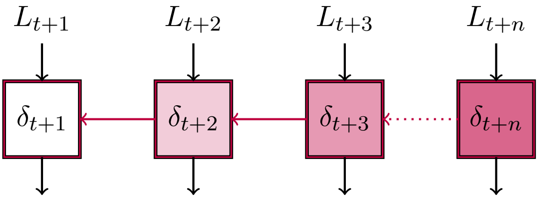


Figure 2: Gradients in Vanilla RNN

$$
\begin{aligned}
L_j &= (p_j^{gt} - p_j)^2 \\
\frac{\partial L_{t+n}}{\partial y_{t+1}} &= \frac{\partial L_{t+n}}{\partial y_{t+n}} \cdot \frac{\partial y_{t+n}}{\partial y_{t+n-1}} \cdot \ldots \frac{\partial y_{t+2}}{\partial y_{t+1}} \\
&= \frac{\partial L_{t+n}}{\partial y_{t+n}} \cdot \prod_{\tau=t+2}^{t+n} \frac{\partial y_\tau}{\partial y_{\tau-1}}
\end{aligned}
$$

Rise of **Recurrent** Neural Networks

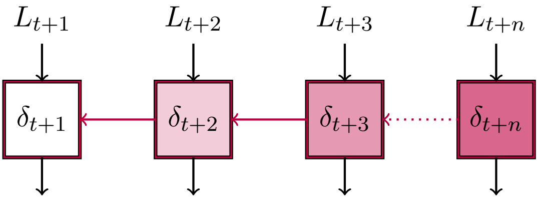


Figure 2: Gradients in Vanilla RNN

$\prod_{\tau=t+2}^{t+n} \frac{\partial y_\tau}{\partial y_{\tau-1}}$

Rise of **Recurrent** Neural Networks

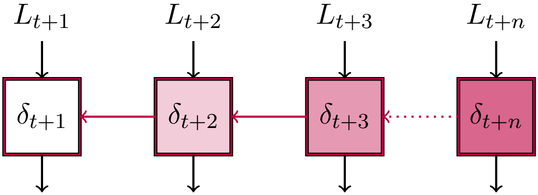


Figure 2: Gradients in Vanilla RNN

$\prod_{\tau=t+2}^{t+n} \frac{\partial y_\tau}{\partial y_{\tau-1}}$

$$\frac{\partial y_\tau}{\partial y_{\tau-1}} = \mathbf{C}^{\mathbf{T}} \sigma' \left(y_{\tau-1} \right)$$

Rise of **Recurrent** Neural Networks

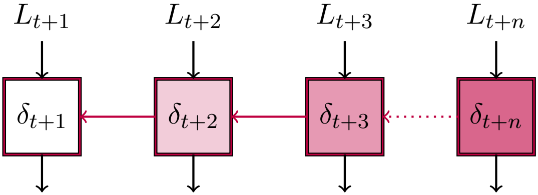


Figure 2: Gradients in Vanilla RNN

$$\prod_{\tau=t+2}^{t+n} \frac{\partial y_\tau}{\partial y_{\tau-1}}$$

$$\begin{aligned} \frac{\partial y_\tau}{\partial y_{\tau-1}} &= \mathbf{C}^{\mathbf{T}} \sigma' \left(y_{\tau-1} \right) \\ \left\| \frac{\partial y_\tau}{\partial y_{\tau-1}} \right\| &\leq \|\mathbf{C}\| \frac{1}{4} \end{aligned}$$

Rise of **Recurrent** Neural Networks

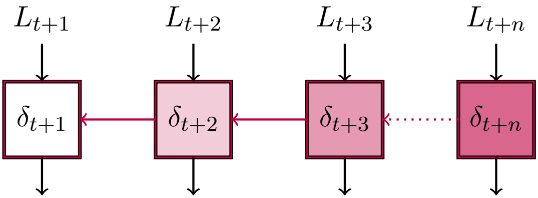


Figure 2: Gradients in Vanilla RNN

$\prod_{\tau=t+2}^{t+n} \frac{\partial y_\tau}{\partial y_{\tau-1}}$

$$\begin{aligned}
\frac{\partial y_\tau}{\partial y_{\tau-1}} &= \mathbf{C}^{\mathbf{T}} \sigma' \left(y_{\tau-1}\right) \\
\left\| \frac{\partial y_\tau}{\partial y_{\tau-1}} \right\| &\leq \|\mathbf{C}\| \frac{1}{4} \\
\prod_{\tau=t+2}^{t+n} \left\| \frac{\partial y_\tau}{\partial y_{\tau-1}} \right\| &\leq \left(\|\mathbf{C}\| \tfrac{1}{4}\right)^{n-1}
\end{aligned}$$

