Recurrent and Convolutional Networks
— Week #6
Neural Network architectures

• Feed-forward neural nets are standard

• We can specialize neural networks for particular data

• Different data have different characteristics
Today

- Two specific architectures:
  A. Recurrent neural networks for sequential data*
  B. Convolutional neural networks for image data

*: presentation inspired and some slides borrowed and adapted from César Laurent's presentation at IVADO's deep learning summer school.
Sequential Data

- Not fixed length
- For example:
  - Text (document)
  - Speech
  - Stock prices
  - Sensor measurements through time
  - Videos
  - DNA
What is machine learning and data mining?

Qu'est-ce que l'apprentissage automatique et l'exploration de données?
From: bcash@crchh410.NoSubdomain.NoDomain (Brian Cash)
Subject: Re: free moral agency
Nntp-Posting-Host: crchh410
Organization: BNR, Inc.
Lines: 17

In article <735295730.25282@minster.york.ac.uk>, cjhs@minster.york.ac.uk writes:
> : Are you saying that their was a physical Adam and Eve, and that all
> : humans are direct decendents of only these two human beings.? Then who
> : were Cain and Able's wives? Couldn't be their sisters, because A&E
> : didn't have daughters. Were they non-humans?
> >
> Genesis 5:4
> >
> and the days of Adam after he begat Seth were eight hundred years, and
> he begat sons and daughters:
> >
> Felicitations -- Chris Ho-Stuart

Yeah, but these were not the wives. The wives came from Nod, apparently
a land being developed by another set of gods.

Brian /-|--\
Task classification

Object Classification

Image - Class

Task classification

- **Object Classification**
- **Image Classification**
- **Image - Class**
- **Image Captioning**
- **Image - Caption**

Task classification

- **Object Classification**
- **Image Classification**
- **Image Captioning**
- **Sentiment Analysis**
- **Text - Sentiment**

Task classification

- **Object Classification**
- **Image Captioning**
- **Sentiment Analysis**
- **Machine Translation**
Task classification

- **Object Classification**
  - Image - Class

- **Image Captioning**
  - Image - Caption

- **Sentiment Analysis**
  - Text - Sentiment

- **Machine Translation**
  - French - English

- **Anomaly Detection**
  - Sensor reading - Class

Intuition

• Imagine reading a book
  
  • You read the text word by word. Each word is added to your memory
  
  • After each page you memory contains some representation of all the words you have read so far
Basic idea

Process through time (t)

\[ h_t \]

\[ y_t \]

\[ x_t \]
Basic idea

Process through time \((t)\)
- \(U, V, W\): parameters
- Shared through time
- Simplest parametrization

\[
\begin{align*}
  h_t &= \tanh(Ux_t + Wh_{t-1}) \\
  y_t &= f(Vh_t)
\end{align*}
\]
Unroll through time
Training RNNs

- Gradient descent from the loss
  \[ E = \sum_{t} (y_t - \hat{y}_t)^2 \]

- Following the structure the gradient is back propagated through time
For example, the gradient for $U$ is

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$$
For example, the gradient for $U$

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$$

![Diagram](http://colah.github.io/)
For example, the gradient for $U$:

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$$

Diagram:

- $E_0$, $E_1$, $E_2$
- $y_0$, $y_1$, $y_2$
- $h_0$, $h_1$, $h_2$
- $x_0$, $x_1$, $x_2$

Related links:

- [http://colah.github.io/](http://colah.github.io/)
For example, the gradient for $U$:

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U}$$
Limitations

- Long-term dependencies are difficult to learn
Limitations

- Long-term dependencies are difficult to learn

Julie joined teammates 80629

Q: Who joined the class?

http://colah.github.io/
Limitations

- Long-term dependencies are difficult to learn

Q: Who joined the class?

http://colah.github.io/
Limitations

- Long-term dependencies are difficult to learn

Q: Who joined the class?

http://colah.github.io/
• Can be difficult because it is unstable

  • Each partial derivative depends on the parameters $W$

    • The largest eigenvalue of $W$:

      • $>1$: the gradient will explode

      • $<1$: the gradient will vanish
• Largest eigenvalue of $W$ is $> 1$
Gradient Clipping

- Heuristic to prevent gradient explosion
- If gradient norm is too large then reduce
• Largest eigenvalue of $W$ is $< 1$
Gradient Vanishing

- No simple solution
- Change the “memory cells” of the RNN
Different architectures of RNNs

Deep RNNs

Bi-directional RNNs

http://colah.github.io/
Generating Sequences from RNNs
The guardian of the land of an heir who is under age shall take from it only reasonable revenues, customary dues, and feudal services. He shall do this without destruction or damage to men or property. If we have given the guardianship of the land to a sheriff, or to any person answerable to us for the revenues, and he commits destruction or damage, we will exact compensation from him, and the land shall be entrusted to two worthy and prudent men of the same 'fee', who shall be answerable to us for the revenues, or to the person to whom we have assigned them. If we have given or sold to anyone the guardianship of such land, and he causes destruction or damage, he shall lose the guardianship of it, and it shall be handed over to two worthy and prudent men of the same 'fee', who shall be similarly answerable to us.
Prevalent methodology

- Frame language modelling as a prediction task
- Predict the next word given the previous word
- Intuition
  - Success at this task implies that you have an understanding of the text (at least short-term)
Training.
Given previous words, predict the next word.

Target: guardian of the land of an

Input: The guardian of the land of
Test.
1. Predict One word at a time.
2. Feed the prediction back to the model

![Diagram of neural network with input and prediction nodes.](http://colah.github.io/)
Test.
1. Predict One word at a time.
2. Feed the prediction back to the model

Prediction

Input

The

\[ h_0 \rightarrow h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow h_4 \rightarrow h_5 \]
1. Predict One word at a time.
2. Feed the prediction back to the model

Prediction: woman

Input: The
1. Predict One word at a time.
2. Feed the prediction back to the model.

Prediction: woman

Input: The woman

Diagram:

- $h_0$ to $h_1$ to $h_2$ to $h_3$ to $h_4$ to $h_5$
1. Predict One word at a time.
2. Feed the prediction back to the model.

Prediction

The woman

Input

h_0 → h_1 → h_2 → h_3 → h_4 → h_5
Test.
1. Predict One word at a time.
2. Feed the prediction back to the model

http://colah.github.io/
Test.
1. Predict One word at a time.
2. Feed the prediction back to the model
1. Predict One word at a time.
2. Feed the prediction back to the model

Test.

Prediction: woman the heir to the crown

Input: The woman the heir to the

http://colah.github.io/
Encoder-Decoder architecture

- One of the most influential recent ideas
Encoder-Decoder for translation

Encoder  Decoder

h₀  h₁  h₂  s₀  s₁  s₂  s₃

J'  aime  papa  <SOS>  I  like  dad  <EOS>

like  dad

http://colah.github.io/
RNNs takeaways (I)

- Can be used to learn from varying-length input
  - Typically for discrete data (e.g., words)
  - Can be used both for predictions and for representations
RNNs takeaways (II)

- Can easily be used as modules inside more complex systems

- Current applications: Natural language understanding (Q&A, Dialogs), machine translation

- Very active research field

- Still difficult to learn very long sequences (reading a book)

- Not available in scikit-learn
In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. “By the time we reached the top of one peak, the water looked blue, with some crystals on top,” said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them — they were so close they could touch their horns.

While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, “We can see, for example, that they have a common ‘language,’ something like a dialect or dialectic.”
The 2008 Summer Olympics torch relay was run from March 24 until August 8, 2008, prior to the 2008 Summer Olympics, with the theme of “one world, one dream”. Plans for the relay were announced on April 26, 2007, in Beijing, China. The relay, also called by the organizers as the “Journey of Harmony”, lasted 129 days and carried the torch 137,000 km (85,000 mi) – the longest distance of any Olympic torch relay since the tradition was started ahead of the 1936 Summer Olympics.

