CENG 783

Deep Learning

Week – 10
Convolutional Neural Networks (cont.)
Recurrent Neural Networks

Sinan Kalkan
Today

- Finalize CNN Applications & CNN
  - Style transfer
- Recurrent Neural Networks
Takes 19 layer VGG as the base (no FC layers)

Max pooling is replaced by avg pooling since it produced more appealing results
Figure 1: Convolutional Neural Network (CNN). A given input image is represented as a set of filtered images at each processing stage in the CNN. While the number of different filters increases along the processing hierarchy, the size of the filtered images is reduced by some downsampling mechanism (e.g., max-pooling) leading to a decrease in the total number of units per layer of the network. **Content Reconstructions.** We can visualise the information at different processing stages in the CNN by reconstructing the input image from only knowing the network’s responses in a particular layer. We reconstruct the input image from layers ′conv1_1′ (a), ′conv2_1′ (b), ′conv3_1′ (c), ′conv4_1′ (d) and ′conv5_1′ (e) of the original VGG-Network. We find that reconstruction from lower layers is almost perfect (a,b,c). In higher layers of the network, detailed pixel information is lost while the high-level content of the image is preserved (d,e). **Style Reconstructions.** On top of the original CNN representations we build a new feature space that captures the style of an input image. The style representation computes correlations between the different features in different layers of the CNN. We reconstruct the style of the input image from style representations built on different subsets of CNN layers (′conv1_1′ (a), ′conv1_1′ and ′conv2_1′ (b), ′conv1_1′, ′conv2_1′ and ′conv3_1′ (c), ′conv1_1′, ′conv2_1′, ′conv3_1′ and ′conv4_1′ (d), ′conv1_1′, ′conv2_1′, ′conv3_1′, ′conv4_1′ and ′conv5_1′ (e)). This creates images that match the style of a given image on an increasing scale while discarding information of the global arrangement of the scene.
Content reconstruction: gradient descent on a white-noise image to find an image that matches the filter responses. So let $\vec{p}$ and $\vec{x}$ be the original image and the image that is generated and $P^l$ and $F^l$ their respective feature representation in layer $l$. We then define the squared-error loss between the two feature representations

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F^l_{ij} - P^l_{ij})^2 .$$

(1)

Style representation:

Gram matrix $G^l \in \mathcal{R}^{N_l \times N_l}$, where $G^l_{ij}$ is the inner product between the vectorised feature map $i$ and $j$ in layer $l$:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} .$$

(3)
Style reconstruction:

To generate a texture that matches the style of a given image (Fig 1, style reconstructions), we use gradient descent from a white noise image to find another image that matches the style representation of the original image. This is done by minimising the mean-squared distance between the entries of the Gram matrix from the original image and the Gram matrix of the image to be generated. So let $\tilde{a}$ and $\tilde{x}$ be the original image and the image that is generated and $A^l$ and $G^l$ their respective style representations in layer $l$. The contribution of that layer to the total loss is then

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{ij} - A^l_{ij})^2$$  \hspace{1cm} (4)$$

and the total loss is

$$\mathcal{L}_{style}(\tilde{a}, \tilde{x}) = \sum_{l=0}^{L} w_l E_l$$  \hspace{1cm} (5)$$

where $w_l$ are weighting factors of the contribution of each layer to the total loss (see below for specific values of $w_l$ in our results).
Overall loss:

\[ \mathcal{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{\text{content}}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) \]
Figure 3: Detailed results for the style of the painting *Composition VII* by Wassily Kandinsky. The rows show the result of matching the style representation of increasing subsets of the CNN layers (see Methods). We find that the local image structures captured by the style representation increase in size and complexity when including style features from higher layers of the network. This can be explained by the increasing receptive field sizes and feature complexity along the network’s processing hierarchy. The columns show different relative weightings between the content and style reconstruction. The number above each column indicates the ratio $\alpha/\beta$ between the emphasis on matching the content of the photograph and the style of the artwork (see Methods).
Fully Convolutional Networks for Semantic Segmentation

Jonathan Long*  Evan Shelhamer*  Trevor Darrell
UC Berkeley
{jonlong,shelhamer,trevor}@cs.berkeley.edu
Industry Deployment

- Used in Facebook, Google, Microsoft
- Image Recognition, Speech Recognition, ....
- Fast at test time

Taigman et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification, CVPR’14
Other Applications

- Tracking (Bazzani et. al. 2010, and many others)

- Pose estimation (Toshev et al. 2013, Jain et al., 2013, ...)