Rise of **Recurrent** Neural Networks

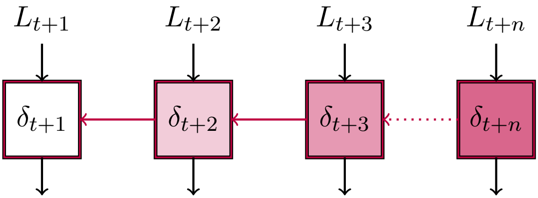


Figure 2: Gradients in Vanilla RNN

$\prod_{\tau=t+2}^{t+n} \frac{\partial y_\tau}{\partial y_{\tau-1}}$

Exponentially Decaying Gradients

$$\frac{\partial y_\tau}{\partial y_{\tau-1}} = \mathbf{C}^{\mathbf{T}} \sigma' \left(y_{\tau-1} \right)$$

$$\left\| \frac{\partial y_\tau}{\partial y_{\tau-1}} \right\| \leq \|\mathbf{C}\| \frac{1}{4}$$

$$\prod_{\tau=t+2}^{t+n} \left\| \frac{\partial y_\tau}{\partial y_{\tau-1}} \right\| \leq \left(\|\mathbf{C}\| \tfrac{1}{4} \right)^{n-1}$$

Rise of **Recurrent** Neural Networks

- GRUs/LSTMs alleviate this problem by using gates on inputs, outputs and hidden states [Hochreiter and Schmidhuber, 1997].

$$y_\tau = \mathbf{X} x_\tau + \mathbf{C} \sigma \left(y_{\tau - 1} \right) \tag{1}$$

Let $\mathbf{C} = \mathbf{I}$ and $\sigma(x) = x$. Then,

Rise of **Recurrent** Neural Networks

- GRUs/LSTMs alleviate this problem by using gates on inputs, outputs and hidden states [Hochreiter and Schmidhuber, 1997].

$$y_\tau = \mathbf{X}x_\tau + \mathbf{C}\sigma\left(y_{\tau-1}\right) \tag{1}$$

Let $\mathbf{C} = \mathbf{I}$ and $\sigma(x) = x$. Then,

$$\frac{\partial y_\tau}{\partial y_{\tau-1}} = \mathbf{C}^T\sigma'(y_{\tau-1})$$

Rise of **Recurrent** Neural Networks

- GRUs/LSTMs alleviate this problem by using gates on inputs, outputs and hidden states [Hochreiter and Schmidhuber, 1997].

$$y_\tau = \mathbf{X}x_\tau + \mathbf{C}\sigma\left(y_{\tau-1}\right) \tag{1}$$

Let $\mathbf{C} = \mathbf{I}$ and $\sigma(x) = x$. Then,

$$\begin{aligned} \frac{\partial y_\tau}{\partial y_{\tau-1}} &= \mathbf{C}^T\sigma'(y_{\tau-1}) \\ &\approx \mathbf{I} \end{aligned}$$

To retain modeling power, introduce a non-linear term

Rise of **Recurrent** Neural Networks

- GRUs/LSTMs alleviate this problem by using gates on inputs, outputs and hidden states [Hochreiter and Schmidhuber, 1997].

$$y_\tau = \mathbf{X}x_\tau + \mathbf{C}\sigma\left(y_{\tau-1}\right) \tag{1}$$

Let $\mathbf{C} = \mathbf{I}$ and $\sigma(x) = x$. Then,

$$\begin{aligned} \frac{\partial y_\tau}{\partial y_{\tau-1}} &= \mathbf{C}^T\sigma'(y_{\tau-1}) \\ &\approx \mathbf{I} \qquad \text{(Constant Error Carrousel)} \end{aligned}$$

To retain modeling power, introduce a non-linear term

Rise of **Recurrent** Neural Networks

- GRUs/LSTMs alleviate this problem by using gates on inputs, outputs and hidden states [Hochreiter and Schmidhuber, 1997].

$$y_\tau = \mathbf{X} x_\tau + \mathbf{C} \sigma \left(y_{\tau-1} \right) \tag{1}$$

Let $\mathbf{C} = \mathbf{I}$ and $\sigma(x) = x$. Then,

$$\begin{aligned} \frac{\partial y_\tau}{\partial y_{\tau-1}} &= \mathbf{C}^T \sigma'(y_{\tau-1}) \\ &\approx \mathbf{I} \qquad \text{(Constant Error Carrousel)} \end{aligned}$$

To retain modeling power, introduce a non-linear term

$$y_\tau = \mathbf{X} x_\tau + y_{\tau-1} + \mathbf{z} \cdot \phi \left(y_{\tau-1} \right)$$

Rise of **Recurrent** Neural Networks

- GRUs/LSTMs alleviate this problem by using gates on inputs, outputs and hidden states [Hochreiter and Schmidhuber, 1997].

$$y_\tau = \mathbf{X} x_\tau + \mathbf{C} \sigma \left(y_{\tau-1} \right) \tag{1}$$

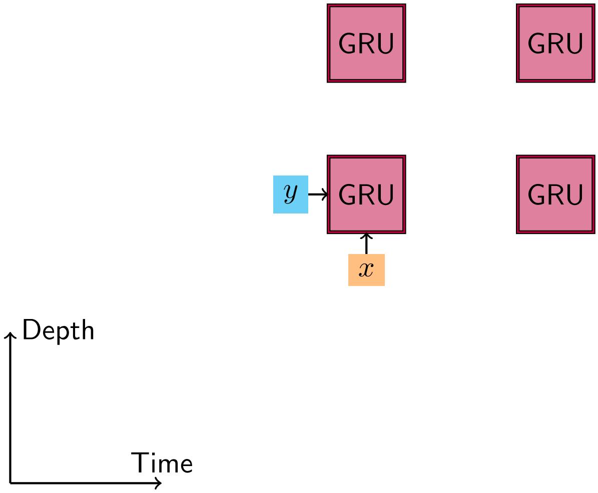
Let $\mathbf{C} = \mathbf{I}$ and $\sigma(x) = x$. Then,

$$\begin{aligned} \frac{\partial y_\tau}{\partial y_{\tau-1}} &= \mathbf{C}^T \sigma'(y_{\tau-1}) \\ &\approx \mathbf{I} \qquad \text{(Constant Error Carrousel)} \end{aligned}$$