After being lit at the birthplace of the Olympic Games in Olympia, Greece on March 24, the torch traveled to the Panathinaiko Stadium in Athens, and then to Beijing, arriving on March 31. From Beijing, the torch was following a route passing through six continents. The torch has visited cities along the Silk Road, symbolizing ancient links between China and the rest of the world. The relay also included an ascent with the flame to the top of Mount Everest on the border of Nepal and Tibet, China from the Chinese side, which was closed specially for the event.

Q: What was the theme?
A: “one world, one dream”.

Q: What was the length of the race?
A: 137,000 km

Q: Was it larger than previous ones?
A: No

Q: Where did the race begin?
A: Olympia, Greece
Neural Network architectures

- Feed-forward neural nets are standard
- We can specialize neural networks for particular tasks
- Different data have different characteristics
- Convolutional nets replace matrix multiplications by convolutions and pooling
This is how I see

This is how my computer sees

Note: I took just 10*10 grid here for understanding, but the image array size = Width*Height*Color.

Source: https://medium.com/deep-math-machine-learning-ai/chapter-8-0-convolutional-neural-networks-for-deep-learning-364971e34ab2
A 100 x 100 pixel image has 10,000 dimensions

- Often have a color dimension (data is a tensor)

- Modern iPhone (12 MP): 4032 x 3024 pixels

Source: https://medium.com/deep-math-machine-learning-ai/chapter-8-0-convolutional-neural-networks-for-deep-learning-364971e34ab2
Convolutional Neural Networks
Convolutional Neural Networks

- Feed-forward networks may not scale to high-dimensional data

- Each additional input/feature adds “m” parameters

- In practice it can be hard to scale to thousands of dimensions
Convolutional Neural Networks

- Feed-forward networks may not scale to high-dimensional data
- Each additional input/feature adds “m” parameters
- In practice it can be hard to scale to thousands of dimensions

Remedies

A. Sparse Connections

B. Parameter sharing
A layer in a CNN

Figure 9.7: The components of a typical convolutional neural network layer. There are two commonly used sets of terminology for describing these layers.

(left) In this terminology, the convolutional net is viewed as a small number of relatively complex layers, with each layer having many “stages.” In this terminology, there is a one-to-one mapping between kernel tensors and network layers. In this book we generally use this terminology.

(right) In this terminology, the convolutional net is viewed as a larger number of simple layers; every step of processing is regarded as a layer in its own right. This means that not every “layer” has parameters.
Some intuitions

• You would like to recognize objects:
  • Regardless of their size (scale invariant)
  • Regardless of their position (translation invariant)
Some intuitions

• You would like to recognize objects:
  • Regardless of their size (scale invariant)
  • Regardless of their position (translation invariant)
• Have small detectors for object parts
  • Run the detectors over all regions of the image
  • Patterns of detection may indicate the presence of a full object
Convolutions
(to the rescue)

For pixel \((i, j)\):

\[
S(i, j) = (K * I)(i, j) = \sum_{m} \sum_{n} I(i + m, j + n)K(m, n)
\]

- Dot product between “the kernel and the region”
Convolutions
(to the rescue)

For pixel \((i, j)\):

\[
S(i, j) = (K \ast I)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n)
\]

- Dot product between “the kernel and the region”
CHAPTER 9. CONVOLUTIONAL NETWORKS

Figure 9.6: Efficiency of edge detection. The image on the right was formed by taking each pixel in the original image and subtracting the value of its neighboring pixel on the left. This shows the strength of all of the vertically oriented edges in the input image, which can be a useful operation for object detection. Both images are 280 pixels tall. The input image is 320 pixels wide while the output image is 319 pixels wide. This transformation can be described by a convolution kernel containing two elements, and requires \( 319 \times 280 \times 3 = 267,960 \) floating point operations (two multiplications and one addition per output pixel) to compute using convolution. To describe the same transformation with a matrix multiplication would take \( 320 \times 280 \times 319 \times 280 \), over eight billion, entries in the matrix, making convolution four billion times more efficient for representing this transformation. The straightforward matrix multiplication algorithm performs over sixteen billion floating point operations, making convolution roughly 60,000 times more efficient computationally. Of course, most of the entries of the matrix would be zero. If we stored only the nonzero entries of the matrix, then both matrix multiplication and convolution would require the same number of floating point operations to compute. The matrix would still need to contain \( 2 \times 319 \times 280 = 178,640 \) entries. Convolution is an extremely efficient way of describing transformations that apply the same linear transformation of a small, local region across the entire input. (Photo credit: Paula Goodfellow)
• Learned kernels (filters)
Sparse connections and shared weights

