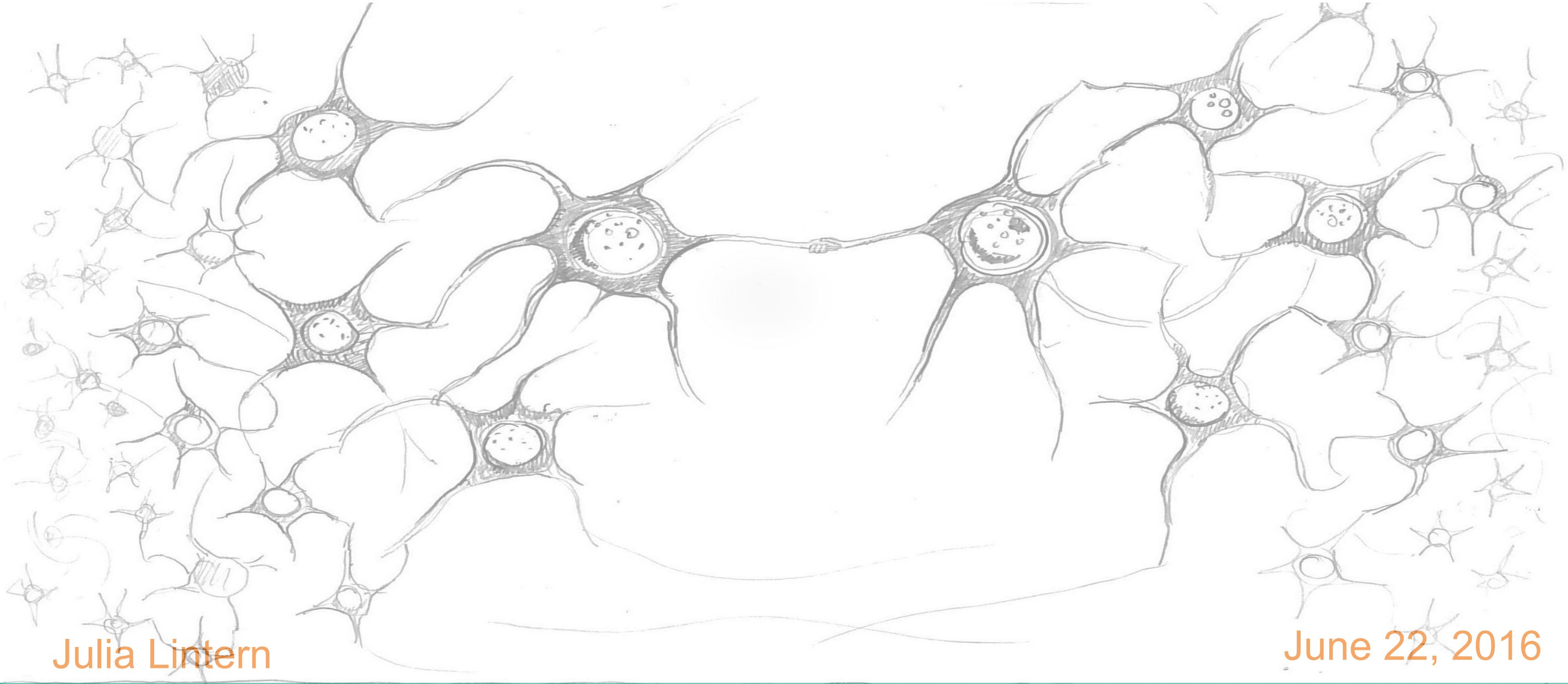
Navigating Neural Nets

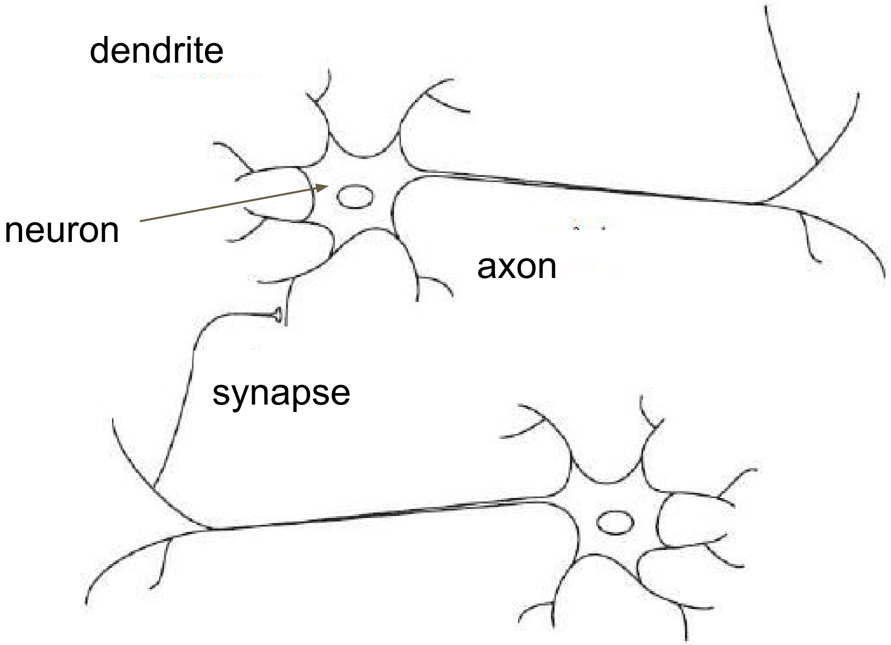


Julia Lintern

June 22, 2016

The Brain Analogy

(our cartoon neuron)