To retain modeling power, introduce a non-linear term

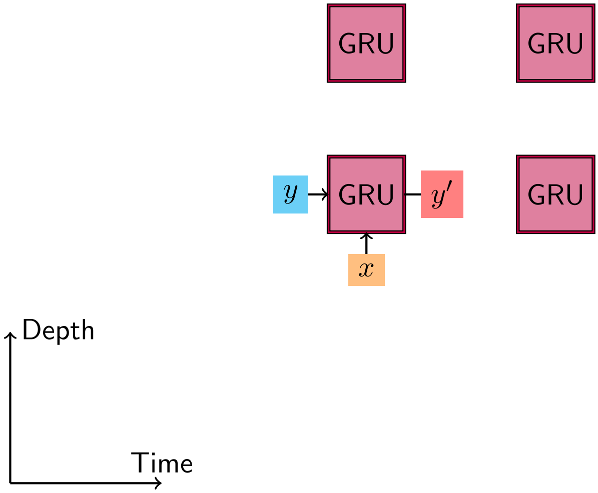
$$\begin{aligned} y_\tau &= \mathbf{X} x_\tau + y_{\tau-1} + \mathbf{z} \cdot \phi \left(y_{\tau-1} \right) \\ \frac{\partial y_\tau}{\partial y_{\tau-1}} &\approx \mathbf{I} + \mathbf{z} \cdot \phi' \left(y_{\tau-1} \right) \end{aligned}$$

GRU Equations



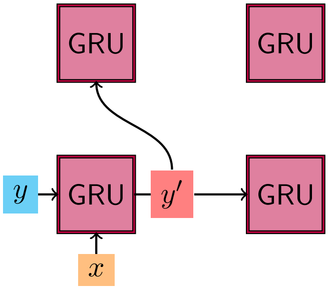
Rise of **Recurrent** Neural Networks

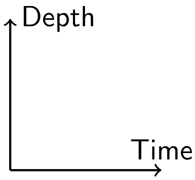
GRU Equations



Rise of **Recurrent** Neural Networks

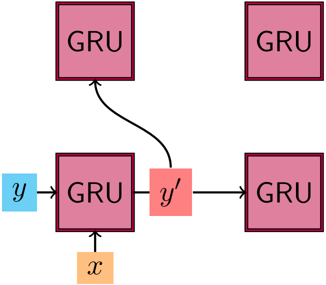
GRU Equations



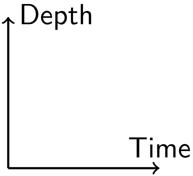


Rise of **Recurrent** Neural Networks

GRU Equations

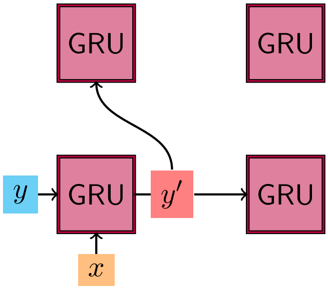


$$z = \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y)$$

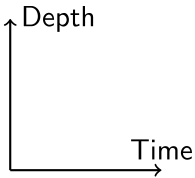


Rise of **Recurrent** Neural Networks

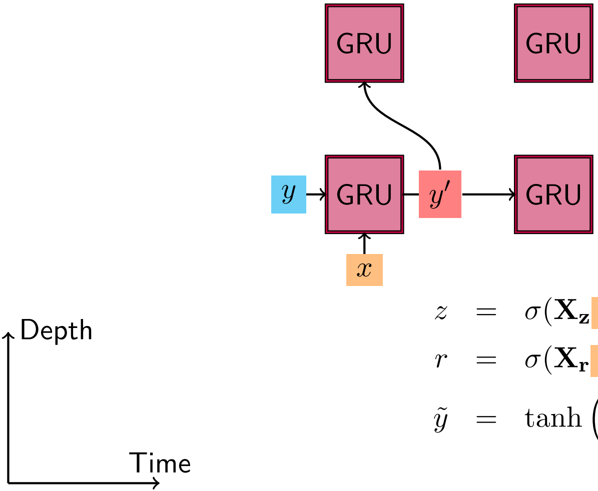
GRU Equations



$$z = \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y)$$
$$r = \sigma(\mathbf{X_r}\, x + \mathbf{Y_r}\, y)$$

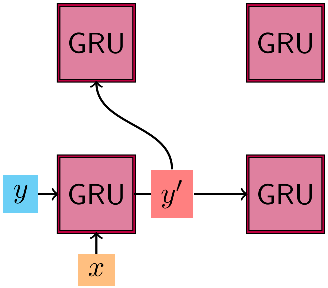


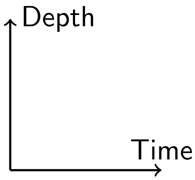
GRU Equations



$$
\begin{aligned}
z &= \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y) \\
r &= \sigma(\mathbf{X_r}\, x + \mathbf{Y_r}\, y) \\
\tilde{y} &= \tanh\left(\mathbf{X}\, x + \mathbf{Y}(r \cdot y)\right)
\end{aligned}
$$

