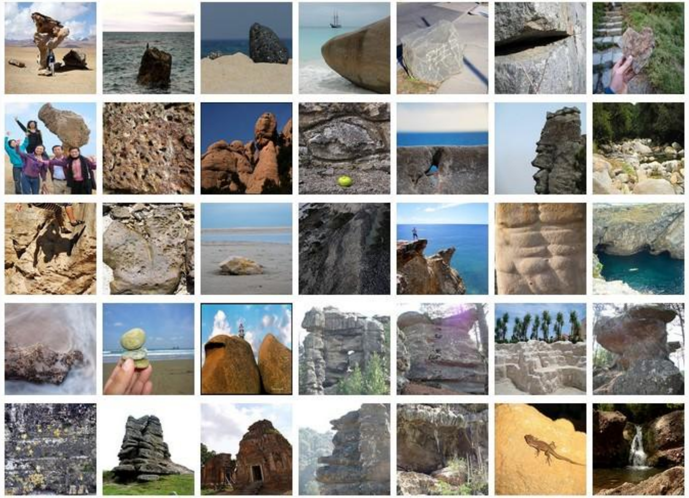
Deep Learning 1

Pointers from Last Time

- Saving your models
  - Just need to save parameters
  - Can Use it anywhere
  - E.g. in linear regression with y=w1x1 + w2 you can just save w1 and w2 forever!
  - No data dependence after training
- Cost function is only used during training. Prediction function is eternal.
- Linear regression can be made non linear just by adding multiple powers of the same feature

Motivation: ImageNet

Image Classification Dataset



~1,000,000 images
~1,000 classes

Ground truths prepared manually
through Amazon Mechanical Turk

ImageNet Top-5 challenge:

You score if ground truth class is one your top 5 predictions

ImageNet in 2012

Best approaches used hand-crafted features
(SIFT, HOGs, Fisher vectors, etc)
+ classifier

Top-5 error rate: ~25%

The Game Has Changed...

Krizhevsky, Sutskever and Hinton;
ImageNet Classification with
Deep Convolutional Neural networks **[Krizhevsky12]**

Top-5 error rate of ~15%

In the last few years, more modern networks have achieved better results.

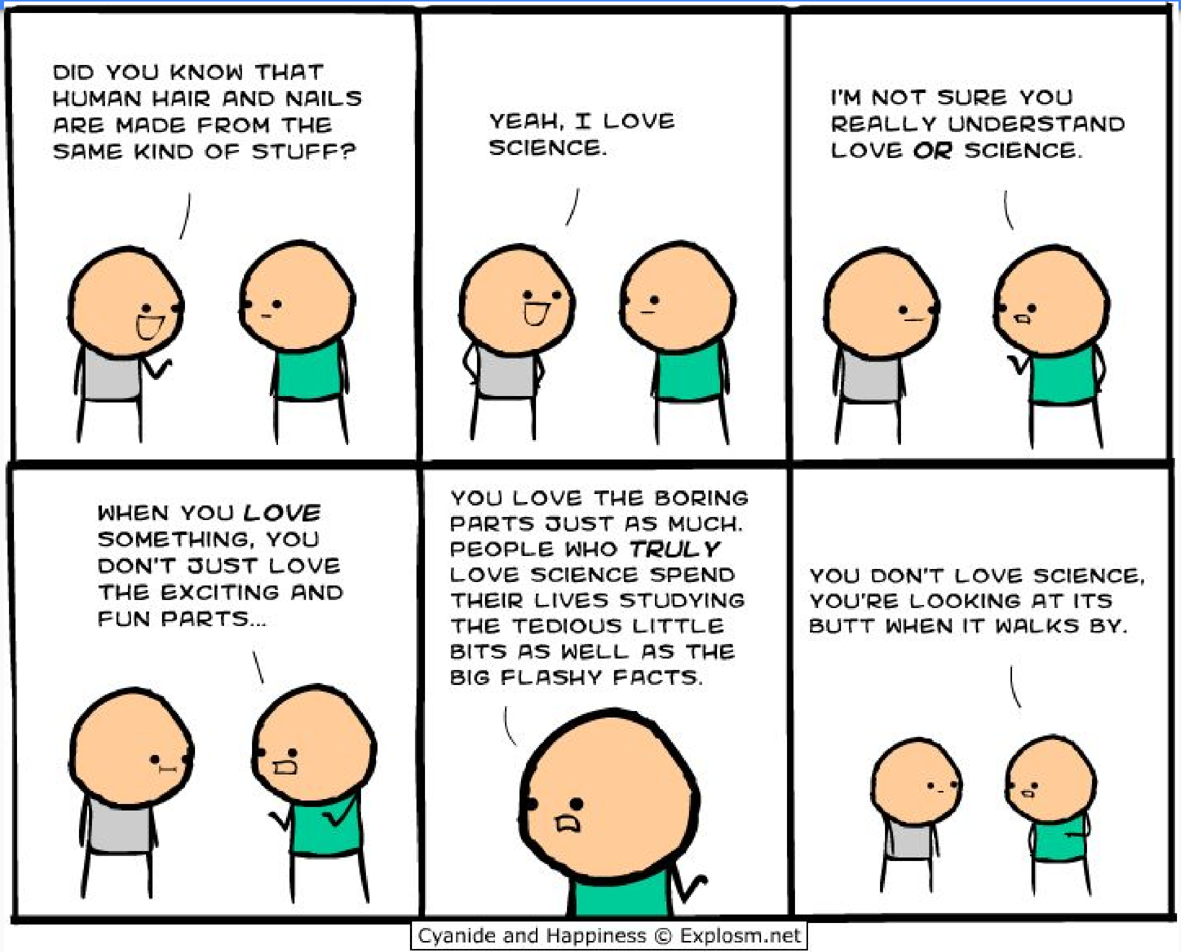
[Simonyan14, He15]

