Stabilizing Gradients for Deep Neural Networks

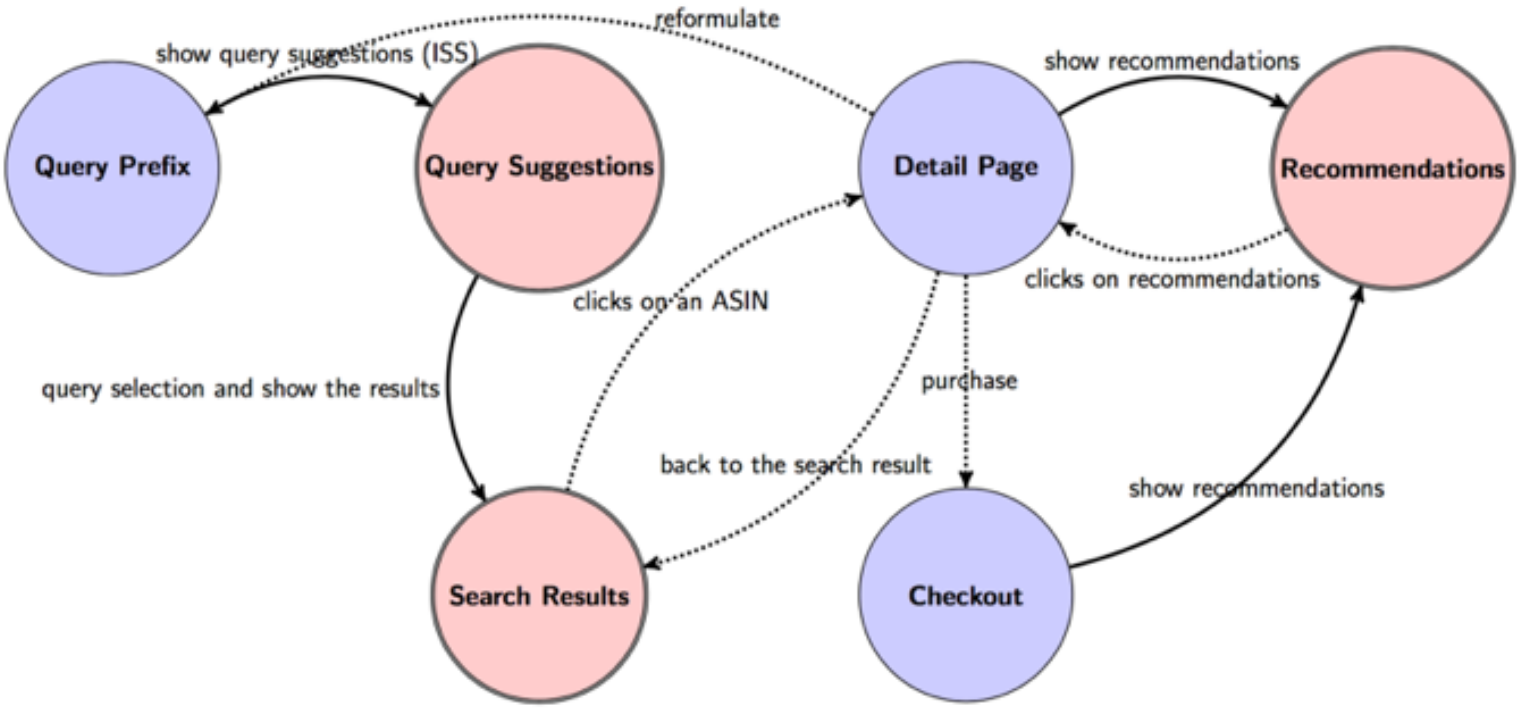
Inderjit S. Dhillon
Amazon & UT Austin
LA/Opt Seminar, Stanford University
May 10, 2018

Joint work with Jiong Zhang and Qi Lei (UT Austin),
Vijai Mohan (Amazon Search), Po-Wei Wang (CMU) & Huan Zhang (UC Davis)

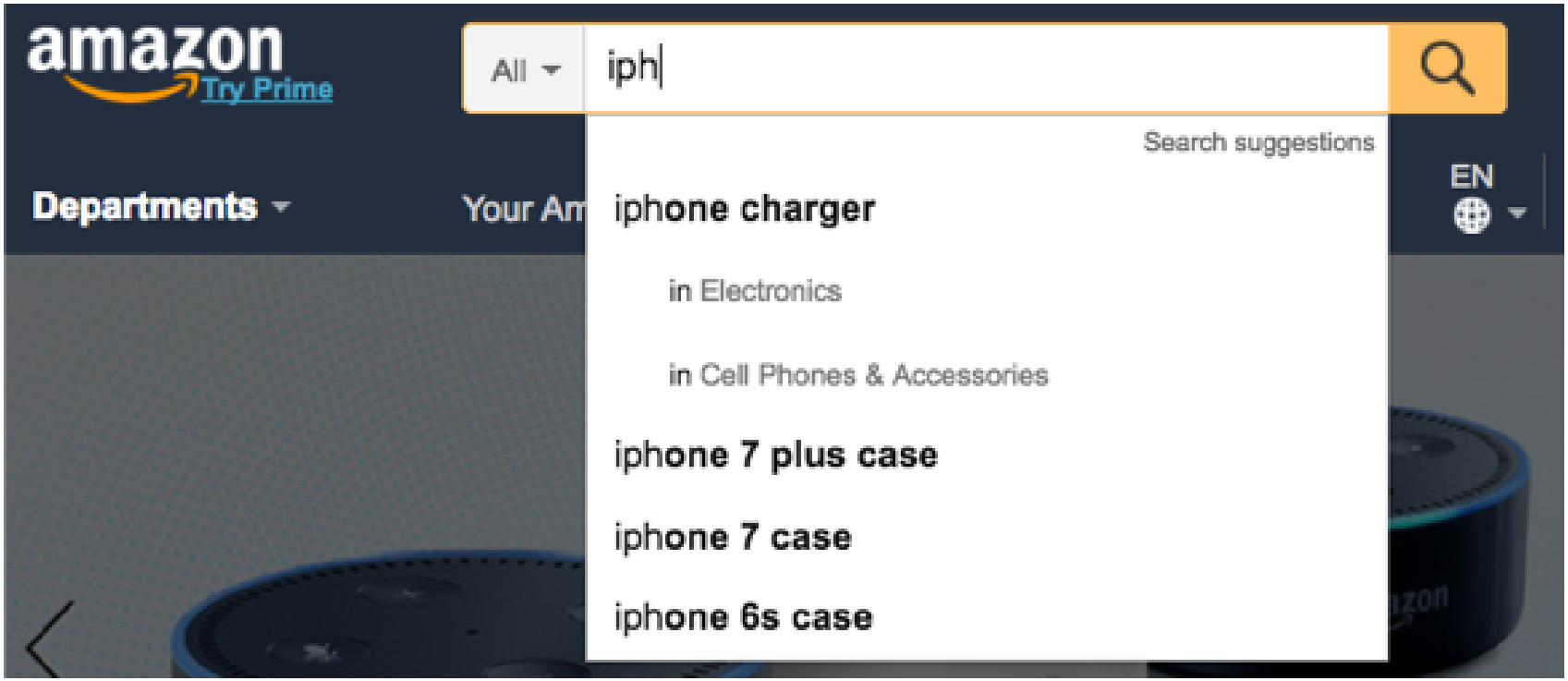
Outline

1. Introduction
   - Applications at Amazon Search
   - Recurrent Neural Networks
   - The Vanishing Gradient Problem
2. Proposed Solution
   - Spectral RNN
   - Training algorithms
   - Extensions
3. Generalization Analysis
4. Experimental Results
   - Synthetic Addition Task
   - Auto-complete Task

Customer Interactions on Amazon.com



Query Auto-completion



Query Auto-completion

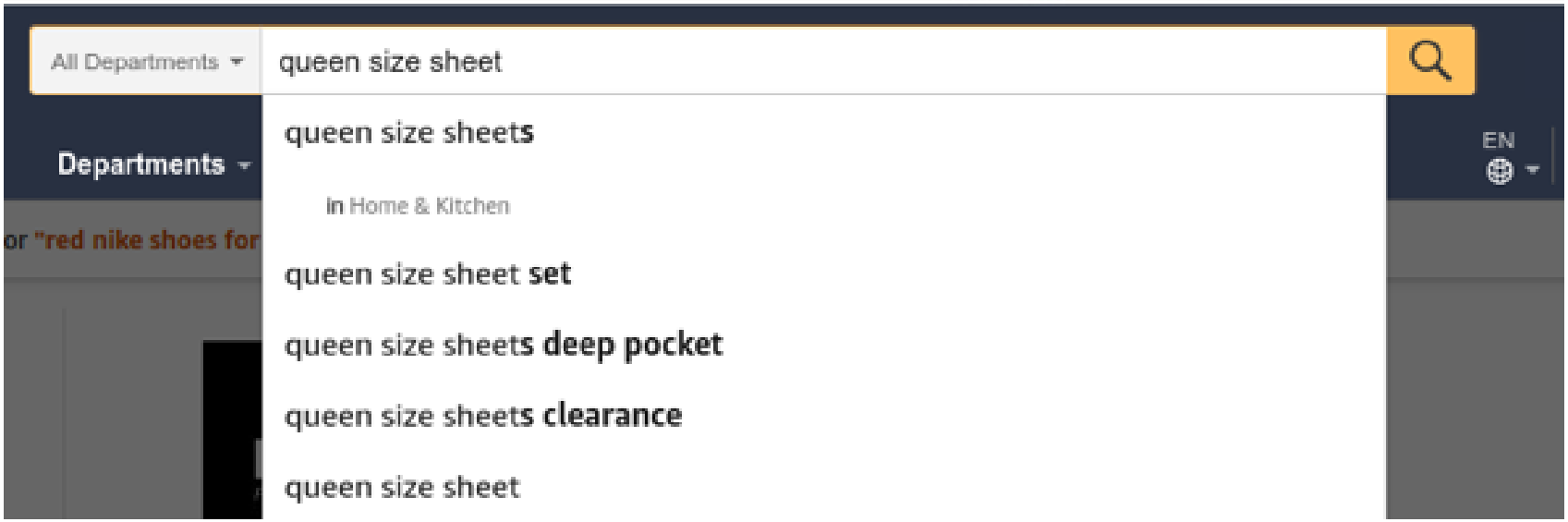
- Query-autocomplete is integral to the Amazon search experience, and leads to a substantial number of search requests
- **Challenge:**
  - Respond faster than $>$ 100MM customers typing
  - Respond to all customer requests
  - Serve different types of customers and shopping missions
    - Shopping for Thanksgiving deals on Amazon.com
    - Searching for Music on Amazon Music
    - Watching Man in the High Castle in Prime Video

Memory-based Approach

- Memorize what customers type!
- Suggestions computed from search behavior
  - Aggregate several days of search logs
  - Generate prefixes for each query
  - Rank prefix suggestion pairs
  - Retain top suggestions
- Store prefix-suggestions pairs in RODB
- **Problem:**
  - Memorizing typed queries do not generalize
  - No suggestion is given for unseen prefixes (potentially leading to unseen queries)
  - Many prefixes have no completion using RODB approach

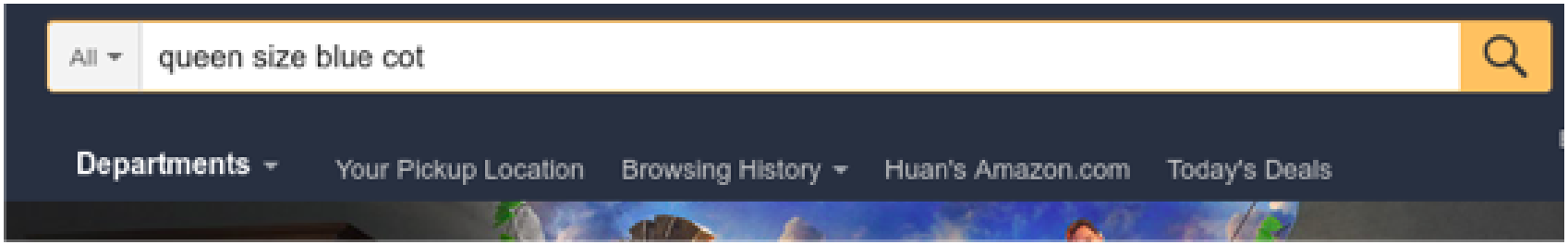
Example

- The prefix “queen size sheet”:



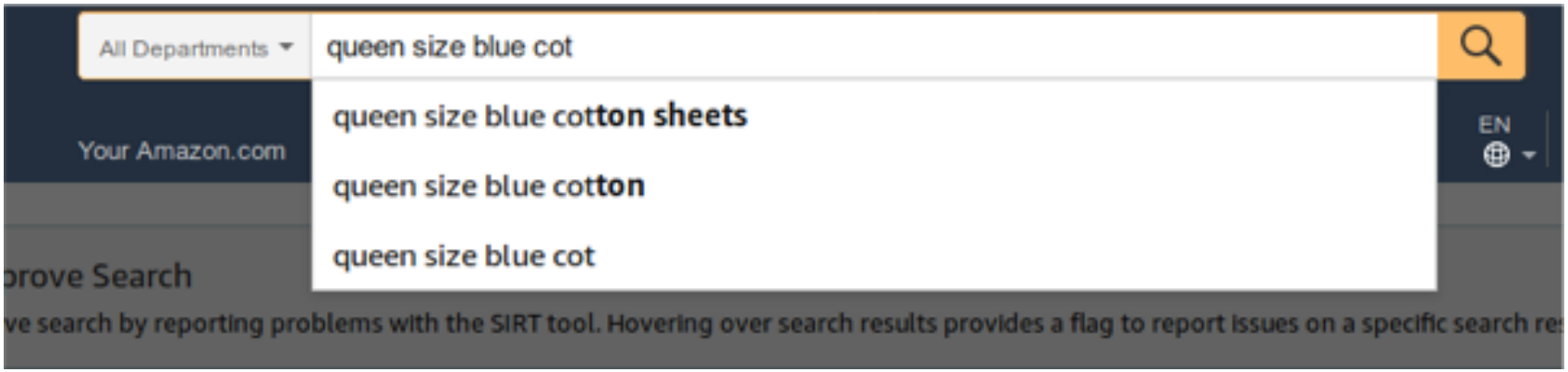
Example

- No completion provided for prefix: "queen size blue cot"



Example - Using Deep Learning

- Deep learning(DL) model is able to generate unseen queries
- DL model learnt that “cotton sheets” is likely to follow this prefix and the context “queen size” and “blue”



Example - Using Deep Learning

- Can help customers find the product and make purchases.

