Intro to Neural Networks

Cynthia Rudin
Duke Machine Learning
Neurons

• $10^{11}$ neurons in a brain, $10^{14}$ synapses (connections).
• Signals are electrical potential spikes that travel through the network.

(Credit: Adapted from Russell and Norvig)
McCulloch-Pitts “Neuron”

\[
\sum_j w_{j,me} o_j \quad \text{“me”} = \phi \left( \sum_j w_{j,me} o_j \right)
\]
McCulloch-Pitts “Neuron”

\[
o_{me} = \phi \left( \sum_{j} w_{j,me} o_j \right)
\]
McCulloch-Pitts “Neuron”

\[ o_1 = ?, \quad w_1 = 1 \]
\[ o_2 = ?, \quad w_2 = 1 \]
\[ o_0 = -1, \quad w_0 = 1.5 \]

Step Function

<table>
<thead>
<tr>
<th>( o_1 )</th>
<th>( o_2 )</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
McCulloch-Pitts “Neuron”

\[
\begin{align*}
o_1 &= ? & w_1 &= 1 \\
o_2 &= ? & w_2 &= 1 \\
o_0 &= -1 & w_0 &= 1.5
\end{align*}
\]

\[0 + 0 - 1.5 = -1.5\]

Step Function

<table>
<thead>
<tr>
<th>o_1</th>
<th>o_2</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
McCulloch-Pitts “Neuron”

\[ o_1 = ? \quad w_1 = 1 \]
\[ o_2 = ? \quad w_2 = 1 \]
\[ o_0 = -1 \quad w_0 = 1.5 \]

\[ 0 + 0 - 1.5 = -1.5 \]
McCulloch-Pitts “Neuron”

\[ o_1 = ? \quad w_1 = 1 \]
\[ o_2 = ? \quad w_2 = 1 \]
\[ o_0 = -1 \quad w_0 = 1.5 \]

\[ 1 + 0 - 1.5 = -0.5 \]

Step Function

\[
\begin{array}{c|c|c|}
\hline
o_1 & o_2 & \text{output} \\
\hline
0 & 0 & 0 \\
1 & 0 & 0 \\
\hline
\end{array}
\]
McCulloch-Pitts “Neuron”

\[ o_1 = \text{?} \quad w_1 = 1 \]
\[ o_2 = \text{?} \quad w_2 = 1 \]
\[ o_0 = -1 \quad w_0 = 1.5 \]

\[ 0 + 1 - 1.5 = -0.5 \]

<table>
<thead>
<tr>
<th>( o_1 )</th>
<th>( o_2 )</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
McCulloch-Pitts “Neuron”

\[ o_1 = ?, \quad w_1 = 1 \]
\[ o_2 = ?, \quad w_2 = 1 \]
\[ o_0 = -1, \quad w_0 = 1.5 \]

1 + 1 - 1.5 = 0.5

<table>
<thead>
<tr>
<th>( o_1 )</th>
<th>( o_2 )</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
McCulloch-Pitts “Neuron”

\[ o_1 = ? \quad w_1 = 1 \]
\[ o_2 = ? \quad w_2 = 1 \]
\[ o_0 = -1 \quad w_0 = 1.5 \]

\[ 1+1-1.5 = 0.5 \]

<table>
<thead>
<tr>
<th>( o_1 )</th>
<th>( o_2 )</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
McCulloch-Pitts “Neuron”

\[ o_1 = ?, \quad w_1 = 1 \]
\[ o_2 = ?, \quad w_2 = 1 \]
\[ o_0 = -1, \quad w_0 = 1.5 \]

This neuron computes the function “and.”

There are “or” and “not” neurons too.

<table>
<thead>
<tr>
<th>( o_1 )</th>
<th>( o_2 )</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
McCulloch-Pitts “Neuron”

\[ \phi \left( \sum_j w_{j,me} a_j \right) = \frac{1}{1 + e^{-x}} \]

“Sigmoid”

Activation Function
Single Layer

$E = \frac{1}{2} (y - o_{100})^2$

Error on one observation
In a brain, the synapses strengthen and weaken in order to learn. Say the same thing happens here. How should we set the weights in order to learn (reduce the error)? Minimize $E$ with respect to the weights.