Top-5 error rates of ~5-7%

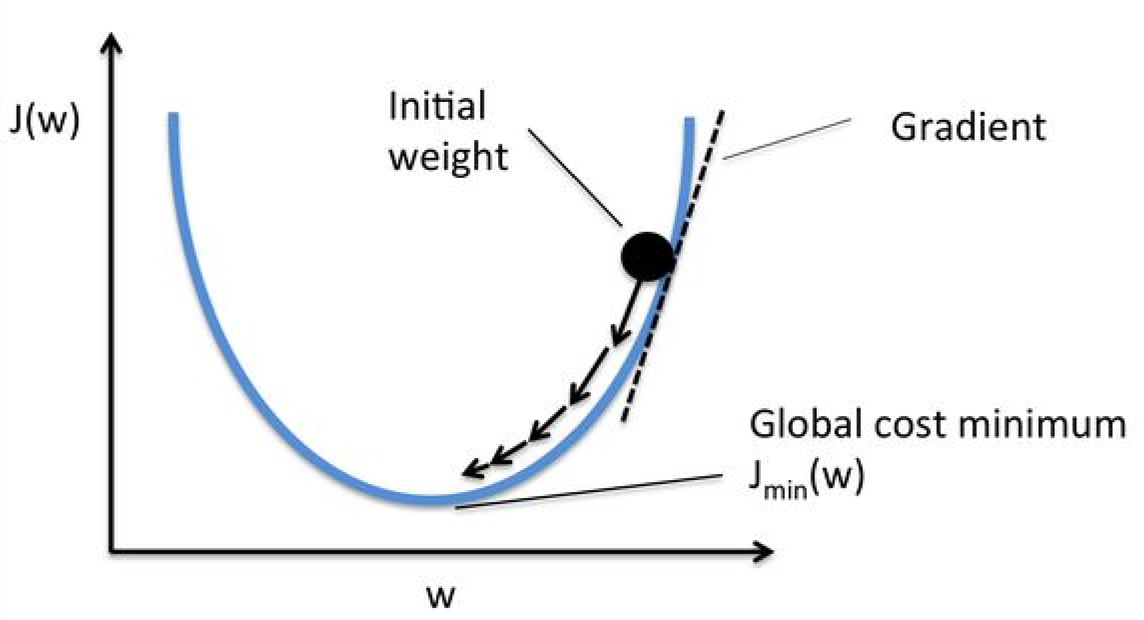
Gradient Descent: Understanding the Math

- We talked about loss functions (i.e. cost functions)
  - Absolute Loss $| h(x) - y |$
  - Quadrative Loss $(h(x) - y)^2$
- Real world is not so generous.
- The following cycle goes on and on until we have a reasonable model
  - Predict using current hypothesis (last time)
  - Find how good the prediction was (last time)
  - **Update the hypothesis. (this time)**
- Q: How Does the update step happen?
  - A: Gradient Descent

Motivation (http://explosm.net/comics/3557/)