400 Thread Count Sheet Set, 100% Long Staple Cotton Light Blue Queen Sheets, Sateen Weave Bed Sheets fit upto 17 inch Deep Pockets, 4Pc Set by Pizuna Linens (Baby Blue Queen 100% Cotton Sheet Set)

by Pizuna Linens

$39.99 prime

FREE Shipping on eligible orders

182

Product Features

... cotton for extra softness and durability. Sizes - Flat sheet: 90 ...

Thread Spread True Luxury 100% Egyptian Cotton - Genuine 1000 Thread Count 4 Piece Sheet Set- Color Light Blue,Size Queen - Fits Mattress Upto 18" Deep Pocket

by Thread Spread

$89.99 prime

FREE Shipping on eligible orders

1,333

Product Features

... against crisp Cotton Sateen. Queen Sheet Set Sizes : One 90 x 102 ...

Solid Navy Queen Size Sheets, 4PC Bed Sheet Set, 100% Cotton, 300 Thread Count, Sateen Solid, Deep Pocket, by Royal Hotel

by Royal Hotel

$49.99 prime

FREE Shipping on eligible orders

Only 5 left in stock - order soon.

340

Product Features

100% Luxury 300 TC Long-Staple Combed Cotton 4pc Queen

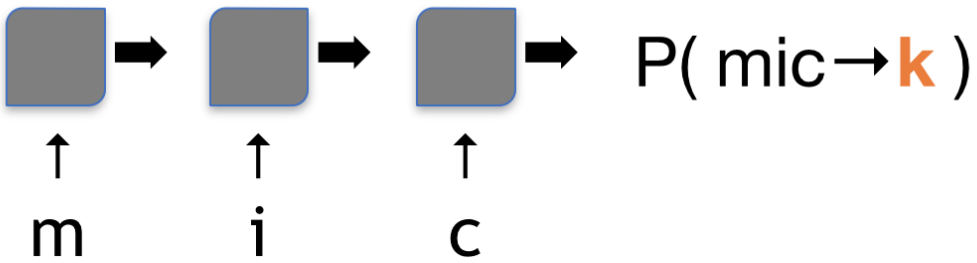
Probabilistic language Model

- We can do query completion if we know the probability P(prefix → query)
  - P(mic → key mouse)=0.25
  - P(mic → cro sd card)=0.36 → **Top candidate**
  - P(mic → hael kors handbags)=0.08
  - P(mic → key bluetooth laser fidget spinner)=0.001

- P(any prefix → any completed query)
- Extremely large space, need to decompose

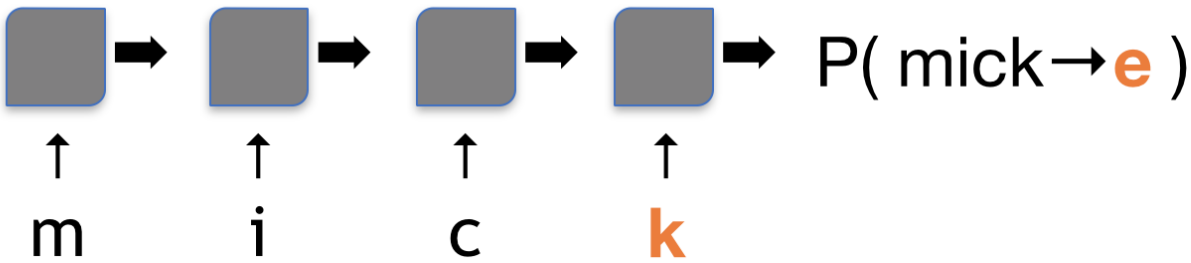
Conditional Probability

- Probability of completed query under prefix P(prefix → query)
- P(mic→ key) = P(mic→ k)· · ·



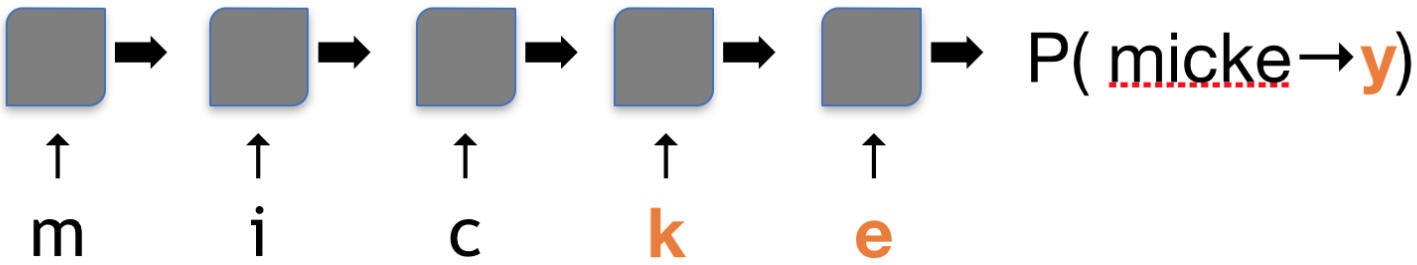
Conditional Probability

- Probability of completed query under prefix P(prefix $\rightarrow$ query)
- P(mic$\rightarrow$ key) = P(mic$\rightarrow$ k)P(mick$\rightarrow$ e)$\cdots$



Conditional Probability

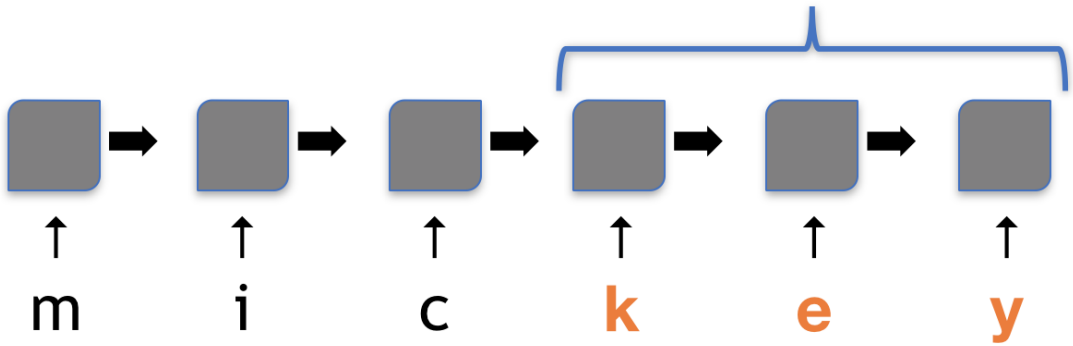
- Probability of completed query under prefix P(prefix → query)
- P(mic→ key) = P(mic→ k)P(mick→ e)P(micke→ y)



Conditional Probability

- Probability of completed query under prefix
  $P(\text{prefix} \rightarrow \text{query})$

$$P(\text{mic} \rightarrow \text{key}) = P(\text{mic} \rightarrow \text{k})P(\text{mick} \rightarrow \text{e})P(\text{micke} \rightarrow \text{y})$$
$$= \Pi \;\; P(\text{next character}|\text{current prefix})$$

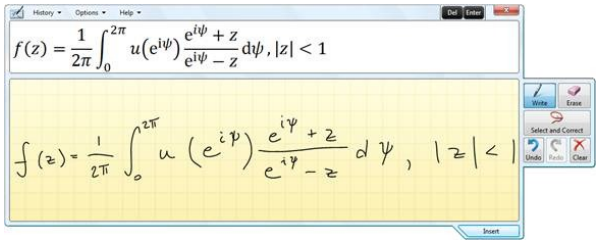


- What model to use?
  - Exploit recent success in Deep Learning: seq2seq models!

Outline

1. Introduction
   - Applications at Amazon Search
   - Recurrent Neural Networks
   - The Vanishing Gradient Problem
2. Proposed Solution
   - Spectral RNN
   - Training algorithms
   - Extensions
3. Generalization Analysis
4. Experimental Results
   - Synthetic Addition Task
   - Auto-complete Task

Things we can do with Recurrent Neural Networks



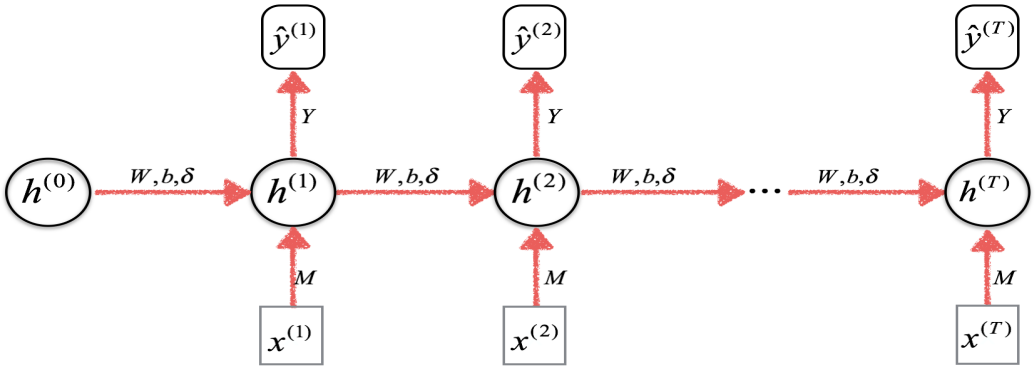
Handwriting Recognition

Translation

Question answering

Speech Recognition

Recurrent Neural Networks (RNN)



- RNN with activation function ϕ:

$$h^{(t)} = \phi(Wh^{(t-1)} + Mx^{(t)} + b) \tag{1}$$
$$\hat{y}^{(t)} = Yh^{(t)}$$

- Note:
  - Parameters $\Theta = \{W, M, b, Y\}$ are shared at all time steps
  - Input $x_i = \{x_i^{(1)}, x_i^{(2)}, \ldots x_i^{(T)}\}$ is fed sequentially
  - Output $\hat{y}_i = \{\hat{y}_i^{(1)}, \hat{y}_i^{(2)}, \ldots \hat{y}_i^{(T)}\}$ evaluated at each time step
- Loss is measured as: $L(X, Y; \Theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \hat{y}_i)$

Loss Functions

- Given dataset $\{x_i, y_i\}_{i=1}^N$, the loss is measured as:

$$L(X, Y; \Theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \hat{y}_i)$$

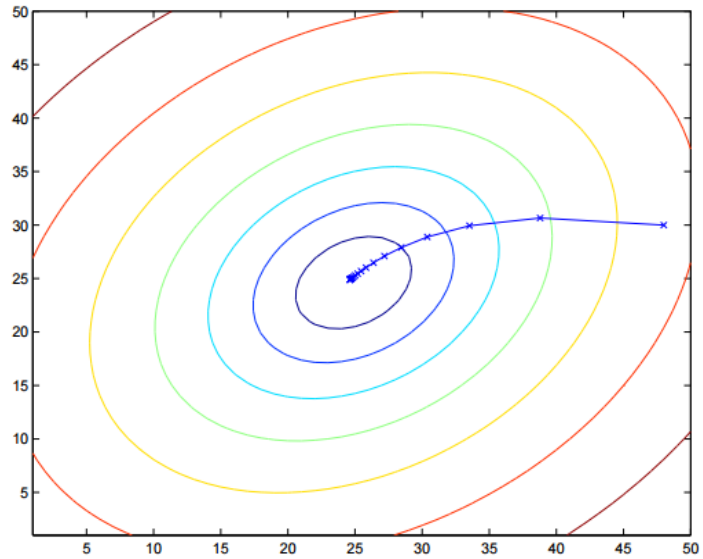
- For regression problems, ℓ could be squared loss:

$$\ell(y_i, \hat{y}_i) = \frac{1}{T} \sum_{t=1}^{T} \left\| y_i^{(t)} - \hat{y}_i^{(t)} \right\|^2$$