GRU Equations





$$z = \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y)$$

$$r = \sigma(\mathbf{X_r}\, x + \mathbf{Y_r}\, y)$$

$$\tilde{y} = \tanh\left(\mathbf{X}\, x + \mathbf{Y}(r \cdot y)\right)$$

$$y_{t+1} = y' = (1 - z) \cdot y + z \cdot \tilde{y}$$

Rise of **Deep** Recurrent Neural Networks

- Even though the gating mechanisms (i.e. CEC) resolve vanishing gradient issues along time,

Rise of **Deep** Recurrent Neural Networks

- Even though the gating mechanisms (i.e. CEC) resolve vanishing gradient issues along time,
- Gradients have been shown to vanish along depth.

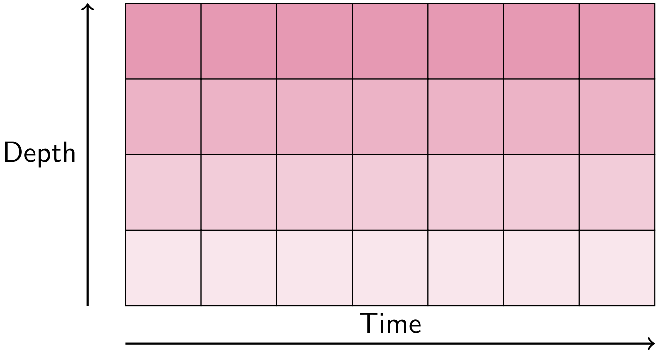


Figure 3: Gradients along Depth in a GRU

Rise of **Deep** Recurrent Neural Networks

- We observe the gradients while training a Deep GRU network on Character-Level Language Model.

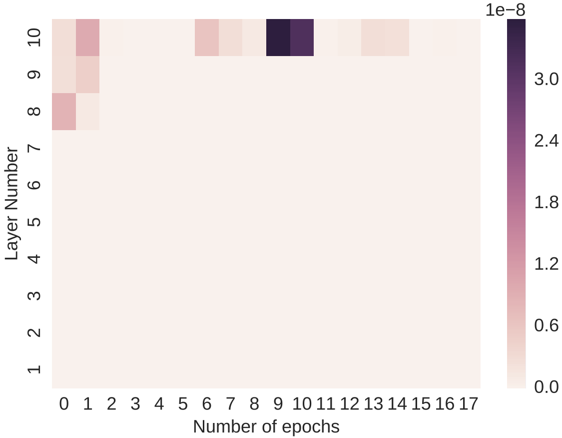
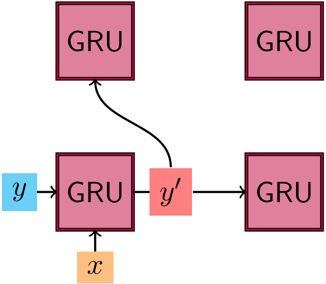


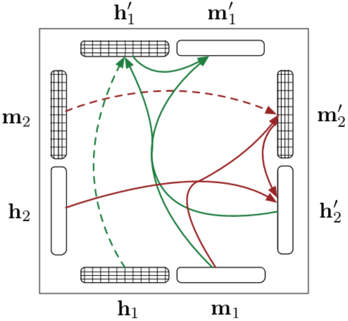
Figure 4: Gradient-Norms across depth in a 10-layered GRU

Rise of **Deep** Recurrent Neural Networks



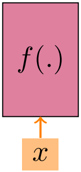
Same hidden state y' passed along time and depth

- Grid-LSTMs [Kalchbrenner et al., 2015] is to have **different hidden states along time and depth**.

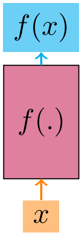


Related Work

- Recurrent Highway Networks [Zilly et al., 2016] use gating in the depth to enforce **Constant Error Carrousel**.

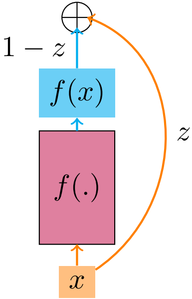


- Recurrent Highway Networks [Zilly et al., 2016] use gating in the depth to enforce **Constant Error Carrousel**.

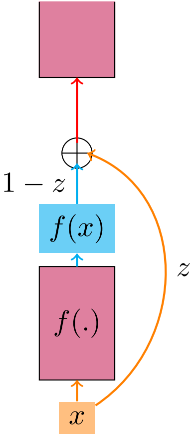


Related Work

- Recurrent Highway Networks [Zilly et al., 2016] use gating in the depth to enforce **Constant Error Carrousel**.



- Recurrent Highway Networks [Zilly et al., 2016] use gating in the depth to enforce **Constant Error Carrousel**.



Research Question

Is it possible to design a **Recurrent Unit**

Research Question

Is it possible to design a **Recurrent Unit**

that passes
different hidden states along depth and time

Research Question

Is it possible to design a **Recurrent Unit**

that passes
different hidden states along depth and time

while
enforcing **Constant Error Carrousel** ?