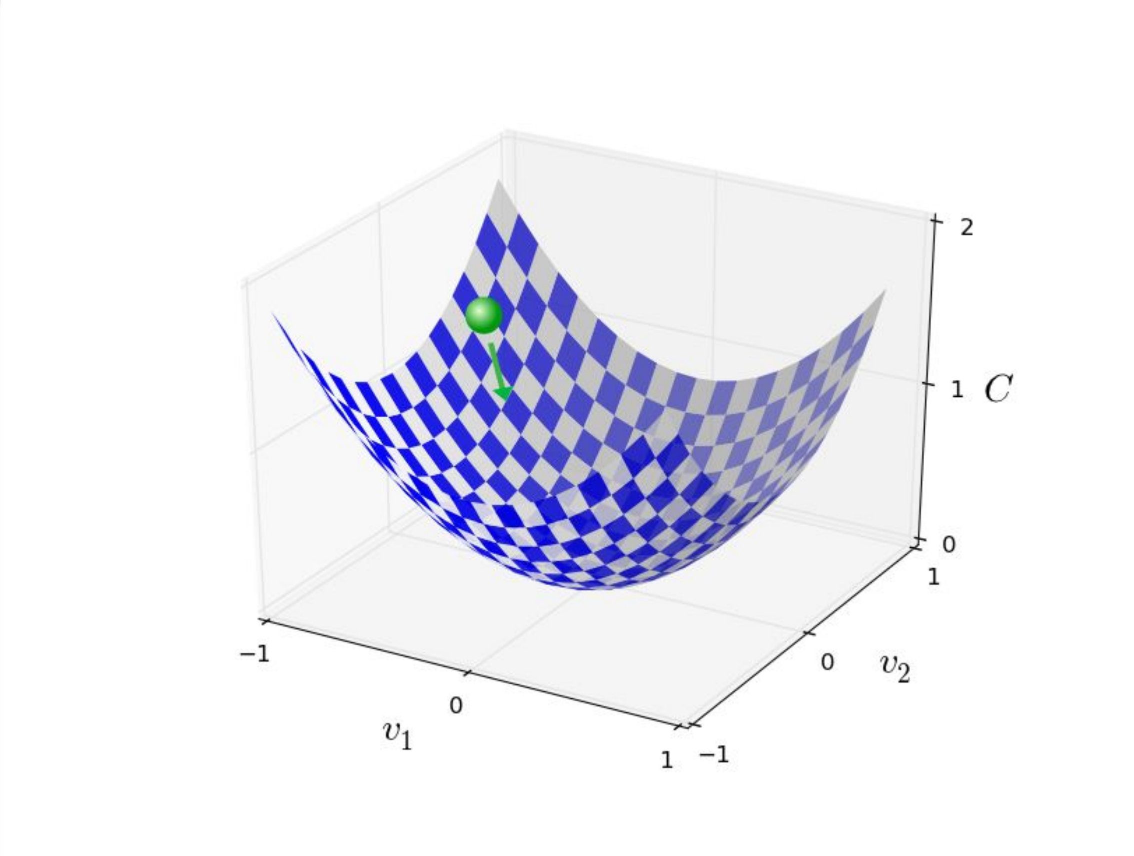


Gradient Descent (Cont.)



https://qph.ec.quoracdn.net/main-qimg-1ec77cdbb354c3b9d439fbe436dc5d4f?convert_to_webp=true

Gradient Descent (cont.)



http://neuralnetworksanddeeplearning.com/images/valley_with_ball.png

Algorithm

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(\text{for } j = 1 \text{ and } j = 0)$$

}

https://2.bp.blogspot.com/-AdV-O-MoZHE/TtLibFTaf9I/AAAAAAAAAVM/aOxUGP7zl98/s1600/gradient+descent+algorithm+OLS.png

Repeat until convergence

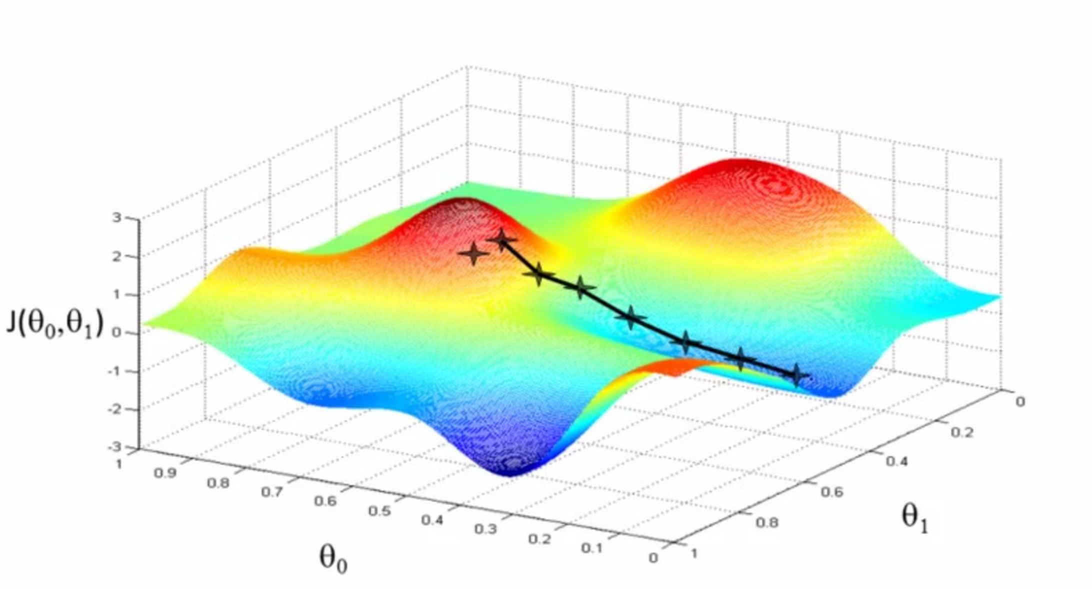
{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