- Caption generation (Vinyals et al. 2015, Xu et al. 2015, ...)

A woman is throwing a frisbee in a park. A dog is standing on a hardwood floor. A stop sign is on a road with a mountain in the background.
Convoultional networks for music recommendation

Image from: http://benanne.github.io/2014/08/05/spotify-cnns.html
To wrap up
CNNs: summary & future directions

• Less parameters

• Allows going deeper

• High flexibility
  • In operations
  • In organization of layers
  • In the overall architecture etc.

• Future directions:
  • Understanding them better
  • Making them deeper, faster and more efficient
  • Getting a trained expensive network and shrink it into a smaller cheaper one.
  • ...

Binary networks

XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks

Mohammad Rastegari†, Vicente Ordonez†, Joseph Redmon*, Ali Farhadi†*

Allen Institute for AI†, University of Washington*
{mohammadr, vicenteor}@allenai.org
{pjreddie, ali}@cs.washington.edu

Abstract. We propose two efficient approximations to standard convolutional neural networks: Binary-Weight-Networks and XNOR-Networks. In Binary-Weight-Networks, the filters are approximated with binary values resulting in $32 \times$ memory saving. In XNOR-Networks, both the filters and the input to convolutional layers are binary. XNOR-Networks approximate convolutions using primarily binary operations. This results in $58 \times$ faster convolutional operations and $32 \times$ memory savings. XNOR-Nets offer the possibility of running state-of-the-art networks on CPUs (rather than GPUs) in real-time. Our binary networks are simple, accurate, efficient, and work on challenging visual tasks. We evaluate our approach on the ImageNet classification task. The classification accuracy with a Binary-Weight-Network version of AlexNet is only 2.9% less than the full-precision AlexNet (in top-1 measure). We compare our method with recent network binarization methods, BinaryConnect and BinaryNets, and outperform these methods by large margins on ImageNet, more than 16% in top-1 accuracy.
Binary networks

![Diagram of binary networks](image)

**Network Variations**

<table>
<thead>
<tr>
<th>Network Variations</th>
<th>Operations used in Convolution</th>
<th>Memory Saving (Inference)</th>
<th>Time Saving on CPU (Inference)</th>
<th>Accuracy on ImageNet (AlexNet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Convolution</td>
<td>$+, -, \times$</td>
<td>1x</td>
<td>1x</td>
<td>%56.7</td>
</tr>
<tr>
<td>Binary Weight</td>
<td>$+, -$</td>
<td>$\sim$32x</td>
<td>$\sim$2x</td>
<td>%53.8</td>
</tr>
<tr>
<td>XNOR-Net</td>
<td>XNOR, bitcount</td>
<td>$\sim$32x</td>
<td>$\sim$58x</td>
<td>%44.2</td>
</tr>
</tbody>
</table>

**Fig. 1:** We propose two efficient variations of convolutional neural networks. **Binary-Weight-Networks**, when the weight filters contain binary values. **XNOR-Networks**, when both weights and input have binary values. These networks are very efficient in terms of memory and computation, while being very accurate in natural image classification. This offers the possibility of using accurate vision techniques in portable devices with limited resources.
CENG 783

Special topics in Deep Learning

Recurrent Neural Networks
Sequence Labeling/Modeling: Motivation
Why do we need them?

Foreign Minister. → FOREIGN MINISTER.

THE SOUND OF
Different types of sequence learning/recognition problems

- **Sequence Classification**
  - A sequence to a label
  - E.g., recognizing a single spoken word
  - Length of the sequence is fixed
  - Why RNNs then? Because sequential modeling provides robustness against translations and distortions.

- **Segment Classification**
  - Segments in a sequence correspond to labels

- **Temporal Classification**
  - General case: sequence (input) to sequence (label) modeling.
  - No clue about where input or label starts.

A. Graves, “Supervised Sequence Labelling with Recurrent Neural Networks”, 2012.
Recurrent Neural Networks
Recurrence Neural Networks (RNNs)

- RNNs are very powerful because:
  - Distributed hidden state that allows them to store a lot of information about the past efficiently.
  - Non-linear dynamics that allows them to update their hidden state in complicated ways.
- With enough neurons and time, RNNs can compute anything that can be computed by your computer.
- More formally, RNNs are Turing complete.