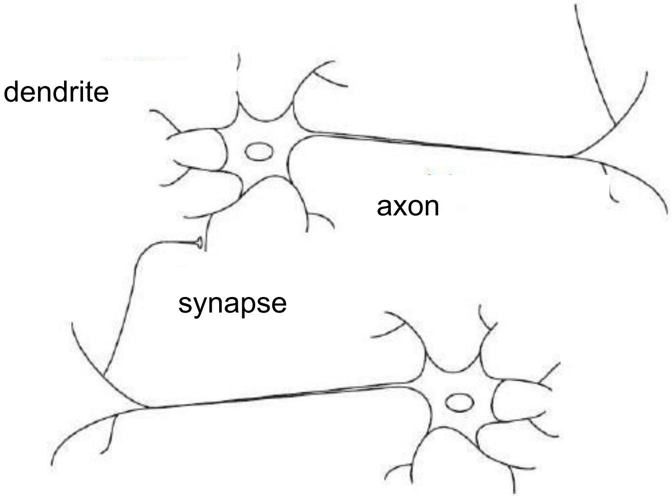


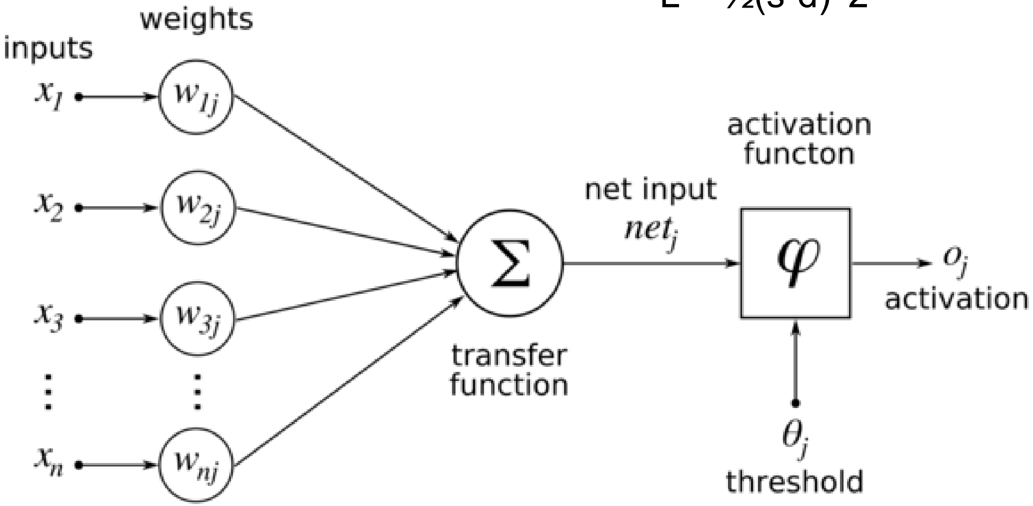
The Brain Analogy

(cartoon neuron & mathematical neuron)

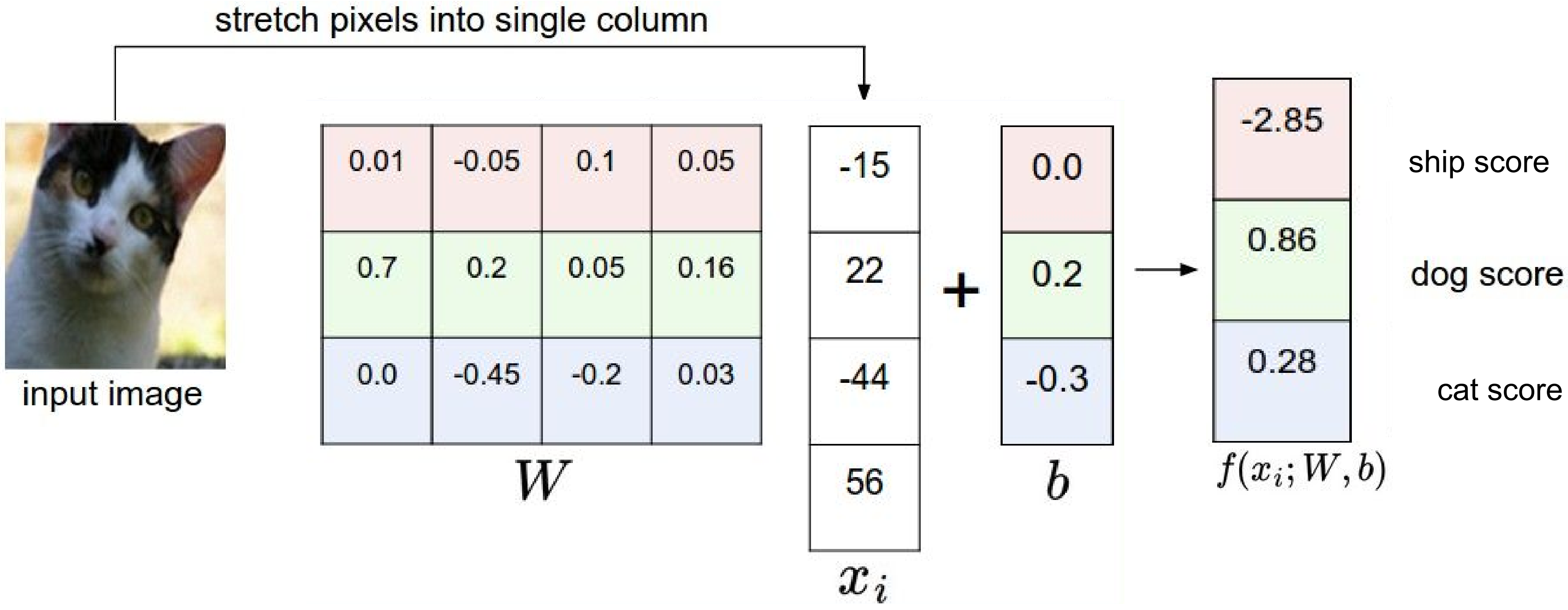
$s = f(x,w)$

$L = \frac{1}{2}(s-d)^2$





The Linear Classifier Analogy



Losses:

Multiclass SVM (Hinge) Loss

$$\underbrace{\begin{bmatrix}0.01 & -0.05 & 0.1 & 0.05\\ 0.7 & 0.2 & 0.05 & 0.16\\ 0.0 & -0.45 & -0.2 & 0.03\end{bmatrix}}_{W} \underbrace{\begin{bmatrix}-15\\ 22\\ -44\\ 56\end{bmatrix}}_{x_i} + \underbrace{\begin{bmatrix}0.0\\ 0.2\\ -0.3\end{bmatrix}}_{b}$$