$E = \frac{1}{2}(y - o_{100})^2$

Error on one observation
Backpropagation

• An algorithm that trains the weights of a neural network
• Requires us to propagate information backwards through the network, then forwards, then backwards, then forwards, etc.
• Propagate backwards = chain rule from calculus.
Backpropagation

Cynthia Rudin
Duke Machine Learning
Backpropagation

• An algorithm that trains the weights of a neural network
• Requires us to propagate information backwards through the network, then forwards, then backwards, then forwards, etc.
• Propagate backwards = chain rule from calculus.
Single Layer

\[ E = \frac{1}{2} (y - o_{100})^2 \]

Error on one observation

\[ \phi(z) = \frac{1}{1 + e^{-z}} \]

\[ \phi'(z) = \frac{d\phi(z)}{dz} = \phi(z)(1 - \phi(z)) \]

\[ \frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \cdot \frac{do_{100}}{d net_{100}} \cdot \frac{d net_{100}}{dw_{99,100}} \]
Single Layer

\[ E = \frac{1}{2} (y - o_{100})^2 \]

Error on one observation

\[
\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\text{net}_{100}} \frac{d\text{net}_{100}}{dw_{99,100}}
\]
Single Layer

\[
E = \frac{1}{2} (y - o_{100})^2
\]

\[
\frac{dE}{do_{100}} = -\frac{1}{2} 2(y - o_{100})
\]

\[
\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \cdot \frac{do_{100}}{d\text{net}_{100}} \cdot \frac{d\text{net}_{100}}{dw_{99,100}}
\]

\[-(y - o_{100})\]
Single Layer

\[
E = \frac{1}{2} (y - o_{100})^2
\]

Error on one observation
Single Layer

\[
E = \frac{1}{2} (y - o_{100})^2
\]

Error on one observation

\[
\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d \phi(\text{net}_{100})} \frac{d \phi(\text{net}_{100})}{d \text{net}_{100}} \frac{d \text{net}_{100}}{dw_{99,100}}
\]

\[-(y - o_{100})\]
Single Layer

\[
\frac{do_{100}}{d \ net_{100}} = \frac{d \phi(\text{net}_{100})}{d \ net_{100}} = \phi'(\text{net}_{100}) = \phi(\text{net}_{100})(1 - \phi(\text{net}_{100})) = o_{100}(1 - o_{100})
\]

\[
\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d \ net_{100}} \frac{d \ net_{100}}{dw_{99,100}} = -(y - o_{100})
\]
Single Layer

\[
\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\text{net}_{100}} \frac{d\text{net}_{100}}{dw_{99,100}} = -(y - o_{100}) \cdot o_{100}(1 - o_{100})
\]

\[
\phi(\text{net}_{100}) = o_{100}
\]

\[
\frac{do_{100}}{d\text{net}_{100}} = \frac{d\phi(\text{net}_{100})}{d\text{net}_{100}} = \phi'(\text{net}_{100}) = \phi(\text{net}_{100})(1 - \phi(\text{net}_{100})) = o_{100}(1 - o_{100})
\]
Single Layer

\[
\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d net_{100}} \frac{d net_{100}}{dw_{99,100}}
\]

\[-(y - o_{100}) = o_{100} (1 - o_{100})\]
Single Layer

\[ E = \frac{1}{2} (y - o_{100})^2 \]