[Unit]

Input

\[
\begin{array}{cccc}
\text{a} & \text{b} & \text{c} & \text{d} \\
\text{e} & \text{f} & \text{g} & \text{h} \\
\text{i} & \text{j} & \text{k} & \text{l}
\end{array}
\]

Kernel

\[
\begin{array}{cc}
\text{w} & \text{x} \\
\text{y} & \text{z}
\end{array}
\]

Output

\[
\begin{array}{cccc}
\text{aw} + \text{bx} + \text{cy} + \text{fz} & \text{bw} + \text{cx} + \text{fy} + \text{gz} & \text{cw} + \text{dx} + \text{gy} + \text{hz} \\
\text{ew} + \text{fx} + \text{iy} + \text{jz} & \text{fw} + \text{gx} + \text{jy} + \text{kz} & \text{gw} + \text{hx} + \text{ky} + \text{lz}
\end{array}
\]

[Figure 9.1, Deep Learning, Book]
Sparse connections and shared weights
Sparse connections and shared weights

- Kernels induce sparse and shared connections
- Kernels must be small compared to the data

Unit

Input

Kernel

Output

$aw + bx + cy + fz$

$bw + cz + fy + gz$

$cw + dx + gy + hz$

$ew + fx + iy + jz$

$fw + gz + jy + kz$

$gw + hx + ky + lz$

[Figure 9.1, Deep Learning, Book]
Figure 9.2: Sparse connectivity, viewed from below:

We highlight one input unit, $x_3$, and also highlight the output units in $s$ that are affected by this unit.

(Top) When $s$ is formed by convolution with a kernel of width 3, only the output $s$ is affected by $x_3$.

(Bottom) When $s$ is formed by matrix multiplication, connectivity is no longer sparse, so all of the outputs are affected by $x_3$. 

Sparse Connections

Dense Connections
Figure 9.3: Sparse connectivity, viewed from above: We highlight one output unit, $s_3$, and also highlight the input units that affect this unit. These units are known as the receptive field of $s_3$.

(Top) When $s$ is formed by convolution with a kernel of width 3, only the inputs affect $s_3$.

(Bottom) When $s$ is formed by matrix multiplication, connectivity is no longer sparse, so all of the inputs affect $s_3$.

Figure 9.4: The receptive field of the units in the deeper layers of a convolutional network is larger than the receptive field of the units in the shallow layers. This effect increases if the network includes architectural features like strided convolution (figure 9.12) or pooling (section 9.3). This means that even though direct connections in a convolutional net are very sparse, units in the deeper layers can be indirectly connected to all or most of the input image.
• Sparsity can reduce the number of parameters

• Fully connected: $O(m \times n)$

• Sparsely connected: $O(m \times k)$

• In practice for image data $k \ll n$
• Sparsity can reduce the number of parameters

• Fully connected: $O(m \times n)$

• Sparsely connected: $O(m \times k)$

• In practice for image data $k << n$

• Parameter sharing reduces the memory requirements of the network
• Sparsity can reduce the number of parameters

• Fully connected: $O(m \times n)$

• Sparsely connected: $O(m \times k)$

• In practice for image data $k << n$

• Parameter sharing reduces the memory requirements of the network

• Convolution is usually followed by a non-linear transformation (detector stage)
• We assumed grayscale images

• The same principle applies to color images where you learn kernels for each RGB channel

• In that case an image is encoded as a 3D tensor

• Kernels are also 3D

[http://www.sketchpad.net/channels1.htm]
Pooling

• Make the representation invariant to small translations in the input

• “Pool” the value of neighbour units

• E.g., max-pooling takes the max from its input.
Pooling

- Make the representation invariant to small translations in the input
- “Pool” the value of neighbour units
- E.g., max-pooling takes the max from its input.
Pooling

- Make the representation invariant to small translations in the input
- “Pool” the value of neighbour units
- E.g., max-pooling takes the max from its input.