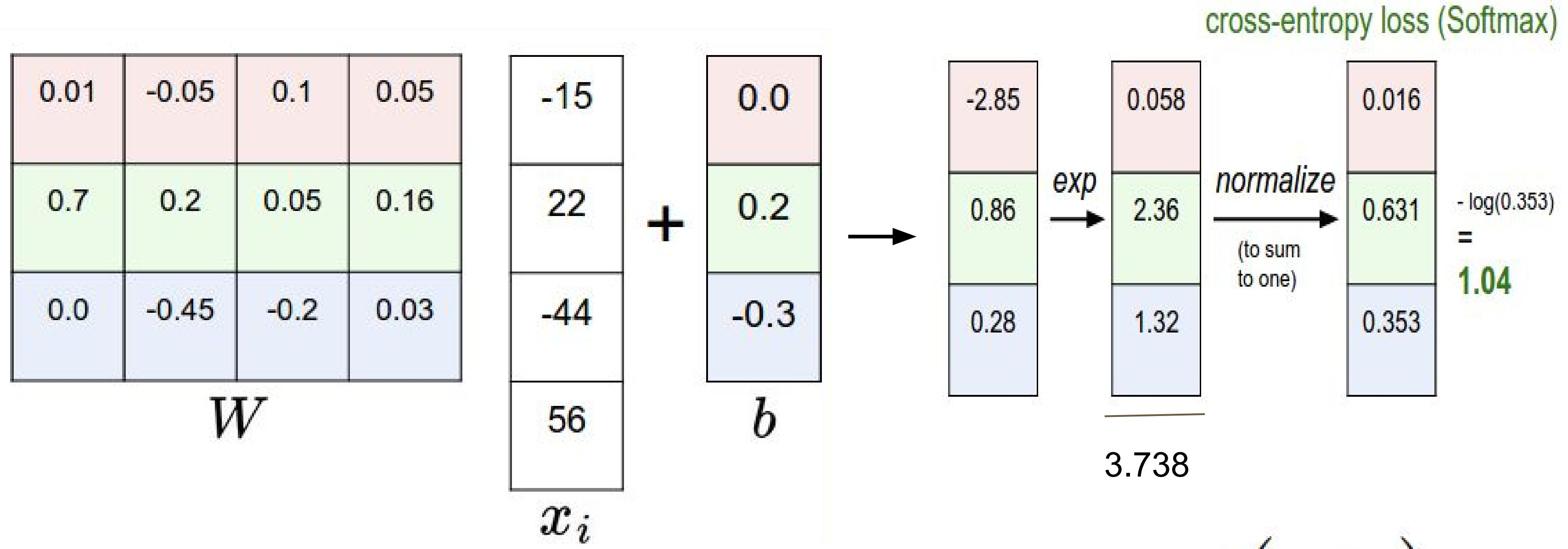
$$\begin{array}{cl} \begin{bmatrix}-2.85\\ 0.86\\ 0.28\end{bmatrix} & \begin{array}{l}\text{ship score}\\ \text{dog score}\\ \text{cat score}\end{array}\end{array}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta)$$

$$= \max(0, -2.85\text{-}.28 + \overset{=1}{\Delta}) + \max(0, 86\text{-}0.28 + \overset{=1}{\Delta}) = 0 + 1.58$$

Losses:

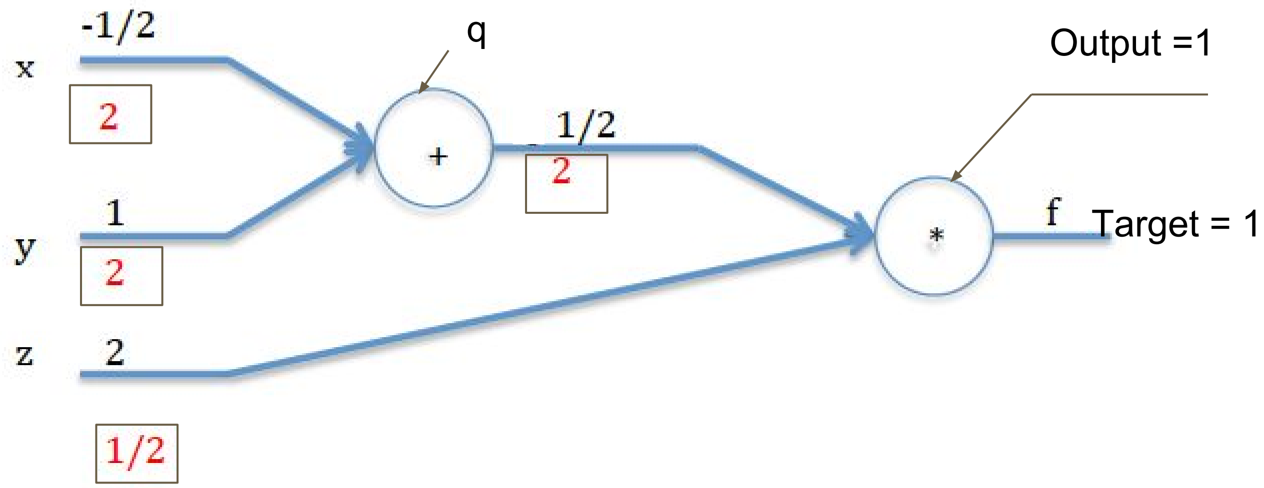
Softmax (Cross-Entropy) Loss



Softmax: $f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$

Cross-Entropy $L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$

BackPropagation

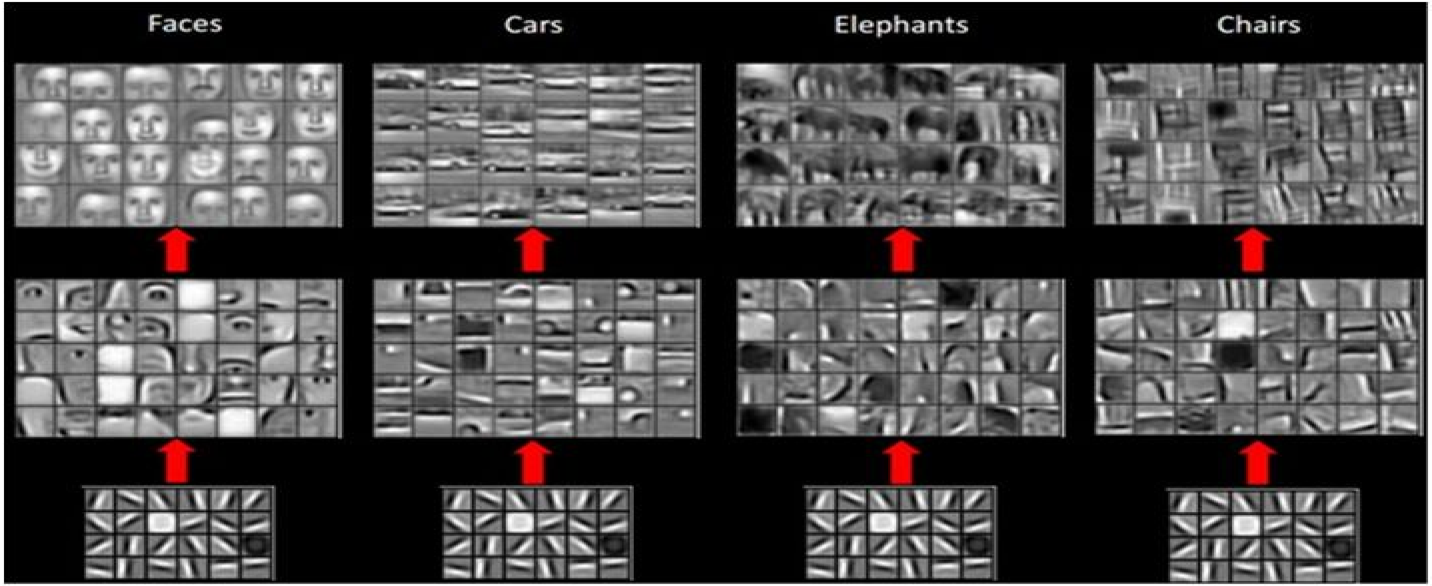


f(x,y,z)= (x+y)z
q= (x+y)

df/dx = ? => chain rule: df/dx = df/dq(dq/dx)
df/dy = ? => chain rule: df/dy = df/dq(dq/dy)

Convolutional Neural Nets

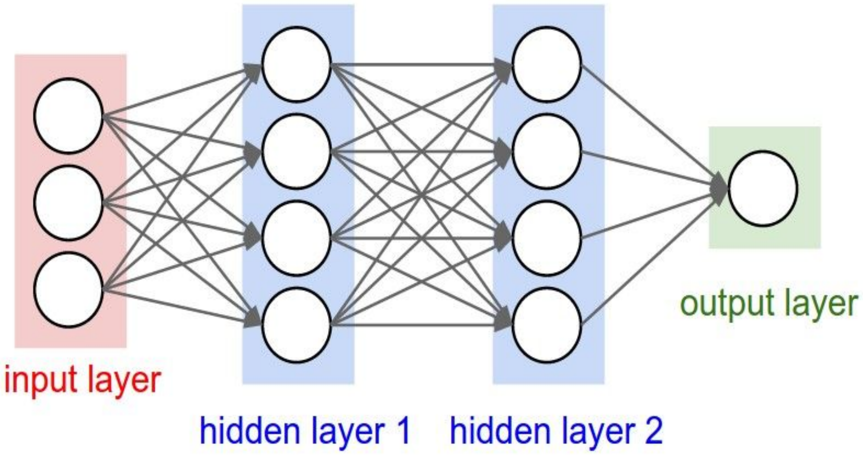
Very similar to Neural Nets.. But how are they different ?

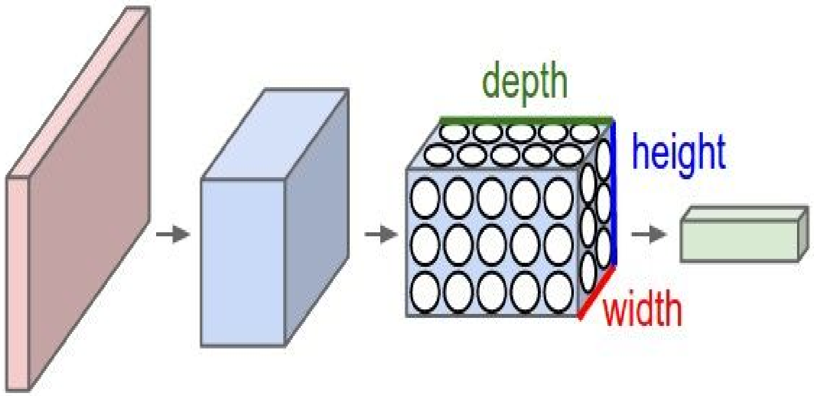


Convolutional Neural Nets

Vs. Neurals Nets

- Input is an image: Leverage 3D Structure
- Fully Connected ?   Not entirely!