- For classification problems, ℓ could be cross entropy loss:

$$\ell(y_i, \hat{y}_i) = \frac{1}{T} \sum_{t=1}^{T} KL(y_i^{(t)} || \text{softmax}(\hat{y}_i^{(t)}))$$

Learning the Parameters: Gradient descent



- To minimize loss $L(\Theta) = \sum_{i=1}^{N} \ell(y_i; \hat{y}_i)$, we can conduct gradient descent to update parameters $\theta \in \Theta = \{W^{(t)}, b^{(t)}\}_{t=1}^{T}$:

$$\theta \leftarrow \theta - \frac{\eta}{N} \sum_{i \in [N]} \frac{\partial \ell}{\partial \theta}(y_i; \hat{y}_i)$$

- **Problem:** Even one iteration is too expensive for huge N

Stochastic Gradient Descent (SGD)

SGD to train a neural network:

- A minibatch $B \subset [N]$ is sampled.
- Each parameter $\theta \in \{W^{(t)}, b^{(t)}\}_{t=1}^{T}$ is updated using an estimate of the gradient:

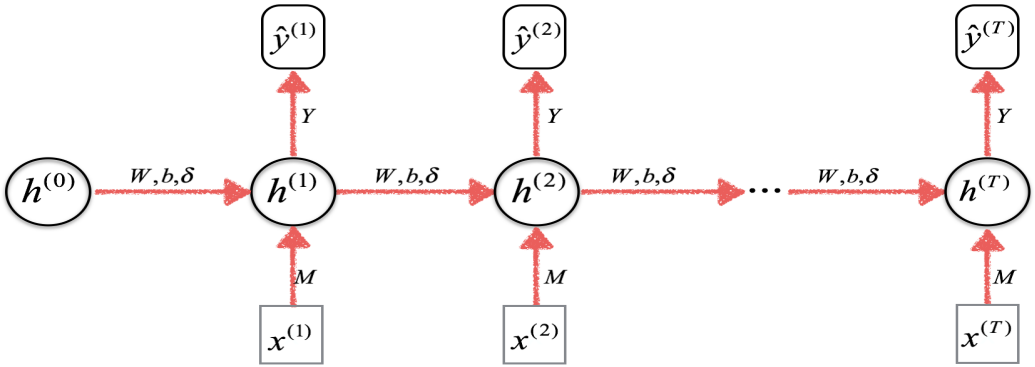
$$\theta \leftarrow \theta - \frac{\eta}{|B|} \sum_{i \in B} \frac{\partial \ell}{\partial \theta}(y_i; \hat{y}_i)$$

- Step size η is usually selected by line search or heuristics like Adam (Kingma et al. 2014).

Outline

1. Introduction
   - Applications at Amazon Search
   - Recurrent Neural Networks
   - The Vanishing Gradient Problem
2. Proposed Solution
   - Spectral RNN
   - Training algorithms
   - Extensions
3. Generalization Analysis
4. Experimental Results
   - Synthetic Addition Task
   - Auto-complete Task

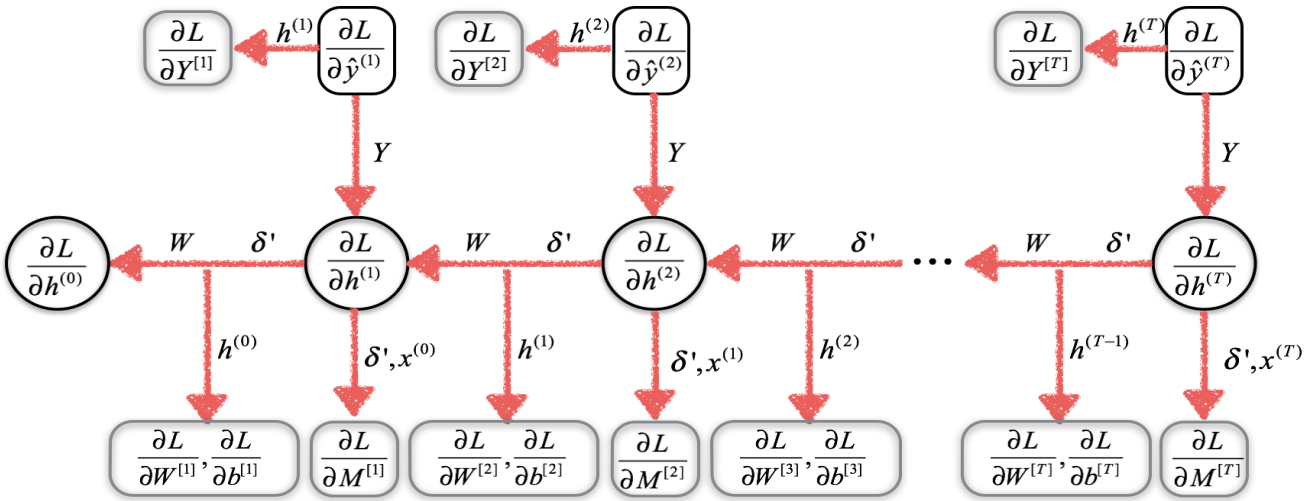
RNN: forward propagation



Forward propagation:

- Evaluate activations
  $h^{(t)} = \phi(Wh^{(t-1)} + Mx^{(t)} + b), t = 1, \ldots, T$
- Evaluate $\hat{y}^{(t)} = Yh^{(t)}$, $t = 1, \ldots, T$

RNN: backward propagation



- Gradients propagate back "through time":
  $\frac{\partial h^{(t)}}{\partial W} = h^{(t-1)} \circ diag(\phi_t'); \frac{\partial h^{(t)}}{\partial h^{(t-1)}} = W^\top diag(\phi_t')$
- Backpropagation:

$$\frac{\partial \ell}{\partial h^{(t)}} \propto \Pi_{i=T}^{t+1} W^\top diag(\phi_i')$$

- Hence, gradients can easily "explode" or "vanish"
- Problem: Long-range dependencies cannot be captured

Sigmoid Activation Function

- Sigmoid function $\sigma(z) = 1/(1 + e^{-z})$ has gradient:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) = \frac{1}{e^z + e^{-z} + 2}$$

- $|z|$ large $\Rightarrow$ vanishing gradients
- Solution indicated: constrain size of each h

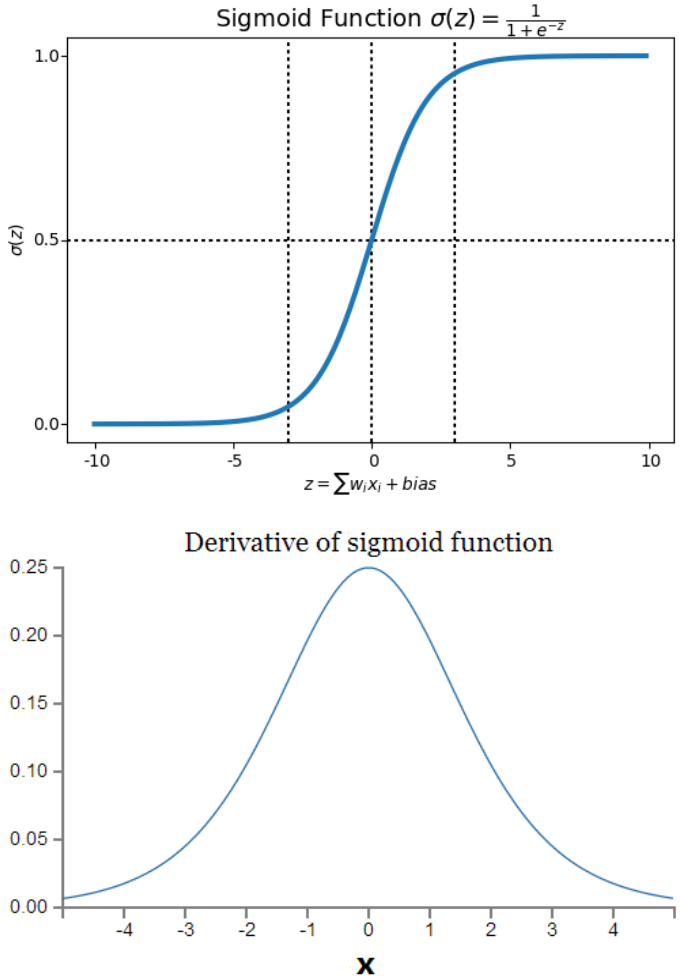
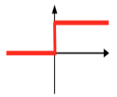
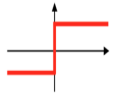
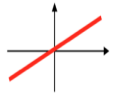
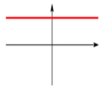
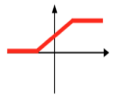
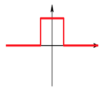
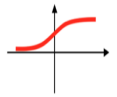
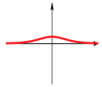
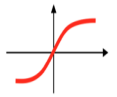
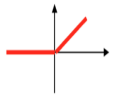
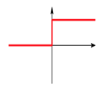
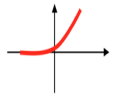


Figure: Sigmoid activation

Activation Functions: Vanishing Gradients

- Saturated activation functions (esp. sigmoid or tanh) $\Rightarrow$ vanishing gradients
- Absolute value of input is large for saturated activations $\Rightarrow$ vanishing gradients
- Solution indicated: constrain size of each h

Activation function	Equation	Example	1D Graph	Derivative
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant		
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant		
Linear	$\phi(z) = z$	Adaline, linear regression		
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine		
Logistic (sigmoid)	$\phi(z) = \frac{1}{1+e^{-z}}$	Logistic regression, Multi-layer NN		
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks		
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks		
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks		

Copyright © Sebastian Raschka 2016 (http://sebastianraschka.com)

Figure: Common activation functions and their derivatives.

Existing Architectural Solutions

- Long Short-Term Memory (LSTM):

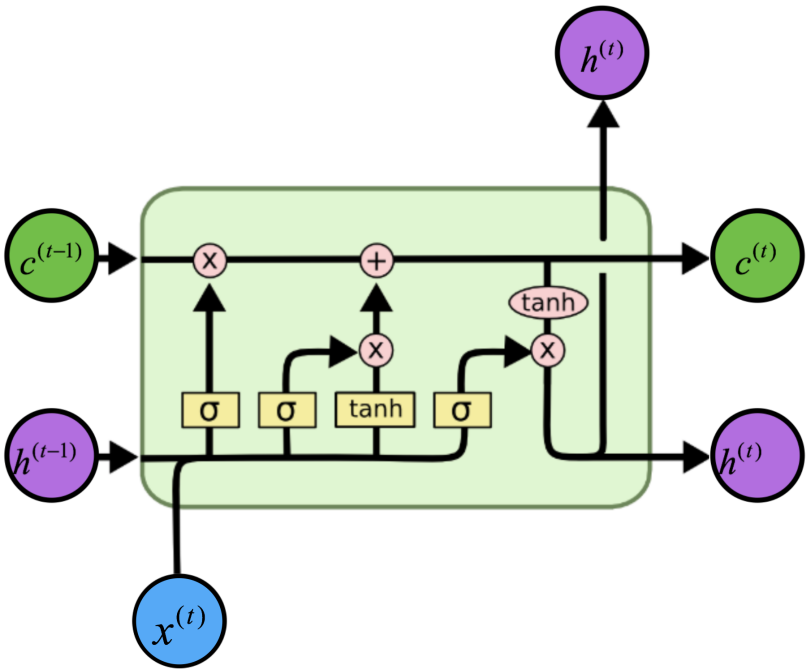


Figure: One node of LSTM. (Colah 2015)

$$
\begin{aligned}
f^{(t)} &= \sigma_g(U_f x^{(t)} + W_f h^{(t-1)} + b_f) \\
i^{(t)} &= \sigma_g(U_i x^{(t)} + W_i h^{(t-1)} + b_i) \\
o^{(t)} &= \sigma_g(U_o x^{(t)} + W_o h^{(t-1)} + b_o) \\
\widetilde{c}^{(t)} &= \sigma_c(U_c x^{(t)} + W_c h^{(t-1)} + b_c) \\
c^{(t)} &= f^{(t)} * c^{(t-1)} + i^{(t)} * \widetilde{c}^{(t)} \\
h^{(t)} &= o^{(t)} * \sigma_h(c^{(t)})
\end{aligned}
$$

- Avoid long-term dependency issues by "forget-gates"

Existing Architectural Solutions

- Gated Recurrent Unit (GRU):