Adapted from Hinton
Recurrent Neural Nets

- Temporal pattern recognition

- Sequence generation

- Pattern completion / constraint satisfaction

Slide: Michael Mozer
Some examples

Jordan Networks

Elman Networks

“context” neurons

Figs: David Kriesel
Challenge

• Back propagation is designed for feedforward nets
• What would it mean to back propagate through a recurrent network?
  • error signal would have to travel back in time
Unfolding

Feed-forward networks

Recurrent networks

Unfolding
Unfolding (another example)

Figure: Michael Mozer
one to one

one to many

many to one

many to many

many to many

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
How an RNN works

projections (activities x weights)

activities (vectors of values)

Learned representation of sequence.

the cat sat on the mat

hidden to hidden input to hidden

Alec Radford
You can stack them too

```
the  cat  sat  on  the  mat
```

hidden to output
hidden to hidden
input to hidden

Alec Radford
Unfolding implications

- Entails duplication of weights => weight sharing
- Sharing weights means their gradients will be accumulated over time and reflected on the weights
- Unfolded network has the same dynamics of the RNN for a fixed number of time steps!
Back-propagation Through Time
Reminder: Backpropagation with weight constraints

- It is easy to modify the backprop algorithm to incorporate linear constraints between the weights.
- We compute the gradients as usual, and then modify the gradients so that they satisfy the constraints.
  - So if the weights started off satisfying the constraints, they will continue to satisfy them.

To constrain: \( w_1 = w_2 \)

we need: \( \Delta w_1 = \Delta w_2 \)

compute: \( \frac{\partial E}{\partial w_1} \) and \( \frac{\partial E}{\partial w_2} \)

use \( \frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial w_2} \) for \( w_1 \) and \( w_2 \)
Backpropagation through time

• We can think of the recurrent net as a layered, feed-forward net with shared weights and then train the feed-forward net with weight constraints.

• We can also think of this training algorithm in the time domain:
  • The forward pass builds up a stack of the activities of all the units at each time step.
  • The backward pass peels activities off the stack to compute the error derivatives at each time step.
  • After the backward pass we add together the derivatives at all the different times for each weight.
Cross-entropy loss:
\[ E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t \]
\[ E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t) = -\sum_t y_t \log \hat{y}_t \]

Accumulate errors over time (treat the whole sequence as a training example):
\[ \frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \]

\[
\begin{align*}
\frac{\partial E_3}{\partial V} &= \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial V} \\
&= \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial z_3} \frac{\partial z_3}{\partial V} \\
&= (\hat{y}_3 - y_3) \otimes s_3 \\
z_3 &= V s_3
\end{align*}
\]

\[
\begin{align*}
\frac{\partial E_3}{\partial W} &= \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W} \\
&= \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}
\end{align*}
\]

\[
\frac{\partial C_t}{\partial W} = \sum_{t'=1}^{t} \frac{\partial C_t}{\partial h_t} \frac{\partial h_t}{\partial h_{t'}} \frac{\partial h_{t'}}{\partial W}, \quad \text{where} \quad \frac{\partial h_t}{\partial h_{t'}} = \prod_{k=t'+1}^{t} \frac{\partial h_k}{\partial h_{k-1}}
\]
An irritating extra issue

- We need to specify the initial activity state of all the hidden and output units.
- We could just fix these initial states to have some default value like 0.5.
- But it is better to treat the initial states as learned parameters.
- We learn them in the same way as we learn the weights.
  - Start off with an initial random guess for the initial states.
  - At the end of each training sequence, backpropagate through time all the way to the initial states to get the gradient of the error function with respect to each initial state.
  - Adjust the initial states by following the negative gradient.
A good toy problem for a recurrent network

- We can train a feedforward net to do binary addition, but there are obvious regularities that it cannot capture efficiently.
  - We must decide in advance the maximum number of digits in each number.
  - The processing applied to the beginning of a long number does not generalize to the end of the long number because it uses different weights.