Research Question

Is it possible to design a **Recurrent Unit**

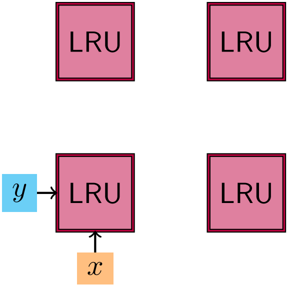
that passes
different hidden states along depth and time

while
enforcing **Constant Error Carrousel** ?

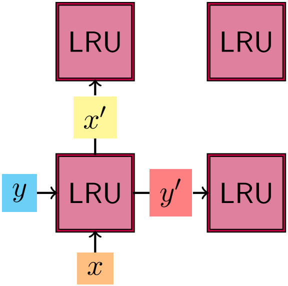
YES!!!

Formulation of Lattice Recurrent Unit

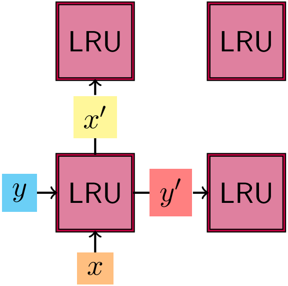
Formulation of Lattice Recurrent Unit



Formulation of Lattice Recurrent Unit

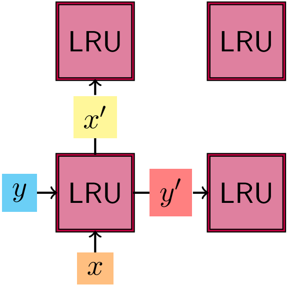


Formulation of Lattice Recurrent Unit



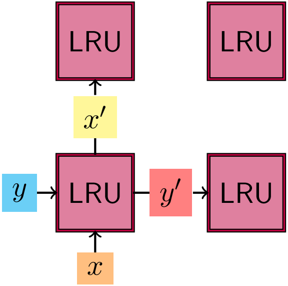
$$z = \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y)$$

Formulation of Lattice Recurrent Unit



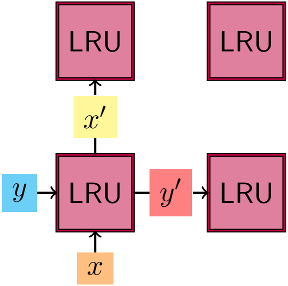
$$z = \sigma(\mathbf{X_z} x + \mathbf{Y_z} y)$$
$$r = \sigma(\mathbf{X_r} x + \mathbf{Y_r} y)$$

Formulation of Lattice Recurrent Unit



$$z = \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y)$$
$$r = \sigma(\mathbf{X_r}\, x + \mathbf{Y_r}\, y)$$
$$q = \sigma(\mathbf{X_q}\, x + \mathbf{Y_q}\, y)$$

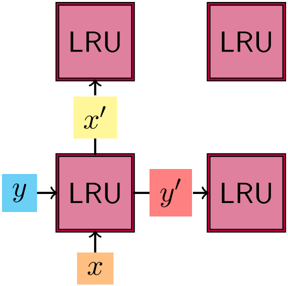
Formulation of Lattice Recurrent Unit



$$\tilde{x} = \tanh\left(\mathbf{X_x}\, x + \mathbf{Y_x}\left(r \cdot y\right)\right)$$

$$z = \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y)$$
$$r = \sigma(\mathbf{X_r}\, x + \mathbf{Y_r}\, y)$$
$$q = \sigma(\mathbf{X_q}\, x + \mathbf{Y_q}\, y)$$

Formulation of Lattice Recurrent Unit



$$z = \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y)$$

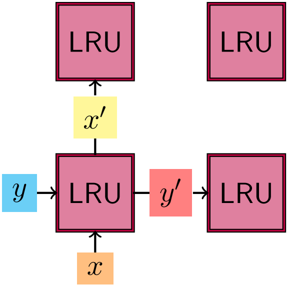
$$r = \sigma(\mathbf{X_r}\, x + \mathbf{Y_r}\, y)$$

$$q = \sigma(\mathbf{X_q}\, x + \mathbf{Y_q}\, y)$$

$$\tilde{x} = \tanh\left(\mathbf{X_x}\, x + \mathbf{Y_x}\left(r \cdot y\right)\right)$$

$$\tilde{y} = \tanh\left(\mathbf{Y_y}\, y + \mathbf{X_y}\left(q \cdot x\right)\right)$$

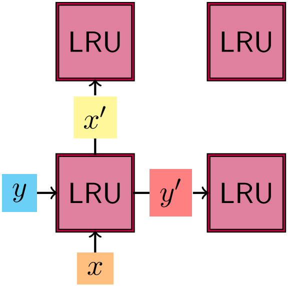
Formulation of Lattice Recurrent Unit



$$z = \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y)$$
$$r = \sigma(\mathbf{X_r}\, x + \mathbf{Y_r}\, y)$$
$$q = \sigma(\mathbf{X_q}\, x + \mathbf{Y_q}\, y)$$

$$\tilde{x} = \tanh\left(\mathbf{X_x}\, x + \mathbf{Y_x}\left(r \cdot y\right)\right)$$
$$\tilde{y} = \tanh\left(\mathbf{Y_y}\, y + \mathbf{X_y}\left(q \cdot x\right)\right)$$
$$x_{t+1} = x' = z \cdot \tilde{y} + (1 - z) \cdot x$$