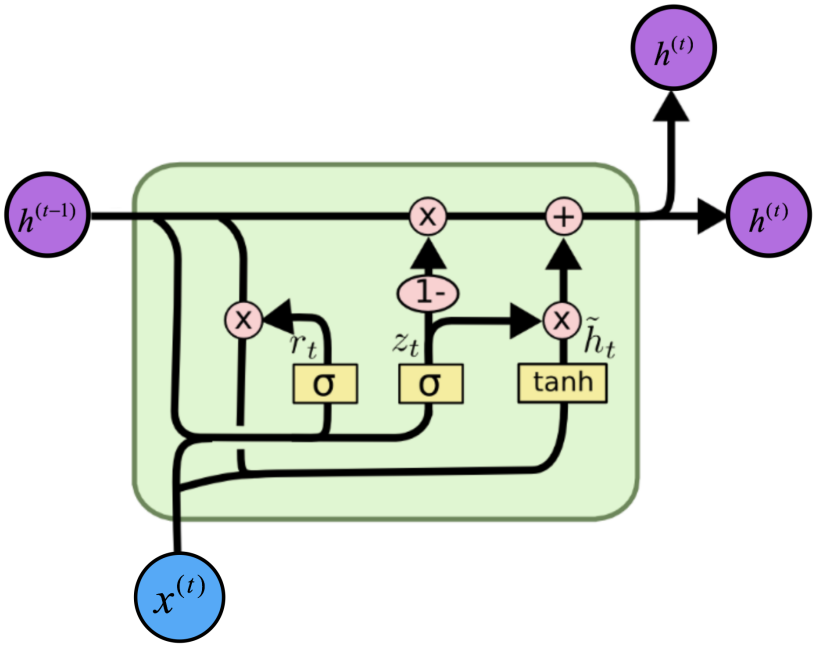
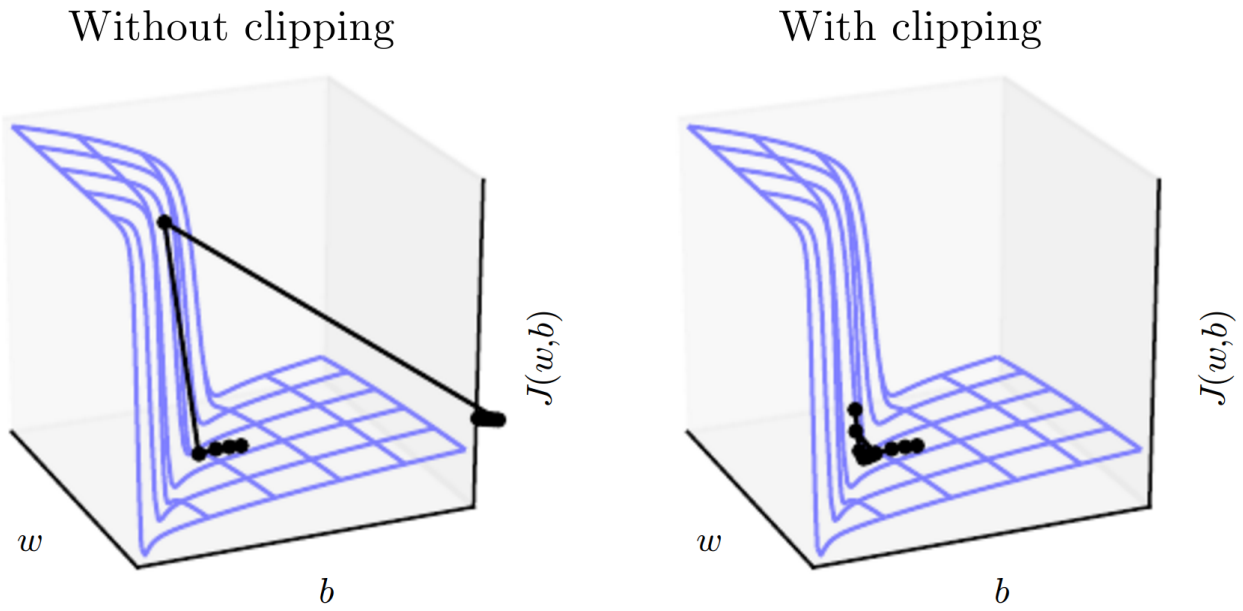


Figure: One node of GRU. (Cho, et al. (2014))

$$z^{(t)} = \sigma_g(U_z x^{(t)} + W_z h^{(t-1)} + b_z)$$
$$r^{(t)} = \sigma_g(U_r x^{(t)} + W_r h^{(t-1)} + b_r)$$
$$\widetilde{h}^{(t)} = \sigma_h(U_h x^{(t)} + W_h(r^{(t)} * h^{(t-1)}) + b_h)$$
$$h^{(t)} = (1 - z^{(t)}) * h^{(t-1)} + z^{(t)} * \widetilde{h}^{(t)}$$

Existing Solutions

- Gradient Clipping (Pascanu et al. 2013)



- Initialization with identity/orthogonal matrix (Le, Jaitly & Hinton 2015)

Existing Solutions

- Solution: Keep $W^\top W = I$
  1. uRNN (Arjovsky et al., 2016)
     - $W \in \mathbb{C}^{n \times n}$ is product of reflection, diagonal, and Fourier transform matrices
  2. Full-Capacity uRNN (Wisdom et al., 2016)
  3. unitary RNN (Hyland & Ratsch , 2017)
     - Allow $W \in \mathbb{C}^{n \times n}$ to span the whole unitary group
  4. oRNN (orthogonal RNN) (Mhammedi et al., 2017)
     - Allow W to span the whole orthogonal space by using Householder reflectors
- Problem: lose expressive power

Existing Solutions

- Solution: encourage orthogonality
  1. factorized RNN (Vorontsov et al. 2017)
     - Parameterize $W = U\Sigma V^\top$, encourage Σ to be close to 1, and update U, V by Cayley transform
  2. Parseval networks (Cisse et al. 2017)
     - Regularize with $\|I - W^\top W\|_F^2$
- Problem: high time complexity

- Question: How to solve the gradient vanishing/exploding problem with full expressive power and high efficiency?

Large-Scale LSTMs in Action

Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu,
Zhifeng Chen, Nikhil Thorat
melvinp,schuster,qvl,krikun,yonghui,zhifengc,nsthorat@google.com

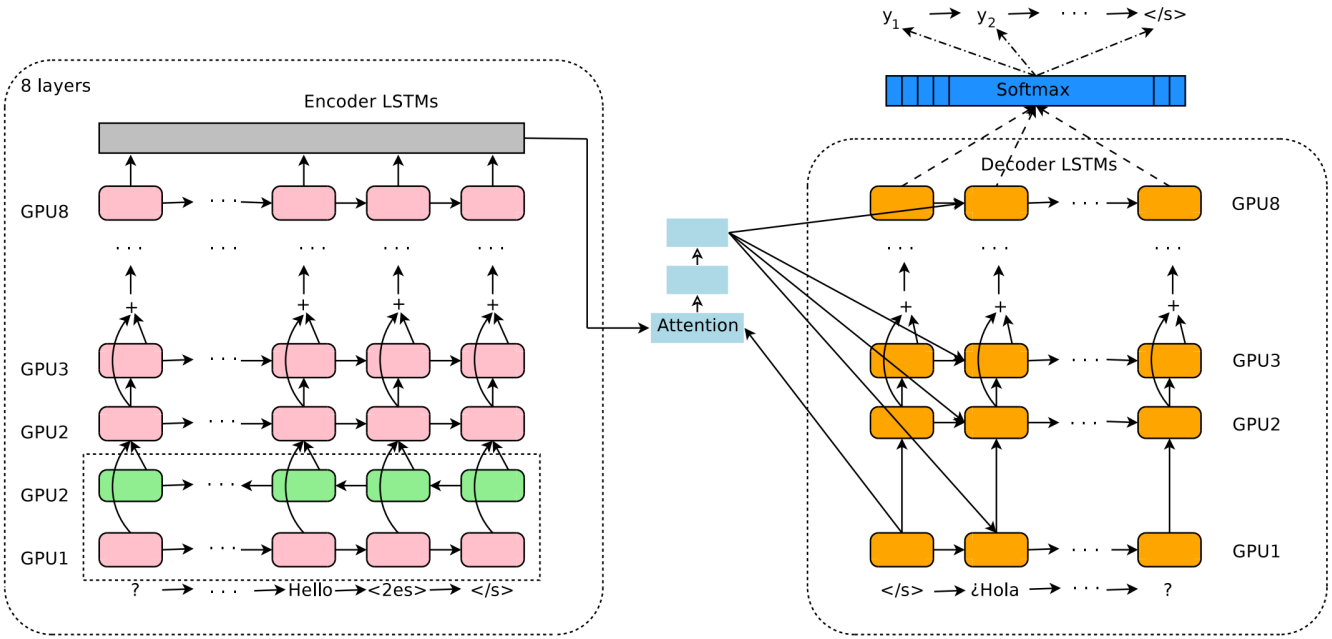
Fernanda Viégas, Martin Wattenberg, Greg Corrado,
Macduff Hughes, Jeffrey Dean

Abstract

We propose a simple solution to use a single Neural Machine Translation (NMT) model to translate between multiple languages. Our solution requires no changes to the model architecture from a standard NMT system but instead introduces an artificial token at the beginning of the input sentence to specify the required target language. The rest of the model, which includes an encoder, decoder and attention module, remains unchanged and is shared across all languages. Using a shared wordpiece vocabulary, our approach enables Multilingual NMT using a single model without any increase in parameters, which is significantly simpler than previous proposals for Multilingual NMT. On the WMT'14 benchmarks, a single multilingual model achieves comparable performance for English→French and surpasses state-of-the-art results for English→German. Similarly, a single multilingual model surpasses state-of-the-art results for French→English and German→English on WMT'14 and WMT'15 benchmarks, respectively. On production corpora, multilingual models of up to twelve language pairs allow for better translation of many individual pairs. In addition to improving the translation quality of language pairs that the model was trained with, our models can also learn to perform implicit bridging between language pairs never seen explicitly during training, showing that transfer learning and zero-shot translation is possible for neural translation. Finally, we show analyses that hints at a universal interlingua representation in our models and show some interesting examples when mixing languages.

Google Neural Machine Translation Architecture

- Eight-layer bidirectional LSTM with Attention (Johnson et. al., 2016)
- For each language pair: 1024 nodes (hidden dimension), 8 LSTM layers with a total of 255M parameters.
- Total training time is on the scale of weeks, on up to 100 GPUs.

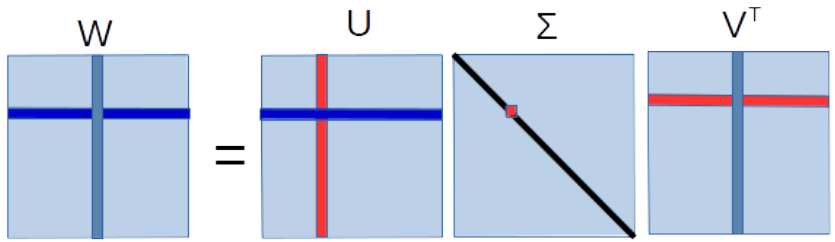


Outline

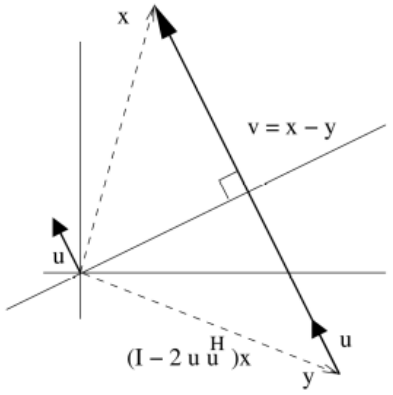
1. Introduction
   - Applications at Amazon Search
   - Recurrent Neural Networks
   - The Vanishing Gradient Problem
2. Proposed Solution
   - Spectral RNN
   - Training algorithms
   - Extensions
3. Generalization Analysis
4. Experimental Results
   - Synthetic Addition Task
   - Auto-complete Task

Spectral Parameterization

- An illustration of the parameterization process:



A Householder reflector:



Represent U as:

$$\left(I_n - 2\frac{u_n u_n^\top}{\|u_n\|^2} \right) \cdot \begin{pmatrix} 1 & \\ & I_{n-1} - 2\frac{u_{n-1} u_{n-1}^\top}{\|u_{n-1}\|^2} \end{pmatrix} \cdots \begin{pmatrix} I_{n-k_1} & \\ & I_{k_1} - 2\frac{u_{k_1} u_{k_1}^\top}{\|u_{k_1}\|^2} \end{pmatrix}$$

Spectral Parameterization

- Maintain SVD of transition matrix:

$$W = U\Sigma V^\top$$

Spectral Parameterization

- Maintain SVD of transition matrix:

$$W = U\Sigma V^\top$$

- Parameterize U, V by products of Householder reflectors:

$$\mathcal{H}_k(u) = \begin{cases} \begin{pmatrix} I_{n-k} & \\ & I_k - 2\frac{uu^\top}{\|u\|^2} \end{pmatrix} & , \quad u \neq \mathbf{0} \\ I_n & , \quad \text{otherwise.} \end{cases}$$

Spectral Parameterization

- Maintain SVD of transition matrix:

$$W = U\Sigma V^\top$$

- Parameterize U, V by products of Householder reflectors:

$$\mathcal{H}_k(u) = \begin{cases} \begin{pmatrix} I_{n-k} & \\ & I_k - 2\frac{uu^\top}{\|u\|^2} \end{pmatrix} & , \quad u \neq \mathbf{0} \\ I_n & , \quad \text{otherwise.} \end{cases}$$