- As a result, feedforward nets do not generalize well on the binary addition task.
The algorithm for binary addition

This is a finite state automaton. It decides what transition to make by looking at the next column. It prints after making the transition. It moves from right to left over the two input numbers.
A recurrent net for binary addition

- The network has two input units and one output unit.
- It is given two input digits at each time step.
- The desired output at each time step is the output for the column that was provided as input two time steps ago.
  - It takes one time step to update the hidden units based on the two input digits.
  - It takes another time step for the hidden units to cause the output.

\[
\begin{array}{cccc}
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
\hline
0 & 1 & 0 & 0 \end{array}
\]

\[
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
\hline
1 & 0 & 0 & 0 \end{array}
\]

Slide: Hinton
Sum of three numbers

Addition and subtraction in the same network

Generalized network: Any length, any operation

https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-sequence-learning/
The problem of exploding or vanishing gradients

- What happens to the magnitude of the gradients as we backpropagate through many layers?
  - If the weights are small, the gradients shrink exponentially.
  - If the weights are big the gradients grow exponentially.

- Typical feed-forward neural nets can cope with these exponential effects because they only have a few hidden layers.

- In an RNN trained on long sequences (e.g. 100 time steps) the gradients can easily explode or vanish.
  - We can avoid this by initializing the weights very carefully.

- Even with good initial weights, it's very hard to detect that the current target output depends on an input from many time-steps ago.
  - So RNNs have difficulty dealing with long-range dependencies.
BPTT vs RTRL

One complaint about BPTT is that it requires storing activity states of all units at all previous times.

Space complexity of BPTT: \( O(NT) \)
Time complexity of BPTT: \( O(N^3T) \)

Williams & Zipser have developed an alternative algorithm called real-time recurrent learning (RTRL) which does not require storing activity states.

Space complexity of RTRL: \( O(N^3) \)
Time complexity of RTRL: \( O(N^4) \)

Both RTRL and BPTT compute exact gradient of the error with respect to the weights. They have the same power and limitations.

For large \( T \), RTRL may be more efficient. Not particularly useful in practice.
Problem With BPTT and RTRL

while BPTT is in principle capable of learning relationships among temporal events, in practice it is weak.

E.g., detecting contingencies spanning temporal gaps 

\[ e_1 \ldots e_k \]

\[ e_2 \text{ dependent on } e_1 \]

Input is a sequence of symbols: ABCDEFXY
Task is to predict next symbol in sequence

Sample sequences:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>XAXBX</td>
<td>XABDEFX</td>
<td></td>
</tr>
<tr>
<td>YABY</td>
<td>YABCDEFY</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{gap} = 2 \]
\[ \text{gap} = 6 \]

Learning two-sequence training set with a sequence-recog. architecture and BPTT is not reliable for gaps of 4 or more

<table>
<thead>
<tr>
<th>gap</th>
<th>% failures after 100 epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>92</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
</tr>
</tbody>
</table>

(Mozer, 1992)

Problem: BPTT ok at discovering structure that is local in time, but not good at handling structure at a more global scale (long temporal intervals, & involving high order statistics).
Exploding and vanishing gradients problem

- **Solution 1**: Gradient clipping for exploding gradients:

  **Algorithm 1** Pseudo-code for norm clipping
  
  \[ \hat{g} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta} \]
  
  if \( \| \hat{g} \| \geq \text{threshold} \)
  
  \[ \hat{g} \leftarrow \frac{\text{threshold}}{\| \hat{g} \|} \hat{g} \]
  
  end if

- For vanishing gradients: Regularization term that penalizes changes in the magnitudes of back-propagated gradients

\[
\Omega = \sum_k \Omega_k = \sum_k \left( \left\| \frac{\partial \mathcal{E}}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial x_k} \right\| - 1 \right)^2
\]

**Figure 6.** We plot the error surface of a single hidden unit recurrent network, highlighting the existence of high curvature walls. The solid lines depicts standard trajectories that gradient descent might follow. Using dashed arrow the diagram shows what would happen if the gradients is rescaled to a fixed size when its norm is above a threshold.
Exploding and vanishing gradients problem

- Solution 2:
  - Use methods like LSTM
Long Short Term Memory (LSTM) Networks
RNN

- Basic block diagram

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
Key Problem

• Learning long-term dependencies is hard
Long Short Term Memory (LSTM)

- Hochreiter & Schmidhuber (1997) solved the problem of getting an RNN to remember things for a long time (like hundreds of time steps).
- They designed a memory cell using logistic and linear units with multiplicative interactions.
- Information gets into the cell whenever its “write” gate is on.
- The information stays in the cell so long as its “keep” gate is on.
- Information can be read from the cell by turning on its “read” gate.
Meet LSTMs