Formulation of Lattice Recurrent Unit



$$z = \sigma(\mathbf{X_z}\, x + \mathbf{Y_z}\, y)$$
$$r = \sigma(\mathbf{X_r}\, x + \mathbf{Y_r}\, y)$$
$$q = \sigma(\mathbf{X_q}\, x + \mathbf{Y_q}\, y)$$

$$\begin{aligned}
\tilde{x} &= \tanh\left(\mathbf{X_x}\, x + \mathbf{Y_x}\left(r \cdot y\right)\right) \\
\tilde{y} &= \tanh\left(\mathbf{Y_y}\, y + \mathbf{X_y}\left(q \cdot x\right)\right) \\
x_{t+1} = x' &= z \cdot \tilde{y} + (1 - z) \cdot x \\
y_{t+1} = y' &= z \cdot \tilde{x} + (1 - z) \cdot y
\end{aligned}$$

Task: **Character Level** Language Modeling

- Out of Vocabulary Words can potentially be modeled.

Task: **Character Level** Language Modeling

- Out of Vocabulary Words can potentially be modeled.
- Character Aware Neural Language Models [Kim et al., 2016]

Task: **Character Level** Language Modeling

- Out of Vocabulary Words can potentially be modeled.
- Character Aware Neural Language Models [Kim et al., 2016]
- Speech Synthesis [Wang et al., 2017]

Datasets

- We take 2 datasets **Penn Tree Bank** and **War and Peace**.

Datasets

- We take 2 datasets **Penn Tree Bank** and **War and Peace**.
- Each of them has around **5 million** characters.

War and Peace (WP)

Well, Prince, so Genoa and Lucca are now just family estates of the Bonapartes. But I warn you, if you don't tell me that this means war, if you still try to defend the infamies and horrors perpetrated by that Antichrist–I really believe he is Antichrist–I will have nothing more to do with you and you are no longer my friend, no longer my 'faithful slave,' as you call yourself!

Penn Tree Bank (PTB)

the asbestos fiber <unk> is unusually <unk> once it enters the <unk> with even brief exposures to it causing symptoms that show up decades later researchers said, <unk> inc. the unit of new york-based <unk> corp. that makes kent cigarettes stopped using <unk> in its <unk> cigarette filters in N.

Experimental Setup

- Multi-Class Classification Problem

Experimental Setup

- Multi-Class Classification Problem
- Categorical Cross-Entropy Loss
  - loss $= \sum_{\forall i \in \mathcal{C}} p_i \log(\hat{p}_i)$
    where $\mathcal{C}$ is the set of all classes.

Experimental Setup

- Multi-Class Classification Problem
- Categorical Cross-Entropy Loss
  - loss $= \sum_{\forall i \in \mathcal{C}} p_i \log(\hat{p}_i)$
    where $\mathcal{C}$ is the set of all classes.

- RNN unrolled 50 time-steps in time.

Experimental Questions

- **Accuracy:** For equal number of parameters does the **loss improve**?

Experimental Questions

- **Accuracy:** For equal number of parameters does the **loss improve**?
- **Convergence Rate:** How many epochs does it take for the model to converge?

Experimental Questions

- **Accuracy:** For equal number of parameters does the **loss improve**?
- **Convergence Rate:** How many epochs does it take for the model to converge?
- **Trainability:** How are the **gradient norms** distributed across layers?

Results - Accuracy

Table 1: Penn Treebank Dataset and losses are in bits per character (BPC). Lower is better. l is the number of layers.

	Tess Loss (BPC)				**Hidden**
Model	$l=2$	$l=4$	$l=6$	$l=8$	**Units**
GRU	1.107	**1.092**	1.125	2.99	387

Results - Accuracy

Table 1: Penn Treebank Dataset and losses are in bits per character (BPC). Lower is better. l is the number of layers.

	Tess Loss (BPC)				**Hidden**
Model	$l = 2$	$l = 4$	$l = 6$	$l = 8$	**Units**
GRU	1.107	**1.092**	1.125	2.99	387
LSTM	1.110	1.13	1.254	1.311	335

Results - Accuracy

Table 1: Penn Treebank Dataset and losses are in bits per character (BPC). Lower is better. l is the number of layers.

	Tess Loss (BPC)				**Hidden**
Model	$l = 2$	$l = 4$	$l = 6$	$l = 8$	**Units**
GRU	1.107	**1.092**	1.125	2.99	387
LSTM	1.110	1.13	1.254	1.311	335
GLSTM	1.119	1.106	1.111	1.105	237
Highway	1.110	1.102	1.107	-	-

Results - Accuracy

Table 1: Penn Treebank Dataset and losses are in bits per character (BPC). Lower is better. l is the number of layers.

	Tess Loss (BPC)				**Hidden**
Model	$l = 2$	$l = 4$	$l = 6$	$l = 8$	**Units**
GRU	1.107	**1.092**	1.125	2.99	387
LSTM	1.110	1.13	1.254	1.311	335
GLSTM	1.119	1.106	1.111	1.105	237
Highway	1.110	1.102	1.107	-	-
LRU	**1.097**	1.102	**1.101**	**1.103**	300

Results - Accuracy

Table 2: War and Peace Dataset and losses are in bits per character (BPC). Lower is better. l is the number of layers.