- $U \leftarrow \prod_{k=n}^{k_1} \mathcal{H}_k(u_k)$

Spectral Parameterization

- Maintain SVD of transition matrix:

$$W = U\Sigma V^\top$$

- Parameterize U, V by products of Householder reflectors:

$$\mathcal{H}_k(u) = \begin{cases} \begin{pmatrix} I_{n-k} & \\ & I_k - 2\frac{uu^\top}{\|u\|^2} \end{pmatrix} & , \quad u \neq \mathbf{0} \\ I_n & , \quad \text{otherwise.} \end{cases}$$

- $U \leftarrow \prod_{k=n}^{k_1} \mathcal{H}_k(u_k)$
- $V \leftarrow \prod_{k=n}^{k_2} \mathcal{H}_k(v_k)$

Spectral Parameterization

Proposed parametrization:

$$\begin{aligned}\mathcal{M}_{k_1,k_2} :&(\mathbb{R}^{k_1} \times ... \times \mathbb{R}^n) \times (\mathbb{R}^{k_2} \times ... \times \mathbb{R}^n) \times (\mathbb{R}^n) \mapsto \mathbb{R}^{n \times n} \\ &\left(\{u_i\}_{i=k_1}^n\right), \left(\{v_i\}_{i=k_2}^n\right), (\sigma) \mapsto \\ &\underbrace{\mathcal{H}_n(u_n)...\mathcal{H}_{k_1}(u_{k_1})}_{U} \underbrace{diag(\sigma)}_{\Sigma} \underbrace{\mathcal{H}_{k_2}(v_{k_2})...\mathcal{H}_n(v_n)}_{V^\top}. \end{aligned} \tag{2}$$

- **Singular values are explicit**:
  $\mathcal{M}_{k_1,k_2}(\{u_i\}_{i=k_1}^n, \{v_i\}_{i=k_2}^n, \sigma)$ is an $n \times n$ real matrix with singular values σ.

Spectral Parameterization

Proposed parametrization:

$$\begin{aligned}\mathcal{M}_{k_1,k_2} :&(\mathbb{R}^{k_1} \times ... \times \mathbb{R}^n) \times (\mathbb{R}^{k_2} \times ... \times \mathbb{R}^n) \times (\mathbb{R}^n) \mapsto \mathbb{R}^{n \times n} \\ &\left(\{u_i\}_{i=k_1}^n\right), \left(\{v_i\}_{i=k_2}^n\right), (\sigma) \mapsto \\ &\underbrace{\mathcal{H}_n(u_n)...\mathcal{H}_{k_1}(u_{k_1})}_{U} \underbrace{diag(\sigma)}_{\Sigma} \underbrace{\mathcal{H}_{k_2}(v_{k_2})...\mathcal{H}_n(v_n)}_{V^\top}. \end{aligned} \tag{2}$$

- **Singular values are explicit**: $\mathcal{M}_{k_1,k_2}(\{u_i\}_{i=k_1}^n, \{v_i\}_{i=k_2}^n, \sigma)$ is an $n \times n$ real matrix with singular values σ.
- **Full expressivity**: The image of $\mathcal{M}_{1,1}$ is the set of $n \times n$ real matrices.

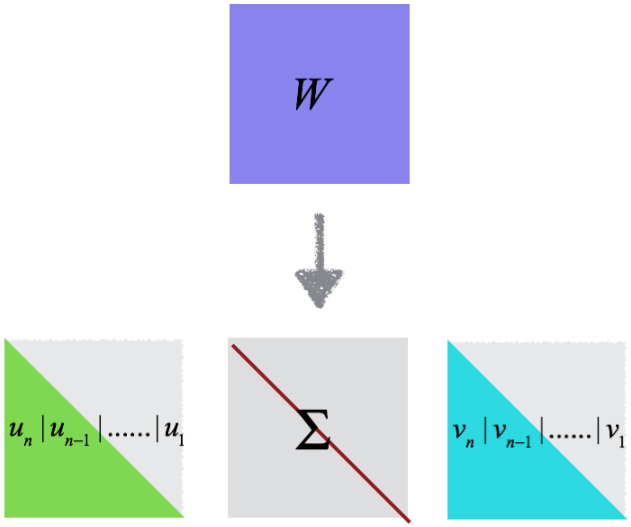
Spectral Parameterization

Proposed parametrization:

$$\begin{aligned}\mathcal{M}_{k_1,k_2} :&(\mathbb{R}^{k_1} \times ... \times \mathbb{R}^n) \times (\mathbb{R}^{k_2} \times ... \times \mathbb{R}^n) \times (\mathbb{R}^n) \mapsto \mathbb{R}^{n \times n} \\ &\left(\{u_i\}_{i=k_1}^n\right), \left(\{v_i\}_{i=k_2}^n\right), (\sigma) \mapsto \\ &\underbrace{\mathcal{H}_n(u_n)...\mathcal{H}_{k_1}(u_{k_1})}_{U} \underbrace{diag(\sigma)}_{\Sigma} \underbrace{\mathcal{H}_{k_2}(v_{k_2})...\mathcal{H}_n(v_n)}_{V^\top}. \end{aligned} \tag{2}$$

- **Singular values are explicit**: $\mathcal{M}_{k_1,k_2}(\{u_i\}_{i=k_1}^n, \{v_i\}_{i=k_2}^n, \sigma)$ is an $n \times n$ real matrix with singular values σ.
- **Full expressivity**: The image of $\mathcal{M}_{1,1}$ is the set of $n \times n$ real matrices.
- **Orthogonal expressivity**: The image of $\mathcal{M}_{k_1,k_2}$ covers the set of $n \times n$ orthogonal matrices if $k_1 + k_2 \leq n + 2$.

Extra notes: matching dimensions



- $\mathcal{M}_{1,1}$ seemingly maps a space of $n^2 + 2n$ dimensions to a space of n^2 dimensions.
- However, $\mathcal{H}_k(u_k)$ is invariant to the norm of u_k, so the domain of $\mathcal{M}_{1,1}$ also has exactly n^2 dimensions.

Spectral RNN: RNN with SVD parameterization

- In Spectral RNN , we parametrize the transition matrix $W \in \mathbb{R}^{n \times n}$ using $m_1 + m_2$ Householder reflectors.
- Can select m_1 and m_2 to balance expressive power versus time/space complexity. (Full expressivity if $m_1 = m_2 = n$)
- Can do both forward and backward propagation in $O(n(m_1 + m_2))$ time. (RNN: $O(n^2)$)
- Can explicitly control the singular values. For example,

$$\sigma_i = \sigma^* + 2r(\text{sigmoid}(\hat{\sigma}_i) - 0.5),\ i \in [n] \tag{3}$$

$\Rightarrow \sigma_i \in [\sigma^* - r, \sigma^* + r]$. Usually σ^* is set to 1 and $r \ll 1$.

Outline

1. Introduction
   - Applications at Amazon Search
   - Recurrent Neural Networks
   - The Vanishing Gradient Problem
2. Proposed Solution
   - Spectral RNN
   - Training algorithms
   - Extensions
3. Generalization Analysis
4. Experimental Results
   - Synthetic Addition Task
   - Auto-complete Task

Forward propagation

- Only aspect different from regular RNN in forward propagation is computation of $Wh^{(t-1)}$:

$$\begin{aligned} Wh^{(t)} =& \mathcal{H}_n(u_n)...\mathcal{H}_{n-m_1+1}(u_{n-m_1+1})diag(\sigma) \\ & \mathcal{H}_{n-m_2+1}(v_{n-m_2+1})...\mathcal{H}_n(v_n)h^{(t)} \end{aligned}$$

- Can be done efficiently through $m_1 + m_2$ inner products and vector additions. For each reflector:

$$\mathcal{H}_k(u_k)h = \left(I_n - \frac{2\hat{u}_k\hat{u}_k^\top}{\hat{u}_k^\top\hat{u}_k}\right)h = h - 2\frac{\hat{u}_k^\top h}{\hat{u}_k^\top\hat{u}_k}\hat{u}_k$$

Backward propagation

- Let $L(\{u_i\}, \{v_i\}, \sigma, M, Y, b)$ be the loss or objective function, $C^{(t)} = Wh^{(t)}, \hat{\Sigma} = diag(\hat{\sigma})$. Given $\frac{\partial L}{\partial C^{(t)}}$, we define:

$$\frac{\partial L}{\partial u_k^{(t)}} := \left[\frac{\partial C^{(t)}}{\partial u_k^{(t)}}\right]^\top \frac{\partial L}{\partial C^{(t)}}; \frac{\partial L}{\partial v_k^{(t)}} := \left[\frac{\partial C^{(t)}}{\partial v_k^{(t)}}\right]^\top \frac{\partial L}{\partial C^{(t)}};$$

$$\frac{\partial L}{\partial \Sigma^{(t)}} := \left[\frac{\partial C^{(t)}}{\partial \Sigma^{(t)}}\right]^\top \frac{\partial L}{\partial C^{(t)}}; \frac{\partial L}{\partial \hat{\Sigma}^{(t)}} := \left[\frac{\partial \Sigma^{(t)}}{\partial \hat{\Sigma}^{(t)}}\right]^\top \frac{\partial L}{\partial \Sigma^{(t)}};$$

$$\frac{\partial L}{\partial h^{(t-1)}} := \left[\frac{\partial C^{(t)}}{\partial h^{(t-1)}}\right]^\top \frac{\partial L}{\partial C^{(t)}}$$

- Back propagation for Spectral RNN requires $\frac{\partial C^{(t)}}{\partial u_k^{(t)}}$, $\frac{\partial C^{(t)}}{\partial v_k^{(t)}}$, $\frac{\partial C^{(t)}}{\partial \hat{\Sigma}^{(t)}}$ and $\frac{\partial C^{(t)}}{\partial h^{(t-1)}}$.

Backward propagation

Partial gradients can be computed iteratively ($\hat{h} := \mathcal{H}_k(u_k)h$ and $g := \frac{\partial L}{\partial \hat{h}}$):

$$\frac{\partial L}{\partial h} = \left[\frac{\partial \hat{h}}{\partial h}\right]^\top \frac{\partial L}{\partial \hat{h}} = \left(I_n - \frac{2\hat{u}_k\hat{u}_k^\top}{\hat{u}_k^\top\hat{u}_k}\right) g = g - 2\frac{\hat{u}_k^\top g}{\hat{u}_k^\top\hat{u}_k}\hat{u}_k$$

$$\frac{\partial L}{\partial \hat{u}_k} = \left[\frac{\partial \hat{h}}{\partial \hat{u}_k}\right]^\top \frac{\partial L}{\partial \hat{h}} = -2\left(\frac{\hat{u}_k^\top h}{\hat{u}_k^\top\hat{u}_k} I_n + \frac{1}{\hat{u}_k^\top\hat{u}_k} h\hat{u}_k^\top - 2\frac{\hat{u}_k^\top h}{(\hat{u}_k^\top\hat{u}_k)^2}\hat{u}_k\hat{u}_k^\top\right) g$$

$$= -2\frac{\hat{u}_k^\top h}{\hat{u}_k^\top\hat{u}_k} g - 2\frac{\hat{u}_k^\top g}{\hat{u}_k^\top\hat{u}_k} h + 4\frac{\hat{u}_k^\top h}{\hat{u}_k^\top\hat{u}_k}\frac{\hat{u}_k^\top g}{\hat{u}_k^\top\hat{u}_k}\hat{u}_k$$

- Thus backward propagation can also be done in $O((m_1 + m_2)n)$ time.

Outline

1. Introduction
   - Applications at Amazon Search
   - Recurrent Neural Networks
   - The Vanishing Gradient Problem
2. Proposed Solution
   - Spectral RNN
   - Training algorithms
   - Extensions
3. Generalization Analysis
4. Experimental Results
   - Synthetic Addition Task
   - Auto-complete Task

Extension to Non-square matrices

- For any real matrix $W \in \mathbb{R}^{m \times n}$ (assume $m < n$) with reduced SVD:

$$W = U(\Sigma|0)(V_L|V_R)^\top = U\Sigma V_L^\top$$

where $U \in \mathbb{R}^{m \times m}$, $\Sigma \in diag(\mathbb{R}^m)$,$V_L \in \mathbb{R}^{n \times m}$.