Error on one observation

\[ \frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\text{net}_{100}} \frac{d\text{net}_{100}}{dw_{99,100}} \]

\[-(y - o_{100})\]

\[ o_{100} (1 - o_{100}) \]
\[
\frac{d \text{ net}_{100}}{dw_{99,100}} = \frac{d \left( w_{99,100} o_{99} + w_{98,100} o_{98} + w_{97,100} o_{97} + \ldots \right)}{dw_{99,100}} = o_{99}
\]

\[
\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d \text{ net}_{100}} \frac{d \text{ net}_{100}}{dw_{99,100}}
\]

\[-(y - o_{100}) \quad o_{100} (1 - o_{100})\]
\[
\frac{dE}{dw_{99,100}} = \frac{dE}{do_{100}} \frac{do_{100}}{d\text{net}_{100}} \frac{d\text{net}_{100}}{dw_{99,100}}
\]

\[= -(y - o_{100}) o_{100} (1 - o_{100})
\]
We will need this later – it depends only on node 100

\[
\delta_{100} = \left( -(y - o_{100}) \right) o_{100} (1 - o_{100}) o_{99}
\]
Backpropagation

- Go one layer deeper.
\[
\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\text{net}_{98}} \frac{d\text{net}_{98}}{dw_{87,98}}
\]

\[
E = \frac{1}{2} (y - o_{100})^2
\]
\[
\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\text{net}_{98}} \frac{d\text{net}_{98}}{dw_{87,98}}
\]

\[
\frac{d\text{net}_{98}}{dw_{87,98}} = \frac{d}{dw_{87,98}} (w_{87,98} o_{87} + w_{86,98} o_{86} + w_{85,98} o_{85} + \ldots) = o_{87}
\]
\[
\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \cdot \frac{do_{98}}{d\text{net}_{98}} \cdot \frac{d\text{net}_{98}}{dw_{87,98}} \quad o_{87}
\]

\[
E = \frac{1}{2} (y - o_{100})^2
\]
\[
\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d \text{net}_{98}} \frac{d \text{net}_{98}}{dw_{87,98}}
\]

\[
\frac{do_{98}}{d \text{net}_{98}} = \frac{d\phi(\text{net}_{98})}{d \text{net}_{98}} = \phi'(\text{net}_{98}) = \phi(\text{net}_{98})(1 - \phi(\text{net}_{98})) = o_{98}(1 - o_{98})
\]
\[
\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \cdot \frac{do_{98}}{d \text{net}_{98}} \cdot \frac{d \text{net}_{98}}{dw_{87,98}}
\]

\[
E = \frac{1}{2} (y - o_{100})^2
\]

\[
o_{98} (1 - o_{98})
\]
\[ \frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\text{net}_{98}} \frac{d\text{net}_{98}}{dw_{87,98}} \]

\[ E = \frac{1}{2} (y - o_{100})^2 \]
\[
\frac{dE}{do_{98}} = \frac{dE}{d\text{net}_{100}} \cdot \frac{d\text{net}_{100}}{do_{98}} = \frac{dE}{dw_{87,98}} \cdot \frac{dw_{87,98}}{do_{98}} \cdot \frac{d\text{net}_{98}}{d\text{net}_{98}}
\]

\[
E = \frac{1}{2}(y - o_{100})^2
\]

\[
o_{98}(1 - o_{98})
\]
\[
\frac{d\text{net}_{100}}{do_{98}} = \frac{d(w_{99,100}o_{99} + w_{98,100}o_{98} + \ldots)}{do_{98}} = w_{98,100}
\]

\[
\frac{dE}{do_{98}} = \frac{d\text{net}_{100}}{d\text{net}_{100}} = \delta_{100}
\]

\[
\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} = \frac{d\text{net}_{98}}{d\text{net}_{98}} = \frac{d\text{net}_{98}}{d\text{net}_{98}} = o_{87}
\]

\[
o_{98}(1 - o_{98})
\]
\[
\frac{dE}{dw_{87,98}} = \frac{dE}{do_{98}} \frac{do_{98}}{d\text{net}_{98}} \frac{d\text{net}_{98}}{dw_{87,98}}
\]

\[
E = \frac{1}{2} (y - o_{100})^2
\]
\[
\frac{dE}{dw_{87,98}} = \delta_{100} w_{98,100} o_{98} (1 - o_{98}) o_{87}
\]