	Tess Loss (BPC)				**Hidden**
Model	$l = 2$	$l = 4$	$l = 6$	$l = 8$	**Units**
GRU	1.285	1.293	1.344	3.104	387

Results - Accuracy

Table 2: War and Peace Dataset and losses are in bits per character (BPC). Lower is better. l is the number of layers.

	Tess Loss (BPC)				**Hidden**
Model	$l = 2$	$l = 4$	$l = 6$	$l = 8$	**Units**
GRU	1.285	1.293	1.344	3.104	387
LSTM	1.297	1.353	1.530	3.100	335

Results - Accuracy

Table 2: War and Peace Dataset and losses are in bits per character (BPC). Lower is better. l is the number of layers.

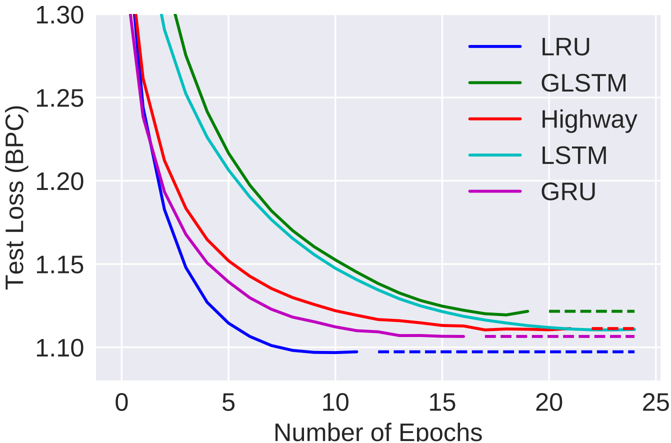
	Tess Loss (BPC)				**Hidden**
Model	$l = 2$	$l = 4$	$l = 6$	$l = 8$	**Units**
GRU	1.285	1.293	1.344	3.104	387
LSTM	1.297	1.353	1.530	3.100	335
GLSTM	1.319	1.317	1.312	1.31	237
Highway	1.294	1.298	1.306	1.305	-

Results - Accuracy

Table 2: War and Peace Dataset and losses are in bits per character (BPC). Lower is better. l is the number of layers.

	Tess Loss (BPC)				**Hidden**
Model	$l = 2$	$l = 4$	$l = 6$	$l = 8$	**Units**
GRU	1.285	1.293	1.344	3.104	387
LSTM	1.297	1.353	1.530	3.100	335
GLSTM	1.319	1.317	1.312	1.31	237
Highway	1.294	1.298	1.306	1.305	-
LRU	**1.279**	**1.280**	**1.281**	**1.287**	300

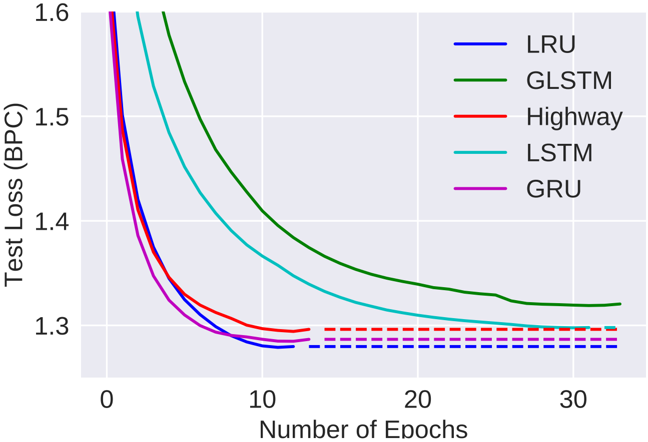
Results - Convergence Rates



(a) PTB dataset

Figure 5: Convergence rates of various models on PTB and WP datasets.

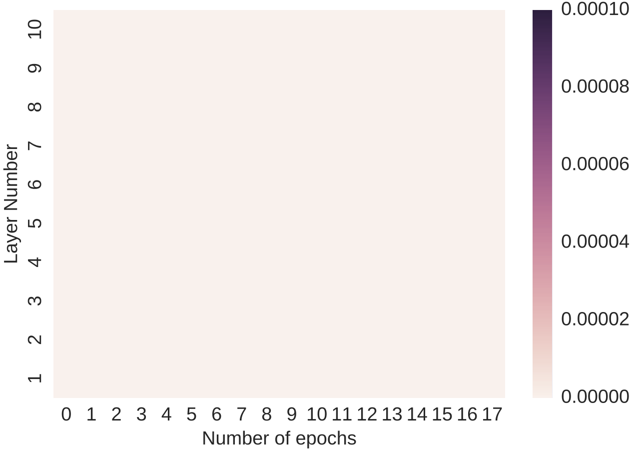
Results - Convergence Rates



(a) WP dataset

Figure 5: Convergence rates of various models on PTB and WP datasets.

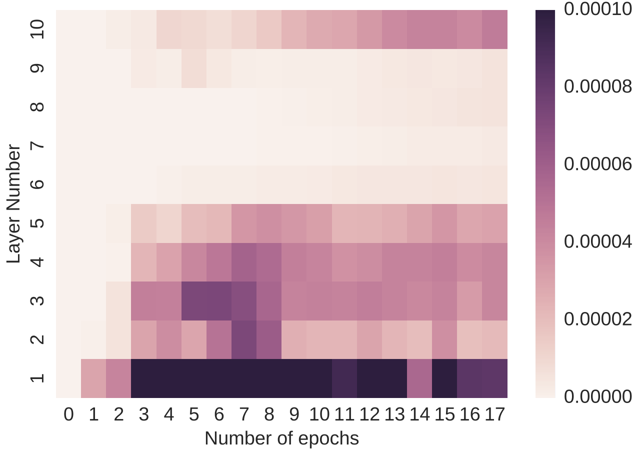
Results - Trainability



(a) GRU

Figure 6: Gradient Norms across layers and number of epochs as a heatmap. Darker values are bigger.