The CNN Family

Winners of the ILSVRC ImageNet challenges

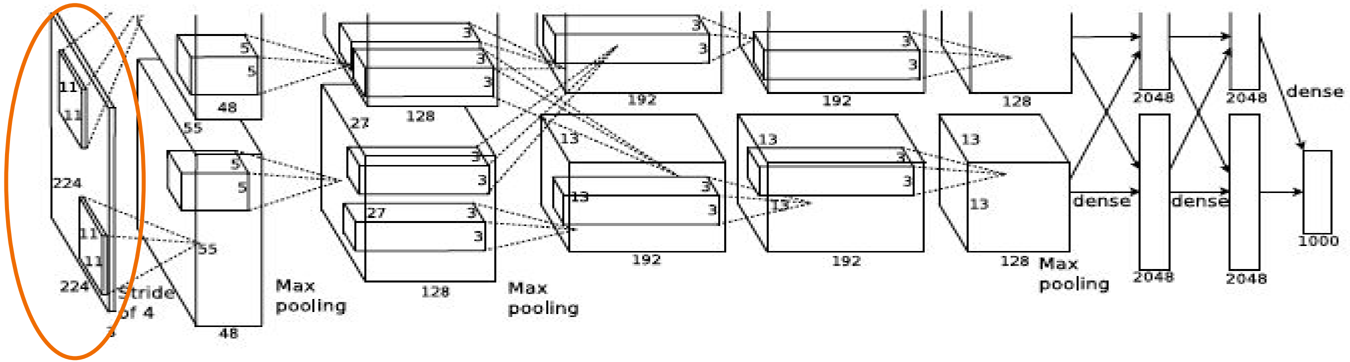
AlexNet (2012, Krizhevsky): Popularized CNNs - 1st to incorporate consecutive convolutional layers

GoogleNet / Inception (2014, Szegedy): Drastically reduced the # of parameters used (from 60 million to 4 million)

ResNet (2015, Kaiming He): Residual Network : famous for skip-connections and heavy use of batch-normalization; also removes some fully connected layers (at end of network)

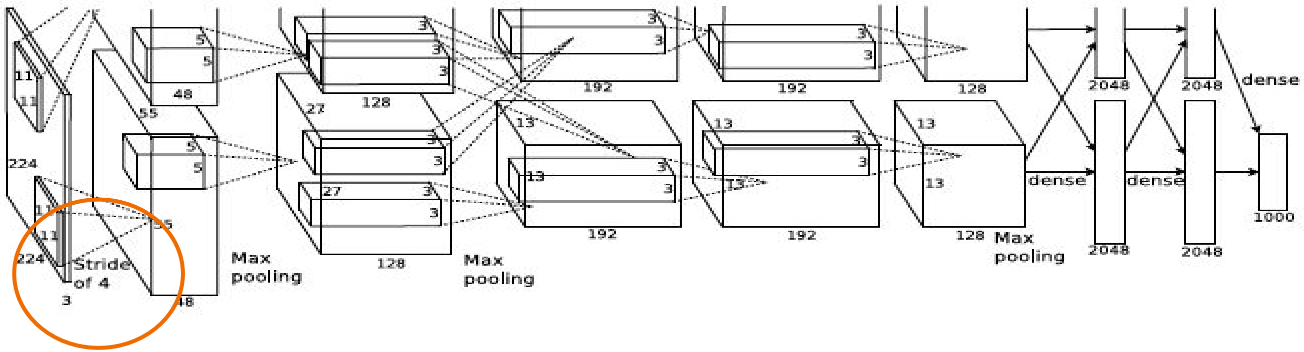
Convolutional Neural Nets : Architecture

1) **Input Layer: Raw pixel values of the image (ex: 224 x 224 x 3 (3 ~ color channels (RGB))**
2) Conv Layer
3) Pool Layer
4) ReLU Layer
5) FC (Fully Connected Layer)



Convolutional Neural Nets : Architecture

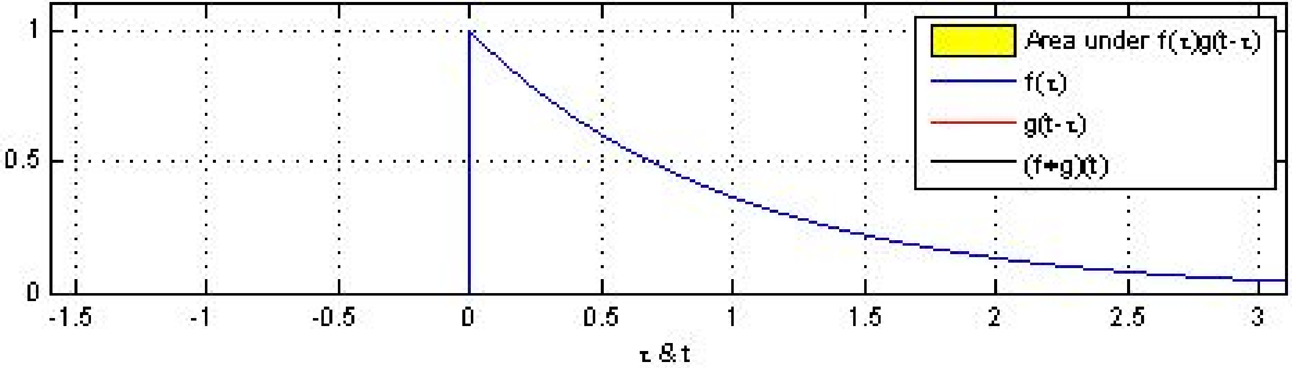
1) Input Layer: Raw pixel values of the image
2) **Conv Layer: Dot product between weights and the small region of input volume (ex: 11 x 11 x 3 filters)**
3) Pool Layer
4) ReLU Layer
5) FC (Fully Connected Layer)



Convolutional Neural Nets : Architecture

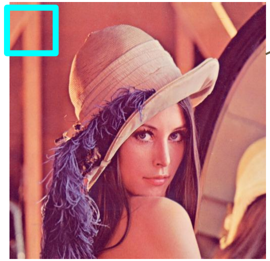
What is a Convolution?

$$f*g = \int f(t - \tau)g(\tau)d\tau$$



Convolutional Neural Nets : Architecture

What is a Convolution?



$$\begin{array}{c|c} \boxed{\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \end{array} \qquad \boxed{\begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}} \qquad \longrightarrow \qquad \mathbf{0}$$

$$\begin{matrix} 1 \\ 1 \\ 1 \end{matrix} \; \boxed{\begin{matrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{matrix}} \qquad \boxed{\begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}} \qquad \longrightarrow \qquad \mathbf{-3}$$

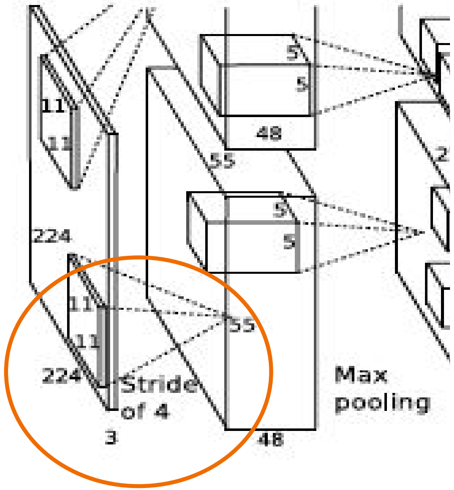
Convolutional Neural Nets : Architecture

What is a Convolution?