\[
E = \frac{1}{2} (y - o_{100})^2
\]
Backpropagation

• Go even one layer deeper.
• Third time is a charm.
\[
\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \cdot \frac{do_{87}}{d\text{net}_{87}} \cdot \frac{d\text{net}_{87}}{dw_{72,87}}
\]

\[
E = \frac{1}{2} (y - o_{100})^2
\]
\[
\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \cdot \frac{do_{87}}{d \text{net}_{87}} \cdot \frac{d \text{net}_{87}}{dw_{72,87}}
\]
\[
E = \frac{1}{2} (y - o_{100})^2
\]
\[
\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \cdot \frac{do_{87}}{d \text{net}_{87}} \cdot \frac{d \text{net}_{87}}{d w_{72,87}} = o_{72}
\]

\[
\frac{do_{87}}{d \text{net}_{87}} = \frac{d \phi(\text{net}_{87})}{d \text{net}_{87}} = \phi'(\text{net}_{87}) = \phi(\text{net}_{87})(1 - \phi(\text{net}_{87})) = o_{87}(1 - o_{87})
\]
\[ E = \frac{1}{2} (y - o_{100})^2 \]
\[
\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \frac{do_{87}}{d \text{net}_{87}} \frac{d \text{net}_{87}}{dw_{72,87}}
\]

\[
E = \frac{1}{2} (y - o_{100})^2
\]
\[
\frac{dE}{dw_{72,87}} = \frac{dE}{do_{87}} \cdot \frac{do_{87}}{d\text{net}_{87}} \cdot \frac{d\text{net}_{87}}{dw_{72,87}}
\]

\[
E = \frac{1}{2} (y - o_{100})^2
\]
\[
E = \frac{1}{2}(y - o_{100})^2
\]

\[
\begin{align*}
\frac{dE}{do_{87}} &= \frac{dE}{d\text{net}_{97}} \cdot \frac{d\text{net}_{97}}{do_{87}} + \frac{dE}{d\text{net}_{98}} \cdot \frac{d\text{net}_{98}}{do_{87}} + \frac{dE}{d\text{net}_{99}} \cdot \frac{d\text{net}_{99}}{do_{87}} + \ldots \\
\frac{dE}{dw_{72,87}} &= \frac{dE}{do_{87}} \cdot \frac{do_{87}}{d\text{net}_{87}} + \frac{dE}{d\text{net}_{87}} \cdot \frac{d\text{net}_{87}}{do_{87}}
\end{align*}
\]
\[ E = \frac{1}{2} (y - o_{100})^2 \]

\[
\frac{dE}{do_{87}} = \delta_{99} w_{87,99} + \delta_{98} w_{87,98} + \delta_{97} w_{87,97} + \ldots
\]

\[
= \sum_{\ell \in L} \delta_{87,\ell} w_{87,\ell}
\]
\[
\frac{dE}{dw_{a,b}} = \frac{dE}{do_{b}} \cdot \frac{do_{b}}{d\text{net}_b} \cdot \frac{d\text{net}_b}{dw_{a,b}} \\
= \frac{dE}{do_{b}} \cdot \frac{do_{b}}{d\text{net}_b} \div O_a
\]
\[
\frac{dE}{dw_{a,b}} = \frac{dE}{do_b} \frac{do_b}{d \text{net}_b} \frac{d \text{net}_b}{dw_{a,b}} = \frac{dE}{do_b} o_b (1 - o_b) o_a
\]
\[
\frac{dE}{dw_{a,b}} = \frac{dE}{do_b} \frac{do_b}{d\text{net}_b} \frac{d\text{net}_b}{dw_{a,b}}
\]

\[
= \left( \sum_{\ell \in L} \delta_{\ell \omega_{b,\ell}} \right) o_b (1 - o_b) o_a
\]

The \( \ell' \)s are downstream. We must have already computed all the \( \delta_{\ell} \)s ahead of us to compute this.
Backpropagation

• Now we know how to compute \( \frac{dE}{dw_{a,b}} \) for all \( w_{a,b} \)'s.