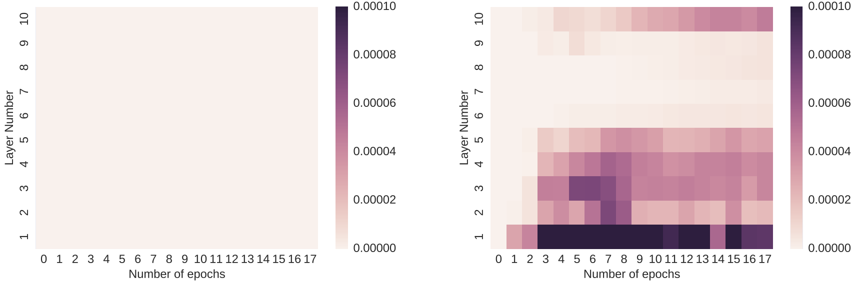
Results - Trainability



(a) LRU

Figure 6: Gradient Norms across layers and number of epochs as a heatmap. Darker values are bigger.

Results - Trainability



(a) GRU

(b) LRU

Figure 6: Gradient Norms across layers and number of epochs as a heatmap. Darker values are bigger.

Conclusions

- **Faster** convergence rates for **LRU**

Conclusions

- **Faster** convergence rates for **LRU**
- Gradients are spread out across the layers. Hence **alleviation** of vanishing gradients.

Conclusions

- **Faster** convergence rates for **LRU**
- Gradients are spread out across the layers. Hence **alleviation** of vanishing gradients.
- While **not aggravating** the performance.

References I

Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural computation, 9(8):1735–1780.

Kalchbrenner, N., Danihelka, I., and Graves, A. (2015).
Grid long short-term memory.
arXiv preprint arXiv:1507.01526.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016).
Character-aware neural language models.
In *Thirtieth AAAI Conference on Artificial Intelligence*.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., et al. (2017).
Tacotron: Towards end-to-end speech syn.
arXiv preprint arXiv:1703.10135.

Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. (2016).
Recurrent highway networks.
arXiv preprint arXiv:1607.03474.

References II

Hochreiter, S. and Schmidhuber, J. (1997).
Long short-term memory.
Neural computation, 9(8):1735–1780.

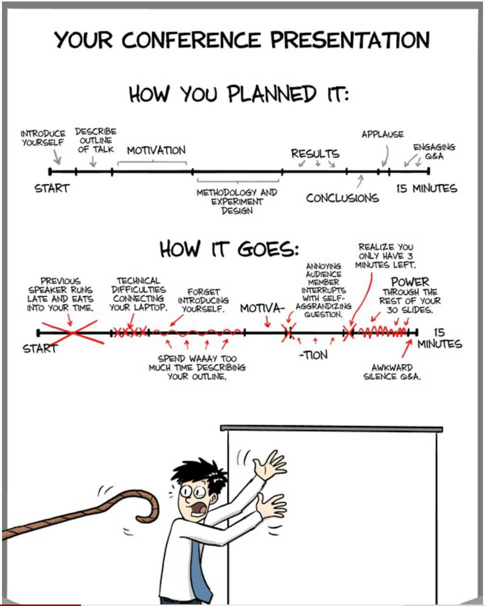
Kalchbrenner, N., Danihelka, I., and Graves, A. (2015).
Grid long short-term memory.
arXiv preprint arXiv:1507.01526.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. (2016).
Character-aware neural language models.
In *Thirtieth AAAI Conference on Artificial Intelligence*.

Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., et al. (2017).
Tacotron: Towards end-to-end speech syn.
arXiv preprint arXiv:1703.10135.

Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. (2016).
Recurrent highway networks.
arXiv preprint arXiv:1607.03474.

Thank You


YOUR CONFERENCE PRESENTATION
HOW YOU PLANNED IT:
INTRODUCE YOURSELF
DESCRIBE OUTLINE OF TALK
MOTIVATION
RESULTS
APPLAUSE
ENGAGING Q&A
START
METHODOLOGY AND EXPERIMENT DESIGN
CONCLUSIONS
15 MINUTES
HOW IT GOES:
PREVIOUS SPEAKER RUNS LATE AND EATS INTO YOUR TIME.
TECHNICAL DIFFICULTIES CONNECTING YOUR LAPTOP.
FORGET INTRODUCING YOURSELF.
MOTIVA-
ANNOYING AUDIENCE MEMBER INTERRUPTS WITH SELF-AGGRANDIZING QUESTION.
REALIZE YOU ONLY HAVE 3 MINUTES LEFT.
POWER THROUGH THE REST OF YOUR 30 SLIDES.
START
15 MINUTES
SPEND WAAAY TOO MUCH TIME DESCRIBING YOUR OUTLINE.
-TION
AWKWARD SILENCE Q&A.