Convolutional Layer: (W-F + 2P)/S +1

- **W : Input Volume size**
- **F: Receptive Field size of the Conv Layer Neuron**
- **P: Zero- Padding**
- **S: Stride**

(224 - 11 +2(3))/4 + 1 = 55

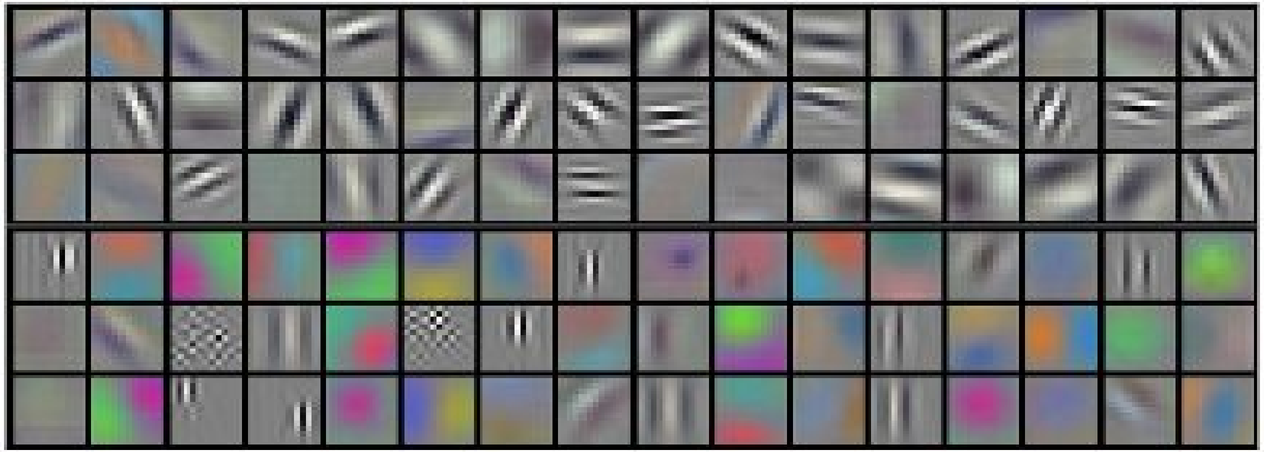


Conv Layer Output ~ 55 x 55 x 96 (ie : 55^2 neurons in each layer)

Convolutional Neural Nets : Architecture

What is a Convolution?

Voila. We have 96 filters.



Convolutional Neural Nets : Architecture

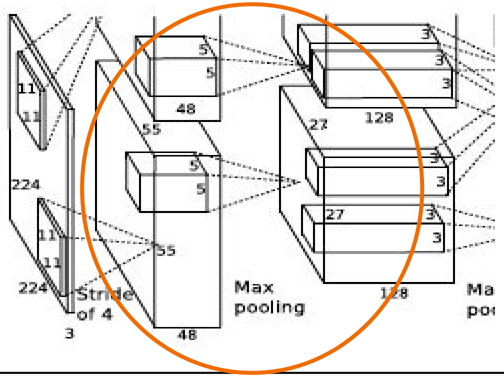
1) Input Layer
2) Conv Layer
3) **Pooling Layer: Performs downsampling operation**
4) ReLU Layer
5) FC (Fully Connected Layer)

$$\begin{matrix} 12 & 23 & 9 \\ 8 & 11 & 13 \\ 22 & 5 & 10 \end{matrix} \longrightarrow \boxed{23}$$

Our Eqn : O = (W- F)/S +1

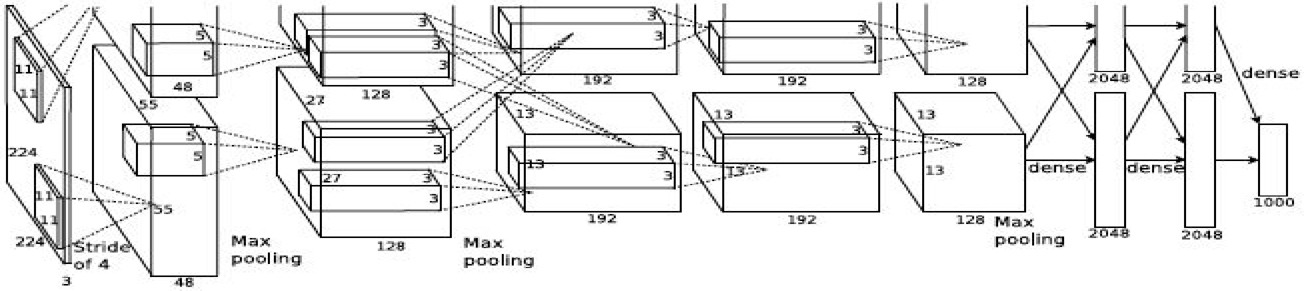
AlexNet: use 3 x 3 MaxPooling w/ stride = 2

O =(55-3)/2 + 1 = 27



Convolutional Neural Nets : Architecture

1) Input Layer: Raw pixel values of the image
2) Conv Layer:
3) Pool Layer:
4) **ReLU Layer: Apply an elementwise activation function**
   (ex : max(0,x) thresholding output dimension ~ same as input)
5) FC (Fully Connected Layer)

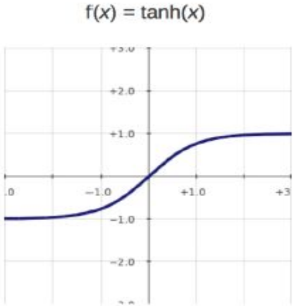


*The ReLU non-linearity is applied to the output of every convolutional and fully-connected layer.