• Let’s do gradient descent.

\[
\begin{align*}
    w_{a,b} \leftarrow w_{a,b} - \alpha \frac{dE}{dw_{a,b}}
\end{align*}
\]

• \( \alpha \) is between 0 and 1. Called the “learning rate”.

• Now we know how to propagate errors back through the network.

• Remember how to go forward?
Backpropagation

• Repeat going backwards (to calculate the gradients), adjusting the weights, and going forwards (to calculate the errors) over and over in order to learn.
Cross-Entropy is Logistic Loss

Cynthia Rudin
Duke Machine Learning
Convergence Problems in Neural Networks

Cynthia Rudin

Duke Machine Learning
Convergence Problems

- NN’s have **problems with convergence due to vanishing/exploding gradients and saddle points.**
- Vanishing gradients come from the flat part of the activation function.
- Exploding gradients happen when we realize that our gradient has vanished and so increase the learning rate and take huge step sizes to compensate (but then mess everything up!)
- Stick to $10^{-5}$ to $10^{-3}$ learning rate perhaps?
Convergence Problems

• With the sigmoid activation, the derivatives of the input weights for each node are always either all positive or all negative. This is a limitation.

\[
\frac{dE}{dw_{a,b}} = \frac{dE}{do_b} \frac{do_b}{d \text{net}_b} \frac{d \text{net}_b}{dw_{a,b}} = \left( \sum_{\ell \in L} \delta_{\ell} w_{b,\ell} \right) o_b (1 - o_b) o_a
\]

does not depend on \(a\), positive, since all outputs of sigmoid are between 0 and 1.
Convergence Problems

- Bottom line – most people do not use sigmoid-like activation functions, even though this is more biologically relevant.

Sigmoid \( \sigma(x) = \frac{1}{1 + e^{-x}} \)
Convergence Problems

- Bottom line – most people do not use sigmoid-like activation functions, even though this is more biologically relevant.

\[
\sigma(x) = \frac{1}{1 + e^{-x}}
\]

Sigmoid

\[
\tanh(x)
\]

Hyperbolic tangent
Convergence Problems

Rectified Linear Unit (ReLU)
$$\text{max}(0, x)$$

Removes vanishing gradients when nodes are “activated,” meaning \(x > 0\).

(Krizhevsky et al., 2012)

Leaky ReLU
$$\text{max}(0.1x, x)$$

Removes vanishing gradients, but prefers that non-activated nodes be as “non-activated” as possible (doesn’t make much sense)

(Mass et al., 2013; He et al., 2015)
Convergence Problems

Convergence Problems

Adding momentum to gradients
- adjust gradient to make current gradient similar to previous gradients
Convergence Problems

• **Initialization** of the networks weights is really important. I have no idea how to do it.

  Due to bad initialization?

![Graph showing loss over time](#)
Batch Normalization (Ioffe and Szegedy, 2015) is a step that:
- Normalizes the outputs $o_i$ of several nodes (a “mini-batch”) in the same layer. (As usual, subtract the mean of the $o_i$’s divide by their standard deviation).
- Includes the mean and standard deviation as separate parameters to be learned.
- Usually the normalization is before the nonlinear activation function.
- This adds regularization and helps to prevent flat gradients in the network but sometimes it messes things up.
Early stopping via validation set

Convergence Problems Summary

• There are lots of convergence problems

• vanishing gradients
  – Adjust the learning rate
  – Change the activation function (tanh, ReLU, leaky ReLU, etc.)
  – Use Batch Norm
  – Add Momentum

• bad minima
  – Initialization (somehow…)

• overfitting
  – Stop early using validation set
Convergence Problems Summary

When training a NN, you “become” part of the algorithm because you control its convergence so heavily.
Convolutional neural networks and the
inguition behind their architectures

Cynthia Rudin
Duke Machine Learning
Convolutional NN’s

• Convolve means to slide the filter over all spatial locations and sum up the filter weights times the inputs.
Convolutional NN’s

- Convolve means to slide the filter over all spatial locations and sum up the filter weights times the inputs.
- An edge filter will detect edges.
Convolutional NN’s

• Convolve means to slide the filter over all spatial locations and sum up the filter weights times the input.