• How about we explicitly encode memory?
LSTM in detail

- We first compute an activation vector, $a$:
  \[ a = W_{x}x_t + W_{h}h_{t-1} + b \]
- Split this into four vectors of the same size:
  \[ a_i, a_f, a_o, a_g \leftarrow a \]
- We then compute the values of the gates:
  \[ i = \sigma(a_i) \quad f = \sigma(a_f) \quad o = \sigma(a_o) \quad g = \tanh(a_g) \]
  where $\sigma$ is the sigmoid.
- The next cell state $c_t$ and the hidden state $h_t$:
  \[ c_t = f \odot c_{t-1} + i \odot g \quad h_t = o \odot \tanh(c_t) \]
  where $\odot$ is the element-wise product of vectors

Eqs: Karpathy
LSTMs Intuition: Memory

Cell State / Memory

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
LSTMs Intuition: Forget Gate

- Should we continue to remember this “bit” of information or not?

\[ f_t = \sigma \left( W_f \cdot [h_{t-1}, x_t] + b_f \right) \]
LSTMs Intuition: Input Gate

• Should we update this “bit” of information or not?
  • If so, with what?

\[
\begin{align*}
i_t &= \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)
\end{align*}
\]
LSTMs Intuition: Memory Update

- Forget that + memorize this

\[
C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t
\]
LSTMs Intuition: Output Gate

- Should we output this “bit” of information to “deeper” layers?

\[ o_t = \sigma (W_o [h_{t-1}, x_t] + b_o) \]
\[ h_t = o_t \times \text{tanh} (C_t) \]

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
LSTMs

• A pretty sophisticated cell

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
LSTM Variants #1: Peephole Connections

• Let gates see the cell state / memory

\[
\begin{align*}
    f_t &= \sigma \left( W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f \right) \\
    i_t &= \sigma \left( W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i \right) \\
    o_t &= \sigma \left( W_o \cdot [C_t, h_{t-1}, x_t] + b_o \right)
\end{align*}
\]
LSTM Variants #2: Coupled Gates

- Only memorize new if forgetting old

\[
C_t = f_t \cdot C_{t-1} + (1 - f_t) \cdot \tilde{C}_t
\]
LSTM Variants #3: Gated Recurrent Units

- Changes:
  - No explicit memory; memory = hidden output
  - \( Z = \) memorize new and forget old

\[
\begin{align*}
  z_t &= \sigma (W_z \cdot [h_{t-1}, x_t]) \\
  r_t &= \sigma (W_r \cdot [h_{t-1}, x_t]) \\
  \tilde{h}_t &= \tanh (W \cdot [r_t \ast h_{t-1}, x_t]) \\
  h_t &= (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t
\end{align*}
\]

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)
References

• A very detailed explanation with nice figures

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
CNN vs RNN

Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An 1x1 convolution is added when residual input and output have different dimensions. (c) An example of residual connection in a TCN. The blue lines are filters in the residual function, and the green lines are identity mappings.
Example: Character-level Text Modeling
Modeling text: Advantages of working with characters

- The web is composed of character strings.
- Any learning method powerful enough to understand the world by reading the web ought to find it trivial to learn which strings make words (this turns out to be true, as we shall see).
- Pre-processing text to get words is a big hassle
  - What about morphemes (prefixes, suffixes etc)
  - What about subtle effects like “sn” words?
  - What about New York?
  - What about Finnish
    - ymmartämätomyydellänsakaän
A sub-tree in the tree of all character strings

There are exponentially many nodes in the tree of all character strings of length N.

In an RNN, each node is a hidden state vector. The next character must transform this to a new node.

- If the nodes are implemented as hidden states in an RNN, different nodes can share structure because they use distributed representations.
- The next hidden representation needs to depend on the conjunction of the current character and the current hidden representation.
An obvious recurrent neural net

1500 hidden units

character: 1-of-86

predicted distribution for next character.