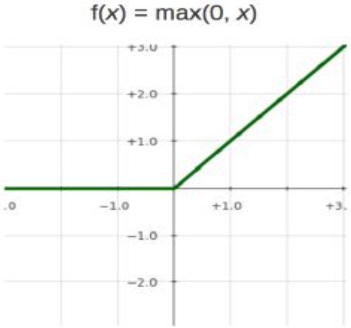
Convolutional Neural Nets : Architecture

ReLU Layer:

Tradionally:
f(x) = tanh(x) or fx= (1+e-x)^-1 (Very slow to train)

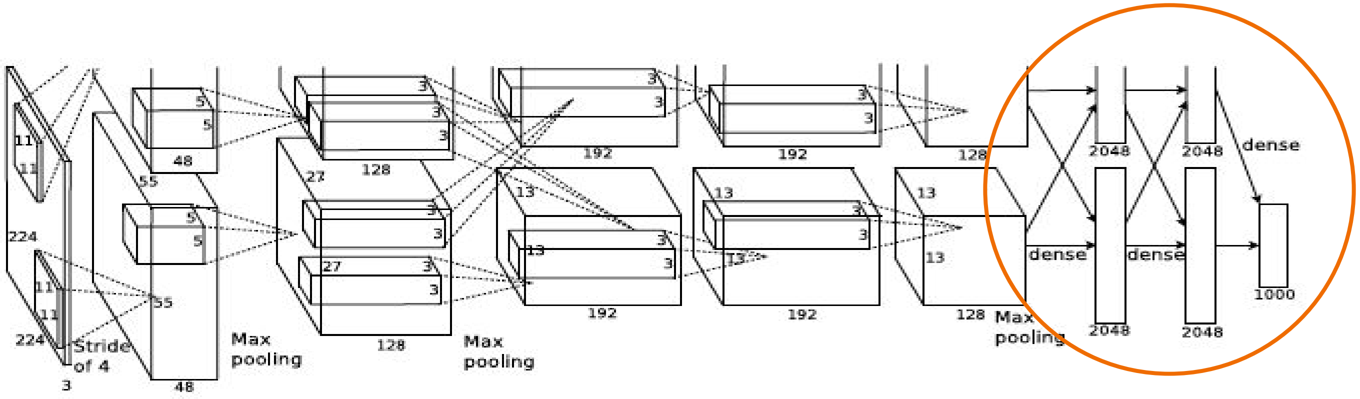


Now:
f(x) = max(0,x) (Faster to train)



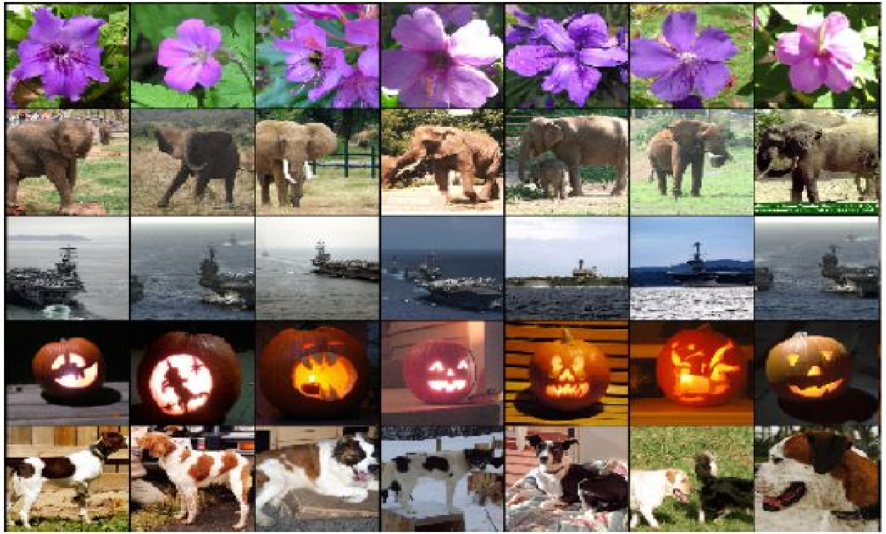
Convolutional Neural Nets : Architecture

1) Input Layer: Raw pixel values of the image
2) Conv Layer:
3) Pool Layer:
4) ReLU Layer:
5) **FC (Fully Connected) Layer : Each neuron will be connected to all activations of the previous volume. The output layer will compute class scores (ex: [1 x 1 x 1000])**



Convolutional Neural Nets : Architecture

Output from the final 4096 fully connected layer :



Torch:

- Fast. Easy to integrate with GPUs
- Many modular pieces that are easy to combine https://gith  th/imagenet-multiGPU.torch/blob/master/models/alexnet.lua
- Written in Lua