  - Stride of 5 means we step by 5’s when we convolve.

  The thickness is the number of filters

The following layer is smaller by a factor of 5.
Convolutional NN’s

Image from LeCun et al 1998, reproduced in color from Li, Johnson, Yeung, 2017
Convolutional NN’s

- **Max pooling** means to convolve with a max function.
- Intuitively keeps track of whether an earlier filter has detected something.

```
1  2  2  4
2  5  1  8
3  0  4  4
6  1  7  6
```

2 x 2 max pool filter and stride 2

```
5  8
6  7
```
Zero-padding

• Add zeros around the image so that the dimensions work out.
AlexNet won the ImageNet Large Scale Visual Recognition Challenge in 2012. It achieved a top-5 error of 15.3%, more than 10.8 percentage points ahead of the runner up.

Image source: unknown
• AlexNet (Krizhevsky et al. 2012)

*ReLU was applied to every output of convolutional and fully connected layers

softmax over 1000 classes
Layer 1 AlexNet filters (Krizhevsky et al. 2012)
Source: Zeiler and Fergus, 2013
Convolutional NN’s

Image from LeCun et al 1998, reproduced also from Li, Johnson, Yeung, 2017
Autoencoders

$x$ → Encoder → $Z$ → “Latent vectors” → Decoder → $x'$
There has been much work since AlexNet.
Next: Improving performance of CNNs for computer vision.
Improving Performance of Neural Networks

Cynthia Rudin

Duke Machine Learning
Data Augmentation

Chinese Lantern Festival, Cary NC, 2017
Data Augmentation
- include artificial data, such as horizontal flips, rotations, resized, cropped training images, change contrast and brightness, distortion, etc.

Chinese Lantern Festival, Cary NC, 2017
Data Augmentation

- include artificial data, such as horizontal flips, rotations, resized, cropped training images, change contrast and brightness, distortion, etc.

Chinese Lantern Festival, Cary NC, 2017
Residual Nets (He et al., 2016)

We hope to fit \( H(x) \).

Slides recreated from Kaiming He’s tutorial
Residual Nets

We hope to fit $\mathbf{H}(x)$. We are now learning a residual of identity.

We hope to fit $\mathbf{F}(x)$. We are now learning a residual of identity.

$\mathbf{H}(x) = \mathbf{F}(x) + x$

Residual Nets

• By adding $x$, the derivative of the error with respect to $x$ increases by 1. Thus, less vanishing derivatives.

• Allowed networks to go much deeper than before. “From 10 to 1000 layers”

$$H(x) = F(x) + x$$

Residual Nets

He et al. Deep Residual Learning for Image Recognition, arXiv2015
Dropout (Srivastava et al., JMLR 2014)

• Forces signal to be “carried” throughout the network
• In each forward pass, for each neuron, with probability $p$, set all of its output weights to 0.
• $p$ is a hyperparameter, usually $p = 0.5$.
• During testing, use all nodes.

![Diagram of standard neural net and dropout application](Image from Srivastava et al JMLR 2014)
Dropout (Srivastava et al., JMLR 2014)

• As if we are training exponentially many “sub” models. Similar idea to bagging (averaging many separately trained models together).

• Creates a redundant encoding.

(a) Standard Neural Net
(b) After applying dropout.

Image from Srivastava et al JMLR 2014)
Dropout (Srivastava et al., JMLR 2014)

- As if we are training exponentially many “sub” models. Similar idea to bagging (averaging many separately trained models together).
- Creates a redundant encoding.

- has hat with a ball on top
- juggles
- oversized shoes
- lots of makeup
- bright colors

clown score
“Transfer” Learning

• Using information about the solution to one problem to help solve another.