  - There exist $u_n, ..., u_{k_1}$ and $v_n, ..., v_{k_2}$ s.t. $U = \mathcal{H}_m^m(u_m)...\mathcal{H}_{k_1}^m(u_{k_1})$, $V = \mathcal{H}_n^n(v_n)...\mathcal{H}_{k_2}^n(v_{k_2})$.
  - SVD parameterization for any matrix:

$$\begin{aligned}\mathcal{M}_{k_1,k_2}^{m,n} :&(\mathbb{R}^{k_1} \times ... \times \mathbb{R}^m) \times (\mathbb{R}^{k_2} \times ... \times \mathbb{R}^n) \times (\mathbb{R}^{\min(m,n)}) \mapsto \mathbb{R}^{m \times n} \\ &\left(\{u_i\}_{i=k_1}^m\right), \left(\{v_i\}_{i=k_2}^m\right), (\sigma) \mapsto \\ &\left(\mathcal{H}_m^m(u_m) \cdots \mathcal{H}_{k_1}^m(u_{k_1})\right)(\Sigma)\left(\mathcal{H}_{k_2}^n(v_{k_2}) \cdots \mathcal{H}_n^n(v_n)\right).\end{aligned}$$

- Can be applied to any Deep MLP (offers an alternative to Residual Networks)

Direct conclusions from the gradients

For Spectral RNN, recall Gradient for activations is:

$$\frac{\partial h^{(T)}}{\partial h^{(t_0)}} = \prod_{T \geq t \geq t_0} \frac{\partial h^{(t)}}{\partial h^{(t-1)}} = \prod_{T \geq t \geq t_0} W^\top \mathrm{diag}(\phi'(h^{(t-1)}))$$

- Solves the exploding gradient problem:

$$\|W\|_2 \leq 1 + \epsilon \implies \left\| \frac{\partial h^{(T)}}{\partial h^{(0)}} \right\| \leq (1+\epsilon)^T$$

- Mitigates the vanishing gradient problem:

$$\sigma_{\min}(W) \geq 1 - \epsilon \implies \sigma_{\min}\left(\frac{\partial h^{(T)}}{\partial h^{(0)}} \right) \geq \min |\phi'|^T (1-\epsilon)^T$$

Spectral MLP

- Generalization of MLP is bounded by its spectral Lipschitz constant L for a T-layered network as in (Bartlett et al. 2017), or similarly bounded by $\|W\|_2^T$ as in (Neyshabur et al. 2017)
  - Spectral MLP guarantees $L \leq \|W\|^T \leq (1+\epsilon)^T$, if we control singular values s.t. $\|W\|_2 \leq 1+\epsilon$
  - Generalization guarantee for recurrent neural network is unclear yet in the literature

- Weight matrices are Parseval tight frames $\Longrightarrow$ robustness in predictions (Cisse et al. 2017)
  - Spectral MLP guarantees near orthogonal weight matrix, namely tight frames

Generalization Analysis

- Generalization bound using PAC-Bayes (McAllester, 2003) analysis on the following **expected margin loss** (Neyshabur et al., 2017):

Definition (Expected Margin Loss)

For any distribution $\mathcal{D}$ and margin $\gamma > 0$, we define the expected margin loss as:

$$L_\gamma(f_w) = \mathbb{P}_{(x,y)\sim\mathcal{D}}\left[f_w(x)[y] \leq \gamma + \max_{j\neq y} f_w(x)[j]\right],$$

where $f_w(x)[y]$ is the probability of predicting y given input x with weight w.

Generalization Analysis: MLP

Generalization Bound (Neyshabur et al. 2017)

For any $B, d, h > 0$, let $f_w : \mathbb{R}^n \to \mathbb{R}^k$ be a d-layered feedforward neural network, where h is the upper bound on the number of output units in each layer. If it satisfies:

- Input x is bounded: $\|x\| \leq B, \forall x$,
- Activation ϕ is ReLU.

Then for any $\delta, \gamma > 0$, with probability $\geq 1 - \delta$ over a training set of size m, for any $w = \text{vec}(\{W_1, W_2, \cdots W_d\})$, we have:

$$L_0(f_w) \leq \hat{L}_\gamma(f_w) + \mathcal{O}\left(\sqrt{\frac{B(w) + \ln \frac{dm}{\delta}}{m}}\right),$$

where $B(w) = \frac{B^2 d^2 h \ln(dh)}{\gamma^2} \Pi_{i=1}^d \|W_i\|_2^2 \sum_{j=1}^d \frac{\|W_j\|_F^2}{\|W_j\|_2^2}$.

Generalization Analysis: RNN

Main Theorem (ours)

For any $B, T, n > 0$, let $f_w : \mathbb{R}^{n \times T} \to \mathbb{R}^n$ be a recurrent neural network with T time steps. If it satisfies

- Bounded input $\{x^{(1)}, x^{(2)}, \cdots x^{(T)}\}$: $\|x^{(t)}\| \leq B, \forall t \leq T$.
- Shrinking activation ϕ:
  $\|\phi(x)\| \leq \|x\|, \|\phi(x) - \phi(y)\| \leq \|x - y\|, \forall x, y$.

Then, for any $\delta, \gamma > 0$, with probability $\geq 1 - \delta$ over a training set of size m, for any $w = \text{vec}(\{W, M, Y\})$, we have:

$$L_0(f_w) \leq \hat{L}_\gamma(f_w) + \mathcal{O}\left(\sqrt{\frac{B(w) + \ln \frac{m}{\delta}}{m}}\right), \text{ where}$$

$B(w) = \frac{B^2 T^4 n \ln(n)}{\gamma^2} \max\{\|W\|_2^{2T-2}, 1\} \max\{\|M\|_2^2, 1\} \max\{\|Y\|_2^2, 1\} \|w\|^2$,
and $\|w\|^2 = \|W\|_F^2 + \|Y\|_F^2 + \|M\|_F^2$.

Outline

1. Introduction
   - Applications at Amazon Search
   - Recurrent Neural Networks
   - The Vanishing Gradient Problem
2. Proposed Solution
   - Spectral RNN
   - Training algorithms
   - Extensions
3. Generalization Analysis
4. Experimental Results
   - Synthetic Addition Task
   - Auto-complete Task

The Addition Task

Each input data includes two sequences

- top sequence: values sampled uniformly from $[0, 1]$
- bottom sequence: binary sequence with two 1's and the rest are 0
- output: the dot product between the two sequences

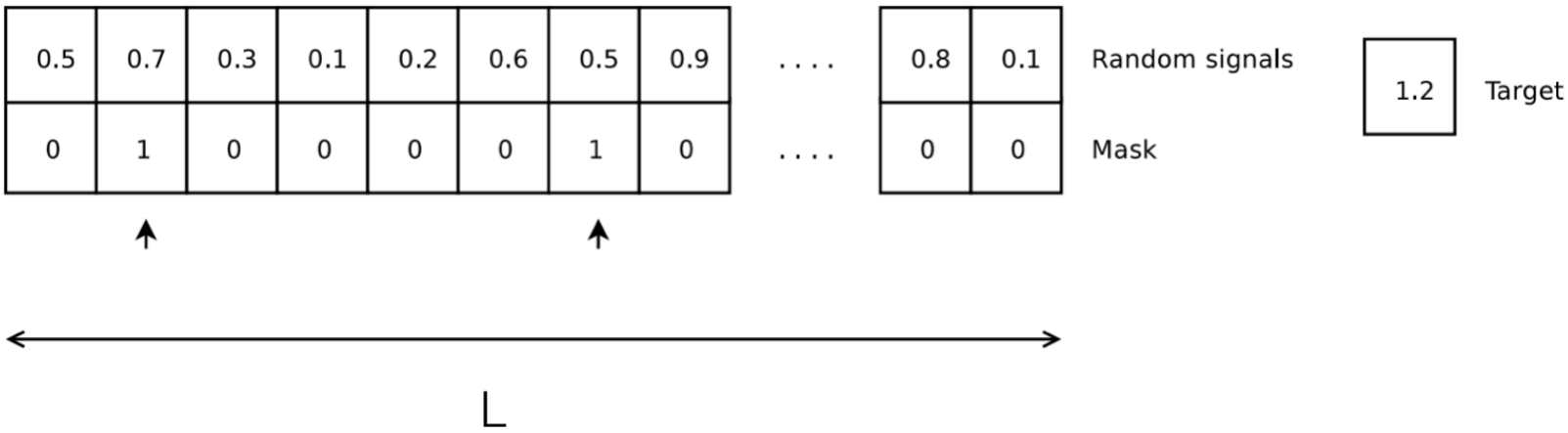


Figure: The Addition Task from (Le et. al 2015).

Addition Task: Results

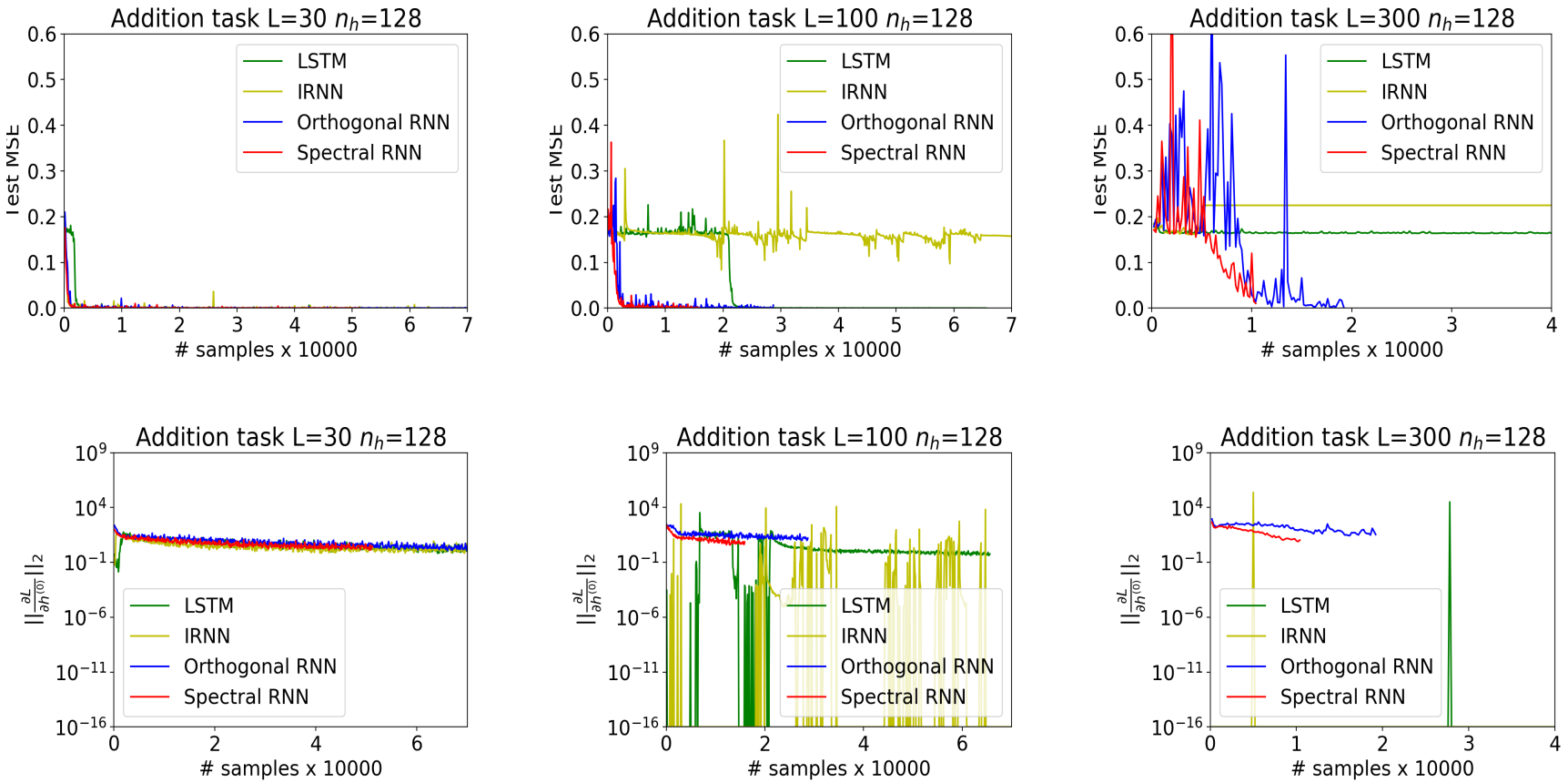


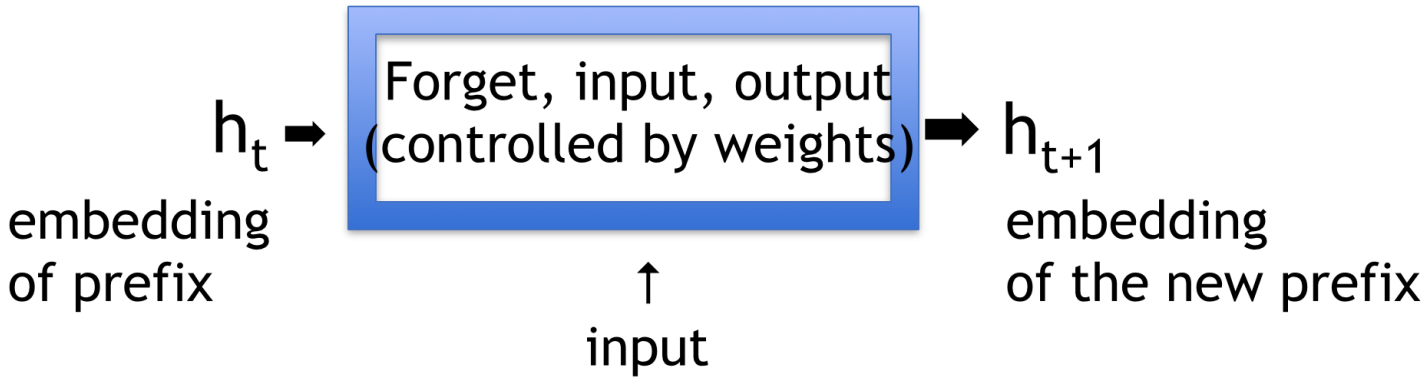
Figure: RNNs on addition task with L layers & n_h hidden dimension.