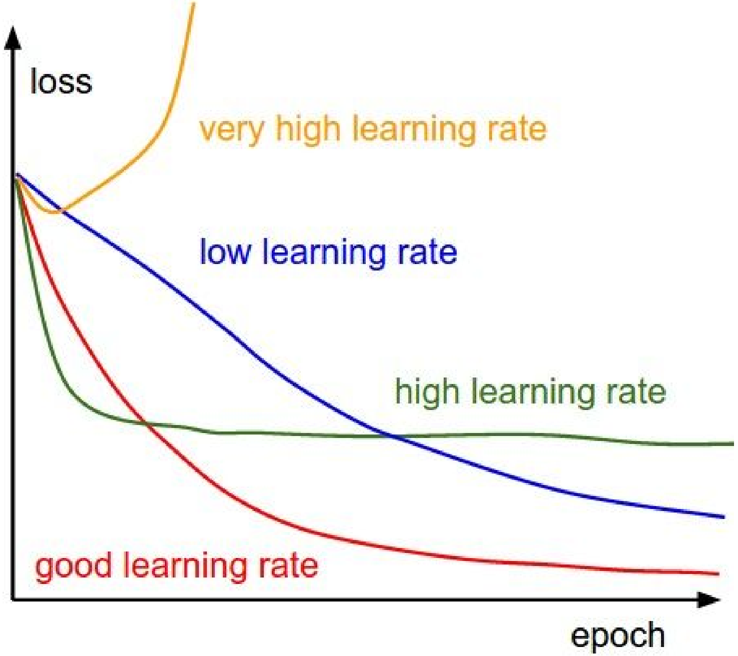
TensorFlow:

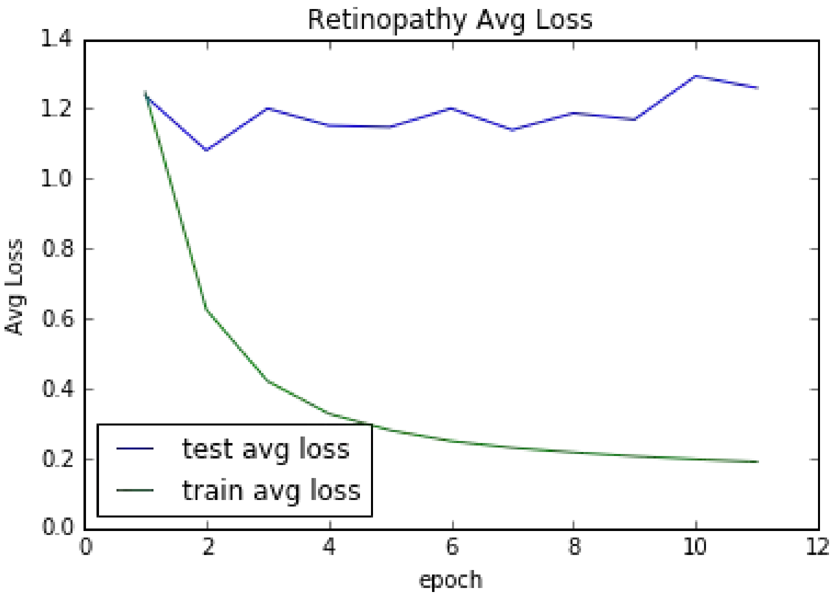
- Written in python & numpy
- Tensorboard for visualization
- Latest releases can be buggy
- Can be many x slower than Torch

https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:

Learning from the Learning Process

1) Loss functions

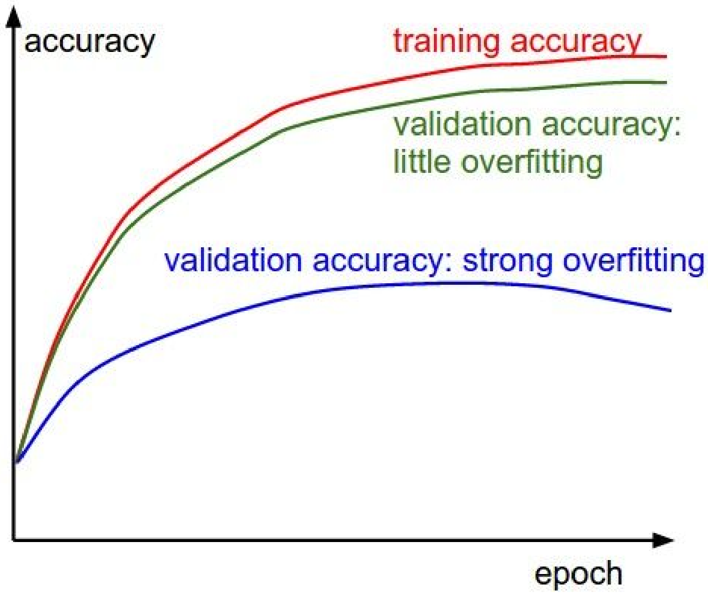


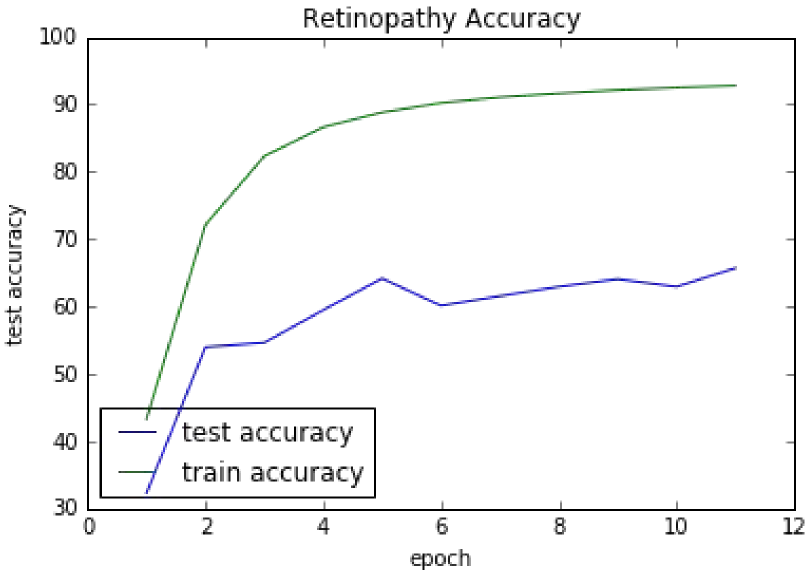


* Tip: Change the learning rate!

Learning from the Learning Process

2) Accuracy





* Tip: Increase L2 weight penalty , Increase Drop-Out, More Data (possibly with jitter) - -try batch norm ?

Learning from the Learning Process

3) First-layer Visualizations

Visualized weights from the 1st layer of the network:

(smooth, diverse features indicate that training is going well)