It’s a lot easier to predict 86 characters than 100,000 words.
A simple scenario

- Alphabet: h, e, l, o
- Text to train to predict: “hello”
Sample predictions (when trained on the works of Shakespeare):

- 3-level RNN with 512 hidden nodes in each layer

PANDARUS:
Alas, I think he shall be come approached and the day
When little strain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Sample predictions (when trained on Wikipedia):

• Using LSTM

Naturalism and decision for the majority of Arab countries' capital Vide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict. Copyright was the succession of independence in the slop of Syrian influence that was a famous German movement based on a more popular servicious, non-doctrinal and sexual power post. Many governments recognize the military housing of the [[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab Resolution]] (PJS)[http://www.humah.yahoo.com/guardian.cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major tripad of aid exile.]]
Sample predictions (when trained on **Latex documents**):

- Using multi-layer LSTM

---

**Lemma 0.1.** Assume (3) and (9) by the construction in the description. Suppose \( X = \lim |X| \) be the formal open covering \( X \) and a single map \( \text{Proj}_X(A) = \text{Spec}(B) \) over \( U \) compatible with the complex

\[
\text{Set}(A) = \Gamma(X, \mathcal{O}_X, \mathcal{O}_X).
\]

When in this case of to show that \( Q \to \mathcal{C}_{\mathcal{U}/X} \) is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If \( T \) is surjective we may assume that \( T \) is connected with residue fields of \( S \). Moreover there exists a closed subspace \( Z \subset X \) of \( X \) where \( U \) in \( X' \) is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

\( X' \) is locally of finite type. Since \( S = \text{Spec}(R) \) and \( Y = \text{Spec}(R) \).

**Proof.** This is form all sheaves of sheaves on \( X \). But given a scheme \( U \) and a surjective étale morphism \( U \to X \). Let \( U \cap U = \bigcap_{i=1}^{n} U_i \) be the scheme \( X \) over \( S \) at the schemes \( X_i \to X \) and \( U = \lim_i X_i \).

The following lemma surjective restrocomposes of this implies that \( \mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x_0} \).

**Lemma 0.2.** Let \( X \) be a locally Noetherian scheme over \( S \), \( E = \mathcal{F}_{X/S} \). Set \( I = \mathcal{I}_1 \subset \mathcal{I}_2 \). Since \( I^n \subset \mathcal{I}_n \) are nonzero over \( I \leq \mathcal{I} \) is a subset of \( \mathcal{I}_n \) works.

**Lemma 0.3.** In Situation ???. Hence we may assume \( q' = 0 \).

**Proof.** We will use the property we see that \( p \) is the next functor (??). On the other hand, by ?? we see that

\[
D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)
\]

where \( K \) is an \( F \)-algebra where \( \delta_{n+1} \) is a scheme over \( S \).
He was elected President during the Revolutionary War and forgave Opus Paul at Rome. The regime of his crew of England, is now Arab women's icons in and the demons that use something between the characters‘ sisters in lower coil trains were always operated on the line of the ephemeral street, respectively, the graphic or other facility for deformation of a given proportion of large segments at RTUS). The B every chord was a "strongly cold internal palette pour even the white blade."
Some completions produced by the model

- Sheila thrunges (most frequent)
- People thrungrg (most frequent next character is space)
- Shiela, Thrungelini del Rey (first try)
- The meaning of life is literary recognition. (6th try)

- The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger. (one of the first 10 tries for a model trained for longer).
What does it know?

- It knows a huge number of words and a lot about proper names, dates, and numbers.

- It is good at balancing quotes and brackets.
  - It can count brackets: none, one, many

- It knows a lot about syntax but its very hard to pin down exactly what form this knowledge has.
  - Its syntactic knowledge is not modular.

- It knows a lot of weak semantic associations
  - E.g. it knows Plato is associated with Wittgenstein and cabbage is associated with vegetable.
RNNs for predicting the next word

- Tomas Mikolov and his collaborators have trained quite large RNNs on quite large training sets using BPTT.
  - They do better than feed-forward neural nets.
  - They do better than the best other models.
  - They do even better when averaged with other models.

- RNNs require much less training data to reach the same level of performance as other models.

- RNNs improve faster than other methods as the dataset gets bigger.
  - This is going to make them very hard to beat.
Word-level RNN for news title generation

https://github.com/larspars/word-rnn

Click-o-Tron

3D Video Brings Clean Energy To The Real Economy

David Beckham & Victoria Beckham’s Talk Show Gets 2014 Golden Girls Love

The area is still in such big power, but I have some advice for those this ye...

White House Attacks The Obama Plan

Women learned from their families in a post-9/11 New York Times story that is...

Not The Same Thing: New Rule Can’t

http://clickotron.com/