![Diagram of pooling stages with max-pooling](Figure 9.8, Deep Learning, Book)
Pooling (II)

- Pooling over different features can learn other types of invariances

- Here, the model learns to be invariant to certain rotations:

![Diagram showing invariance to rotations]
Pooling (III)

- Often used to reduce the dimensionality

- Two parameters:
  - Width: size of neighbourhood to pool from
  - Stride: space between different neighborhoods

**Figure 9.9:** Example of learned invariances: A pooling unit with three filters can learn to be invariant to transformations of the input. Here we show how a set of three learned filters and a max pooling unit can learn to become invariant to rotation. All three filters are intended to detect a hand-written 5. Each filter attempts to match a slightly different orientation of the 5. When a 5 appears in the input, the corresponding filter will match it and cause a large activation in a detector unit. The max pooling unit then has a large activation regardless of which detector unit was activated. We show here how the network processes two different inputs, resulting in two different detector units being activated. The effect on the pooling unit is roughly the same either way. This principle is leveraged by maxout networks (Goodfellow et al., 2013a) and other convolutional networks. Max pooling over spatial positions is naturally invariant to translation; this multi-channel approach is only necessary for learning other transformations.

**Figure 9.10:** Pooling with downsampling. Here we use max-pooling with a pool width of three and a stride between pools of two. This reduces the representation size by a factor of two, which reduces the computational and statistical burden on the next layer. Note that the rightmost pooling region has a smaller size, but must be included if we do not want to ignore some of the detector units.
Putting it all together

Figure 9.7: The components of a typical convolutional neural network layer. There are two commonly used sets of terminology for describing these layers. (Left) In this terminology, the convolutional net is viewed as a small number of relatively complex layers, with each layer having many “stages.” In this terminology, there is a one-to-one mapping between kernel tensors and network layers. In this book we generally use this terminology. (Right) In this terminology, the convolutional net is viewed as a larger number of simple layers; every step of processing is regarded as a layer in its own right. This means that not every “layer” has parameters.

[Figure 9.7, Deep Learning, Book]
Typical Task

- Object recognition
- Architectures:
  - Repeat: Conv-Relu layers followed by pooling
  - Last layer is fully-connected

[https://blog.goodaudience.com/convolutional-neural-net-in-tensorflow-e15e43129d7d]
Variety of architectures

Inception

U-nets

ResNets


Pretrained models are available

- [https://keras.io/applications/](https://keras.io/applications/)