• Use the early layers from a pretrained model in another network. Retrain only the weights from the last few layers.
A Big Bag of Tricks

• Dropout
• Batch Normalization
• Data Augmentation
• Residual Networks
• Activation Functions (ReLU, Leaky ReLU)
• Initialization
• Transfer Learning
Other ways to improve neural networks

• Change the dataset. Use fine-grained labels

Is there a fence in this picture?

• Understand the model so you know what’s wrong with it.
Warnings about Neural Networks for Computer Vision

Cynthia Rudin

Duke Machine Learning
CNNs can use the wrong information (confounding)
CNNs can use the wrong information (confounding)

Source: Wikimedia commons, West German soldiers in 1983

Ok, well, that was a bad dataset…
CNNs can use the wrong information (confounding)

Solution to this? Interpretability? Heavy testing? Massive data augmentation?
Deep fakes are dangerous
Deep fakes are dangerous

Lip-syncing Obama: New tools turn audio clips into realistic video

Jennifer Langston
UW News

University of Washington researchers have developed new algorithms that solve a thorny challenge in the field of computer vision: turning audio clips into realistic, lip-synced video of the person speaking those words.
Deep fakes are dangerous
GANS – Generative Adversarial Networks

- GANS are actor-critic models
- They produce realistic-looking images/data
- Used commonly for AI artwork / deep fakes

If the generator creates images that the discriminator can’t tell apart, it’s good. (The “arms race” is between the generators and the discriminators.)

*Goodfellow et al 2014*
GANS – Generative Adversarial Networks

From Goodfellow et al 2014:

\[ D \text{ and } G \text{ play the following two-player minimax game with value function } V(D, G): \]

\[
\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))].
\]

Discriminator maximizes likelihood of real data

Discriminator minimizes likelihood of generated data.

Generator maximizes likelihood of generated data.
GANS – Generative Adversarial Networks

From Goodfellow et al 2014:

\[ \min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]. \]

\( D \) and \( G \) play the following two-player minimax game with value function \( V(G, D) \):

- Discriminator maximizes likelihood of real data
- Discriminator minimizes likelihood of generated data.
- Generator aims to make discriminator not work well.
- Generator maximizes likelihood of generated data.
GANS – Generative Adversarial Networks

From Goodfellow et al 2014:

\[
\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))].
\]

\[
\max_G \mathbb{E}_{z \sim p_z(z)}[\log(1 \div D(G(z)))].
\]

Gradient ascent steps on discriminator
Gradient descent steps on generator
GANs are totally useful for artwork!

Is artificial intelligence set to become art’s next medium?

AI artwork sells for $432,500 — nearly 45 times its high estimate — as Christie’s becomes the first auction house to offer a work of art created by an algorithm.
GANs are totally useful for artwork!

Figure adapted from L. Gatys et al. "A Neural Algorithm of Artistic Style" (2015) by Google AI Blog
Menon et al. PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models, CVPR 2020


A twitter user’s result from the PULSE algorithm, which uses StyleGAN

GANs are totally useful for artwork!
PULSE shows us that there is often no hope of identifying someone in a grainy security video.

There could be many high res images corresponding to one low res image.

Menon et al. PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models, CVPR 2020


GANs are totally useful for artwork!
Neural networks can be brittle

- Adversarial attacks show that changing a single pixel in an image can change the predicted class in modern ML systems.
- It is easy to fool a computer vision system.

Eykholt et al., 2018 Robust Physical-World Attacks on Deep Learning Models,
The model will not always be used in the way it is intended

Black man wrongfully arrested because of incorrect facial recognition

Robert Williams spent nearly 30 hours in a detention center.

By Ella Torres

June 25, 2020, 2:01 PM • 6 min read
So…

• Much care is needed in many applications of neural networks.
  – medical image processing (confounding)
  – automated driving systems (not robust, not perfect)
  – facial recognition (not perfect, watch for bias)
  – deep fakes (easily fraudulent)

• Neural networks are great for artwork.