Outline

1. Introduction
   - Applications at Amazon Search
   - Recurrent Neural Networks
   - The Vanishing Gradient Problem
2. Proposed Solution
   - Spectral RNN
   - Training algorithms
   - Extensions
3. Generalization Analysis
4. Experimental Results
   - Synthetic Addition Task
   - Auto-complete Task

Auto-Complete using RNNs

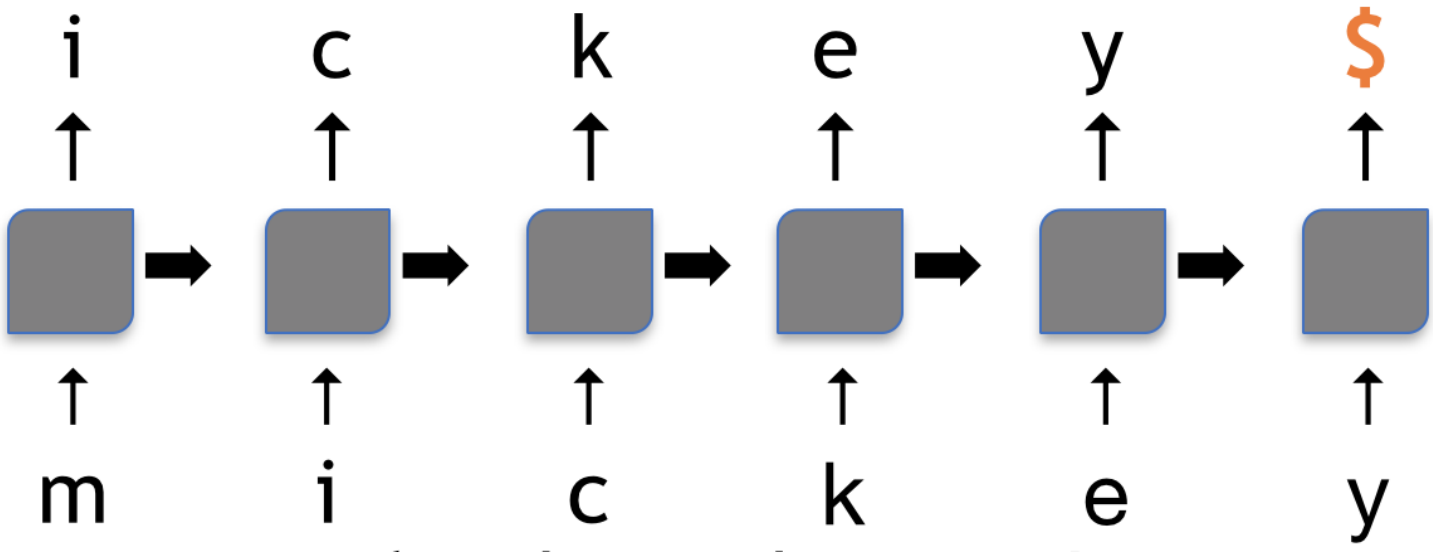
- $P(\text{prefix} \rightarrow \text{query}) = \Pi \;\; P(\text{next character}|\text{current prefix})$
- All we need to know is $P(\text{next character}|\text{current prefix})$!
- Long short term memory(LSTM) unit



- $P(\text{next character}|\text{current prefix}) = \text{softmax}\,(Wh_{t+1}) \in \mathbb{R}^{200}$

Training the model

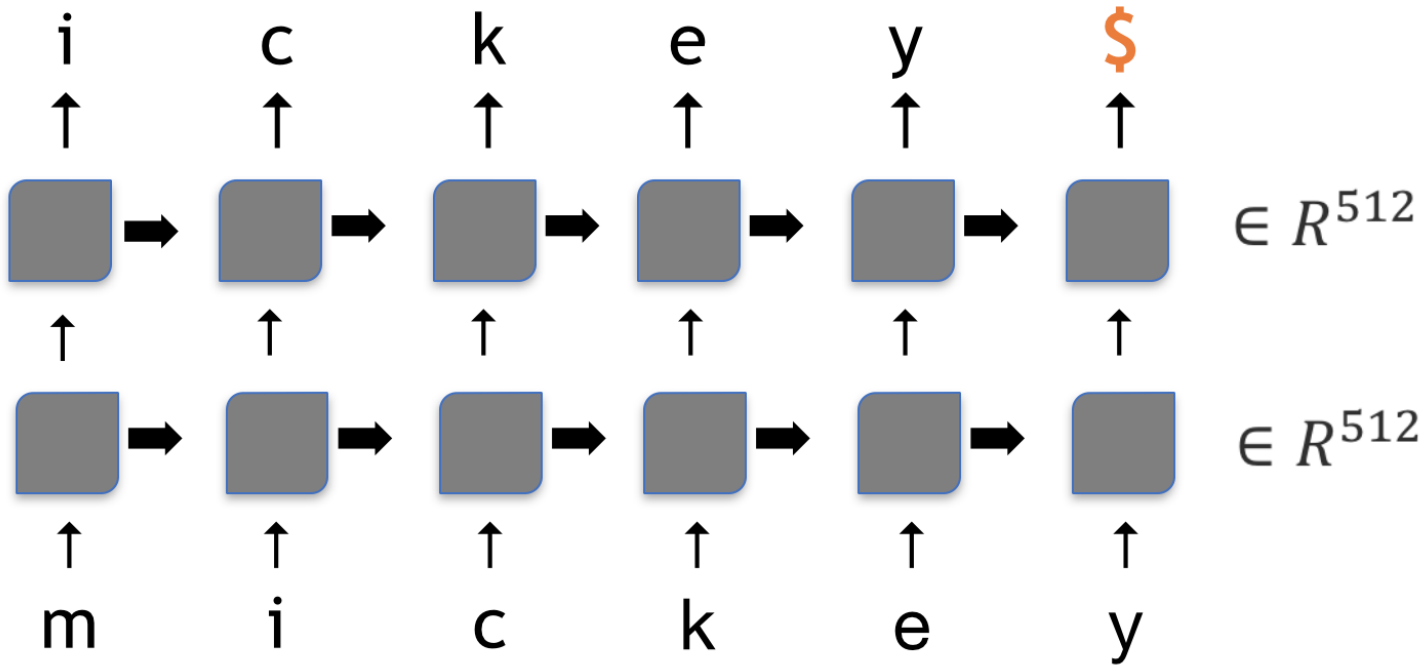
- $P(\text{prefix} \rightarrow \text{query}) = \Pi \;\; P(\text{next character}|\text{current prefix})$
- Goal: model on $P(\text{next character}|\text{prefix})$
- Train to predict the next character for every query



- Totally unsupervised, only need queries...
- Learn the language model of queries from data

Neural Network Architecture for Auto-complete

- Output: Dense **Softmax** $\mathbb{R}^{512} \rightarrow \mathbb{R}^{200}$



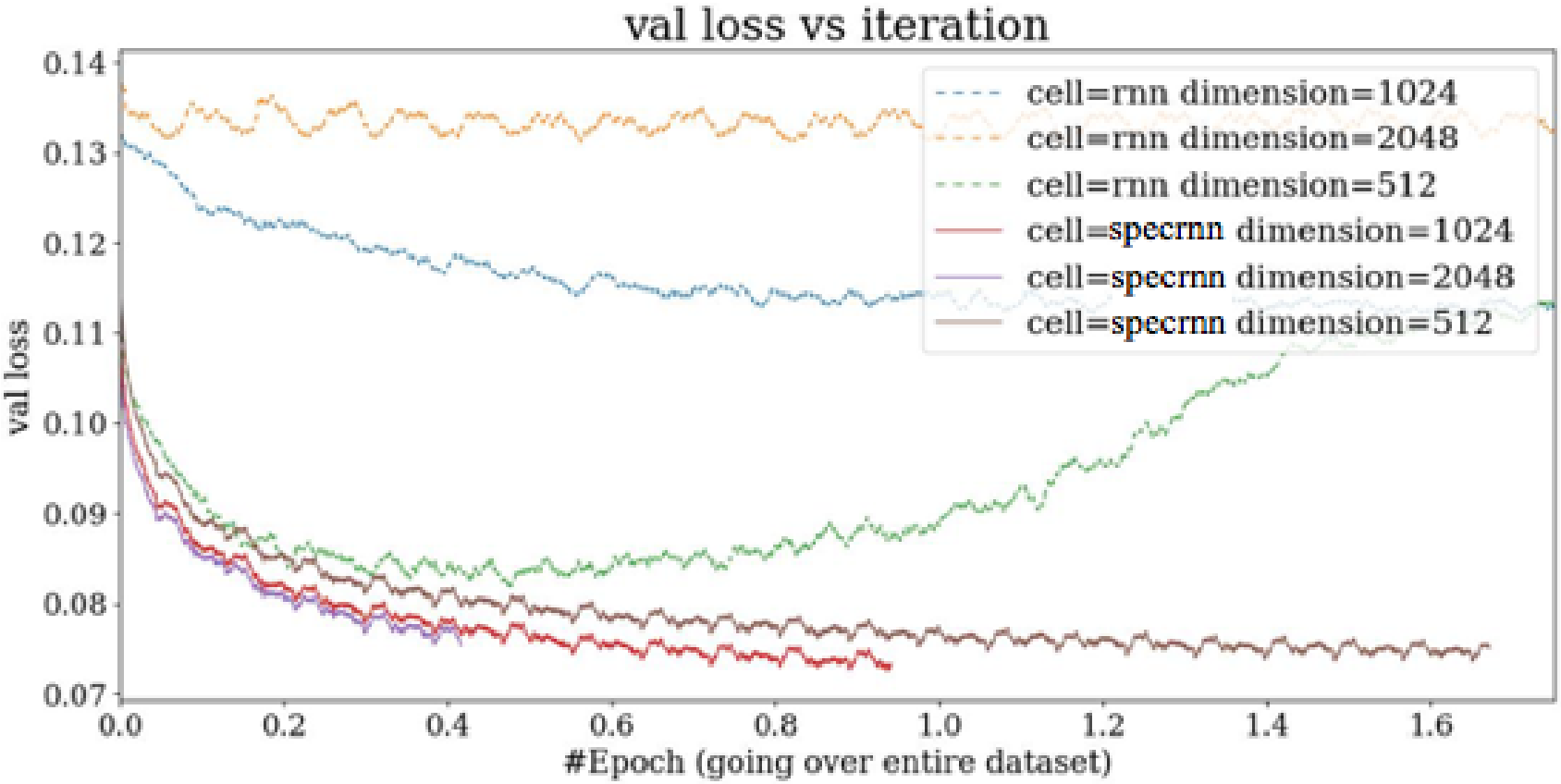
- Input: one-hot encoding $\mathbb{R}^{200}$
- 2 layers for character-level and word-level transition