### Documentation for individual models

<table>
<thead>
<tr>
<th>Model</th>
<th>Size</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
<th>Parameters</th>
<th>Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xception</td>
<td>88 MB</td>
<td>0.790</td>
<td>0.945</td>
<td>22,910,480</td>
<td>126</td>
</tr>
<tr>
<td>VGG16</td>
<td>528 MB</td>
<td>0.713</td>
<td>0.901</td>
<td>1,385,377,44</td>
<td>23</td>
</tr>
<tr>
<td>VGG19</td>
<td>549 MB</td>
<td>0.713</td>
<td>0.900</td>
<td>1,436,672,40</td>
<td>26</td>
</tr>
<tr>
<td>ResNet50</td>
<td>98 MB</td>
<td>0.749</td>
<td>0.921</td>
<td>25,636,712</td>
<td>-</td>
</tr>
<tr>
<td>ResNet101</td>
<td>171 MB</td>
<td>0.764</td>
<td>0.928</td>
<td>44,707,176</td>
<td>-</td>
</tr>
<tr>
<td>ResNet152</td>
<td>232 MB</td>
<td>0.766</td>
<td>0.931</td>
<td>60,419,944</td>
<td>-</td>
</tr>
<tr>
<td>ResNet50V2</td>
<td>98 MB</td>
<td>0.760</td>
<td>0.930</td>
<td>25,613,800</td>
<td>-</td>
</tr>
<tr>
<td>ResNet101V2</td>
<td>171 MB</td>
<td>0.772</td>
<td>0.938</td>
<td>44,675,560</td>
<td>-</td>
</tr>
<tr>
<td>ResNet152V2</td>
<td>232 MB</td>
<td>0.780</td>
<td>0.942</td>
<td>60,380,648</td>
<td>-</td>
</tr>
<tr>
<td>ResNetXt50</td>
<td>96 MB</td>
<td>0.777</td>
<td>0.938</td>
<td>25,097,128</td>
<td>-</td>
</tr>
<tr>
<td>ResNetXt101</td>
<td>170 MB</td>
<td>0.787</td>
<td>0.943</td>
<td>44,315,560</td>
<td>-</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>92 MB</td>
<td>0.779</td>
<td>0.937</td>
<td>23,851,784</td>
<td>159</td>
</tr>
<tr>
<td>InceptionResNetV2</td>
<td>215 MB</td>
<td>0.803</td>
<td>0.953</td>
<td>55,873,736</td>
<td>572</td>
</tr>
<tr>
<td>MobileNet</td>
<td>16 MB</td>
<td>0.704</td>
<td>0.895</td>
<td>4,253,864</td>
<td>88</td>
</tr>
<tr>
<td>MobileNetV2</td>
<td>14 MB</td>
<td>0.713</td>
<td>0.901</td>
<td>3,538,984</td>
<td>88</td>
</tr>
<tr>
<td>DenseNet121</td>
<td>33 MB</td>
<td>0.750</td>
<td>0.923</td>
<td>8,062,504</td>
<td>121</td>
</tr>
<tr>
<td>DenseNet169</td>
<td>57 MB</td>
<td>0.762</td>
<td>0.932</td>
<td>14,307,880</td>
<td>169</td>
</tr>
<tr>
<td>DenseNet201</td>
<td>80 MB</td>
<td>0.773</td>
<td>0.936</td>
<td>20,242,984</td>
<td>201</td>
</tr>
<tr>
<td>NASNetMobile</td>
<td>23 MB</td>
<td>0.744</td>
<td>0.919</td>
<td>5,326,716</td>
<td>-</td>
</tr>
<tr>
<td>NASNetLarge</td>
<td>343 MB</td>
<td>0.825</td>
<td>0.960</td>
<td>88,949,818</td>
<td>-</td>
</tr>
</tbody>
</table>

The top-1 and top-5 accuracy refers to the model’s performance on the ImageNet validation dataset.
CNNs can be used as modules inside larger networks

- **Image Captioning**
  
  ![Image Captioning Examples](https://cs.stanford.edu/people/karpathy/cvpr2015.pdf)

- **Visual Question Answering**
  
  Is the umbrella upside down? yes no
  
  ![Visual Question Answering Example](https://arxiv.org/pdf/1612.00837.pdf)
Conclusions

- CNNs and RNNs are two classes of neural networks
  1. Lift limitations of feed-forward networks
  2. Allow modelling of sequential and image data
  3. Can be composed and learned end-to-end
Conclusions

• CNNs and RNNs are two classes of neural networks

  1. Lift limitations of feed-forward networks
  2. Allow modelling of sequential and image data
  3. Can be composed and learned end-to-end

• Other classes may emerge (e.g., graph networks)
Data

Neural Network Model

Convolutional Neural Network
Data

Neural Network Model

Convolutional Neural Network

Recurrent Neural Network
Data

Neural Network Model

Convolutional Neural Network

Recurrent Neural Network
Data

Neural Network Model

Convolutional Neural Network

Recurrent Neural Network

Graph Neural Networks
(Graph Convolutional Networks)