Spectral RNN Training Time

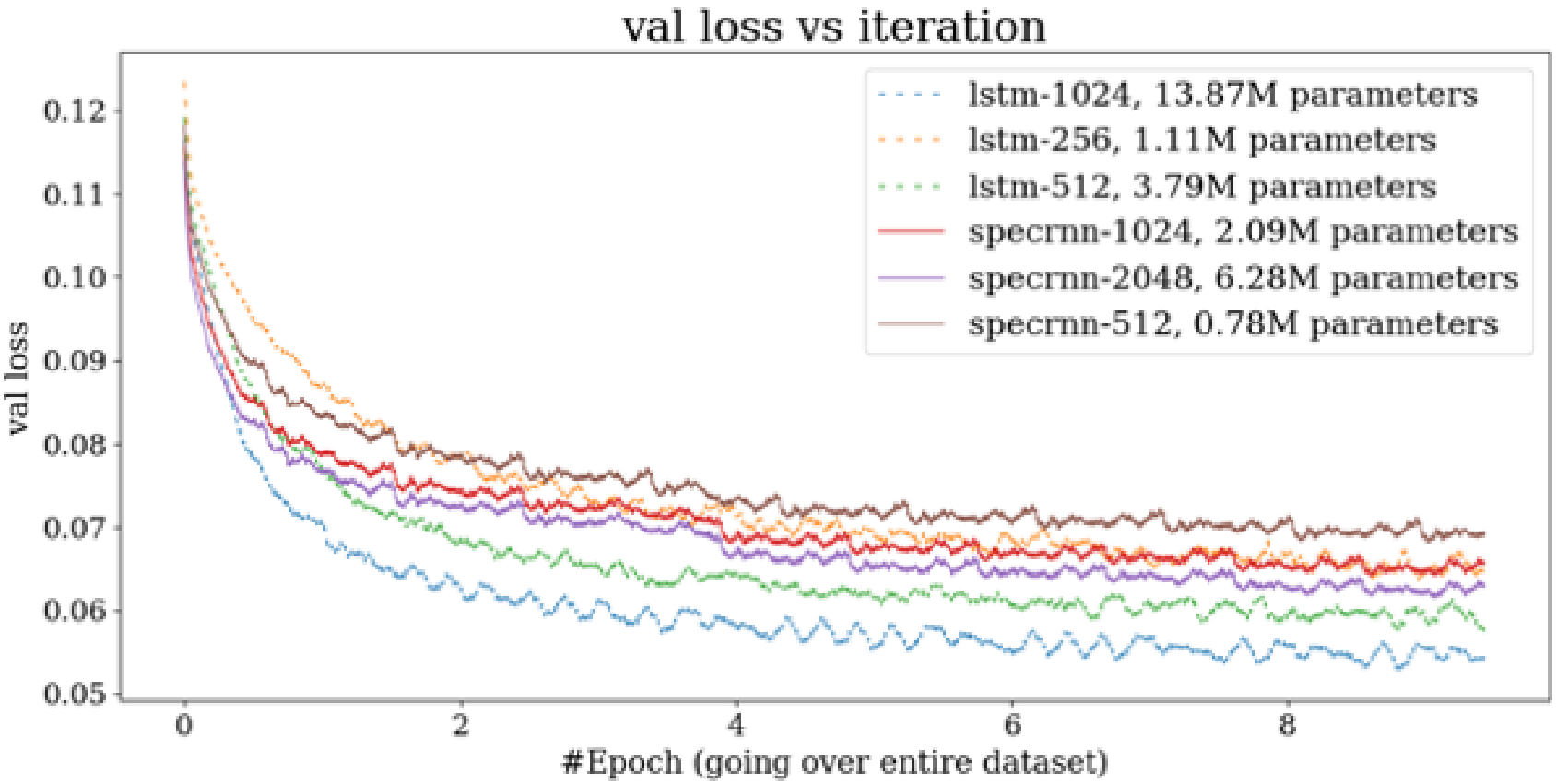
- Time Measure on V100 (P3 instance), batch size 1024, training time for 10 batches:
- Spectral RNN (0.125*dimension reflectors):
  - Dimension = 512: 9.0 s (slow!)
  - Dimension = 1024: 5.7 s
  - Dimension = 2048: 14.6 s (fast)
- LSTM (batch size 1024):
  - Dimension = 512: 2.3 s
  - Dimension = 1024: 5.7 s
  - Dimension = 2048: 25.5 s

Spectral RNN vs RNN

- Vanilla RNN diverges badly, while Spectral RNN shows convergence

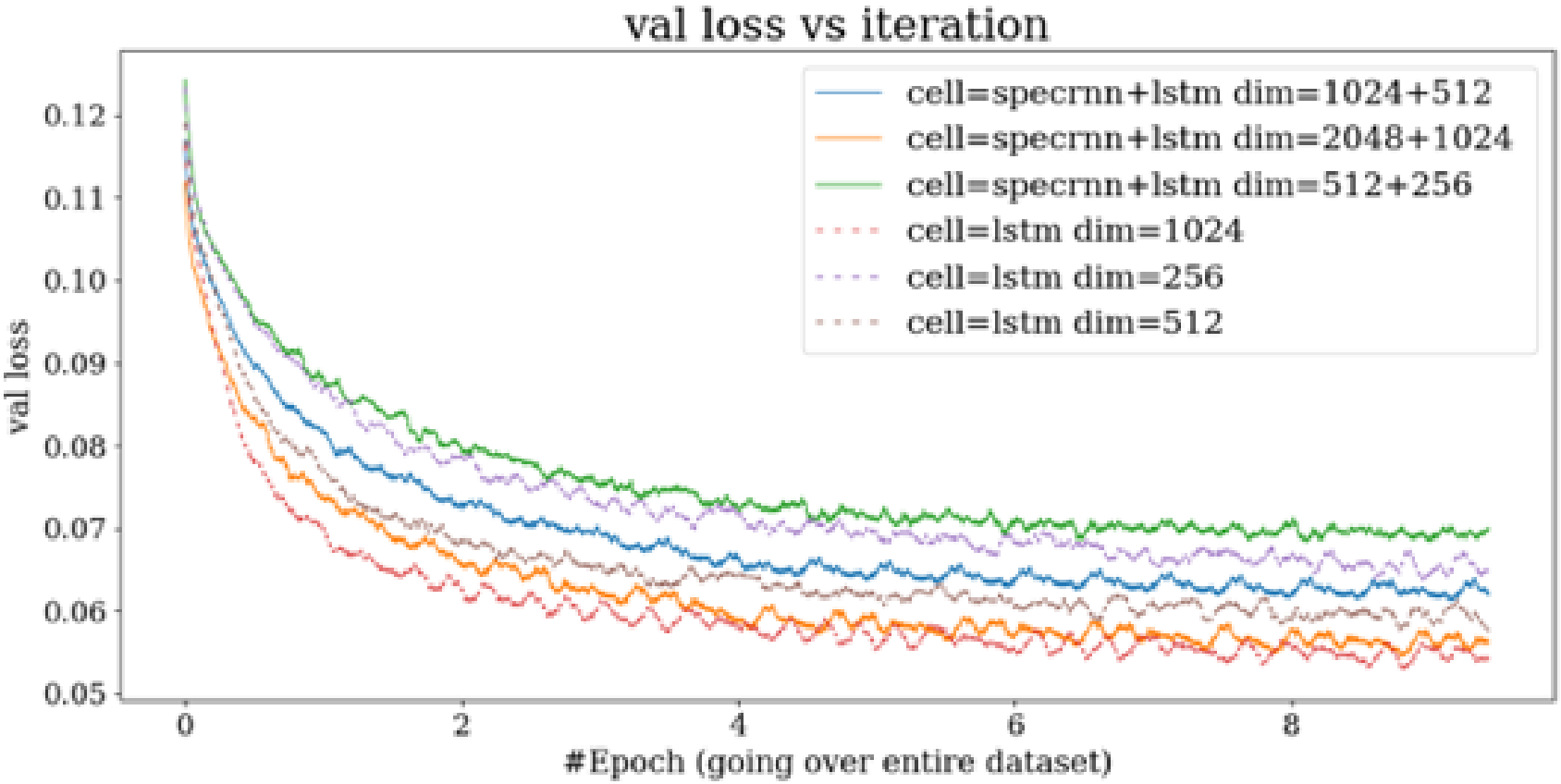


Spectral RNN vs LSTM



Spectral RNN + LSTM vs LSTM

- Number of parameters in 2 layers of LSTM and SpecRNN+LSTM are very similar
- SpecRNN+LSTM 2048+1024 comparable to 2-layer LSTM-1024 (both have 13.9 M parameters)



Conclusions

Efficient spectral parameterization of weight matrices in deep networks that:

- allows explicit control over its singular values to eliminate/reduce the exploding/vanishing gradient problem
- no loss of expressive power
- similar time complexity as vanilla RNN
- seems to have better generalization and is easier to train

Promising direction, but lots of work to be done....

References

[1] J. Zhang, Q. Lei, I. S. Dhillon. *Stabilizing Gradients for Deep Neural Networks via Efficient SVD Parametrization*. arXiv preprint arXiv:1803.09327, 2018.

[2] R. Pascanu, T. Mikolov, and Y. Bengio. *On the difficulty of training recurrent neural networks*. In ICML (2013).

[3] D. Kingma, Diederik, and J. Ba. *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980 (2014).

[4] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, and N. Thorat,*Google's multilingual neural machine translation system: enabling zero-shot translation*. arXiv preprint arXiv:1611.04558. (2016).

[5] M. Arjovsky, A. Shah, and Y. Bengio.*Unitary evolution recurrent neural networks*. In ICML. pp. 1120-1128, (2016).

[6] S. Wisdom, T. Powers, J. Hershey, J. L. Roux, and L. Atlas. *Full-capacity unitary recurrent neural networks*. In NIPS, pp. 4880-4888, (2016).

[7] Z. Mhammedi, A. Hellicar, A. Rahman, and J. Bailey. *Efficient orthogonal parametrisation of recurrent neural networks using Householder reflections*. In ICML, pp. 2401-2409, (2017).

References

[8] E. Vorontsov, C. Trabelsi, S. Kadoury, and C. Pal. *On orthogonality and learning recurrent networks with long term dependencies*. In ICML, pp. 3570-3578, (2017).

[9] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. *Parseval networks: Improving robustness to adversarial examples*. In ICML, pp. 854-863, (2017).

[10] P. Bartlett, D. J. Foster, and M. Telgarsky. *Spectrally-normalized margin bounds for neural networks*. arXiv preprint arXiv:1706.08498, (2017).

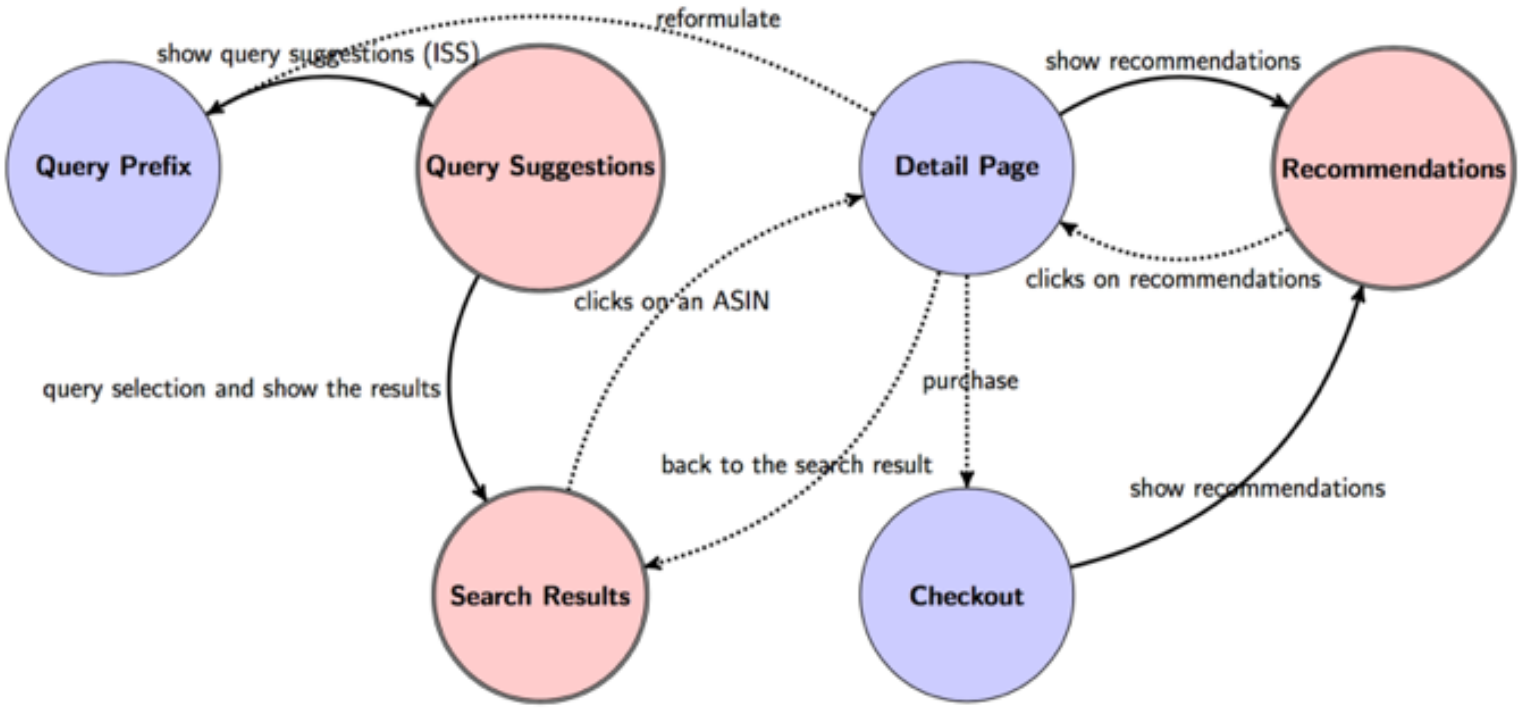
[11] Y. Prabhu and M. Varma. *FastXML: a fast, accurate and stable tree-classifier for extreme multi-label learning*. In KDD, pages 263-272, (2014).

[12] Q. V. Le, N. Jaitly, and G. E. Hinton. *A simple way to initialize recurrent networks of rectified linear units*. arXiv preprint arXiv:1504.00941, (2015).

[13] D. McAllester. *Simplified PAC-Bayesian margin bounds. In Learning theory and Kernel machines*, pp. 203215. Springer, 2003.

[14] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. *A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks.* arXiv preprint arXiv:1707.09564, 2017.

Customer Interactions on Amazon.com



Join us in solving these problems.
Internships, Visitor & Fulltime positions available.