}

http://2.bp.blogspot.com/-ZxJ87cWjPJ8/TtLtwqv0hCI/AAAAAAAAAV0/9FYqcxJ6dNY/s1600/gradient+descent+algorithm+OLS.png

Real world is ugly

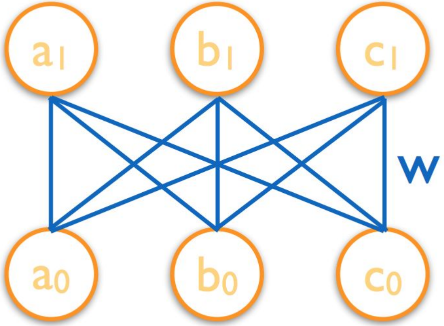


https://kousikk.files.wordpress.com/2014/11/screen-shot-2014-11-12-at-2-58-28-pm.png

TensorFlow

TensorFlow: Introduction

```
import tensorflow as tf
```



$$\underbrace{\begin{bmatrix} a_0 & b_0 & c_0 \end{bmatrix}}_{x} \cdot \underbrace{\begin{bmatrix} w_{a,a} & w_{a,b} & w_{a,c} \\ w_{b,a} & w_{b,b} & w_{b,c} \\ w_{c,a} & w_{c,b} & w_{c,c} \end{bmatrix}}_{w} = \begin{bmatrix} a_1 & b_1 & c_1 \end{bmatrix}$$

```
y = tf.matmul(x, w)
```

$$\boxed{a_1} = relu(\boxed{a_1})$$

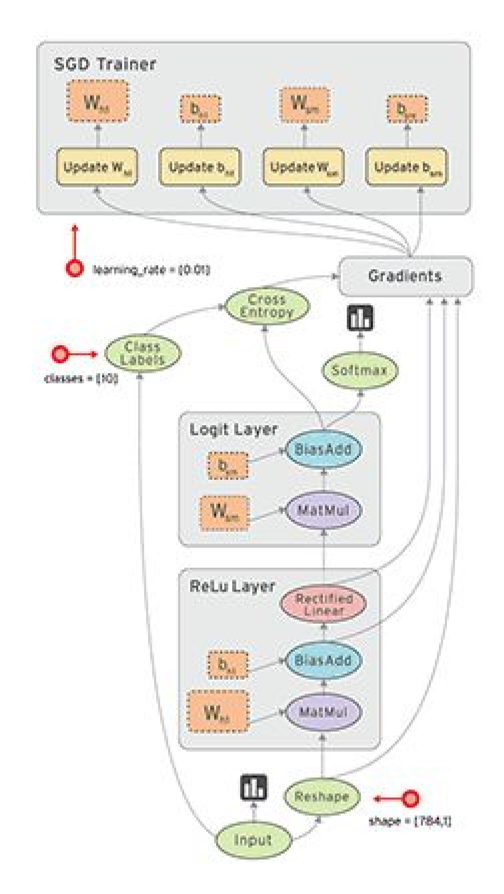
$$\boxed{b_1} = relu(\boxed{b_1})$$

$$\boxed{c_1} = relu(\boxed{c_1})$$

```
out = tf.nn.relu(y)
```

Data Flow Graph

- Describe mathematical computation with a directed graph of <u>nodes</u> & <u>edges</u>.
    - Nodes in the graph represent mathematical operations.
    - Edges describe the i/o relationships between nodes.
    - Data edges carry dynamically-sized multidimensional data arrays, or **tensors**.
- Nodes are assigned to computational devices and execute asynchronously and in parallel once all the tensors on their incoming edges becomes available.



TensorFlow Playground

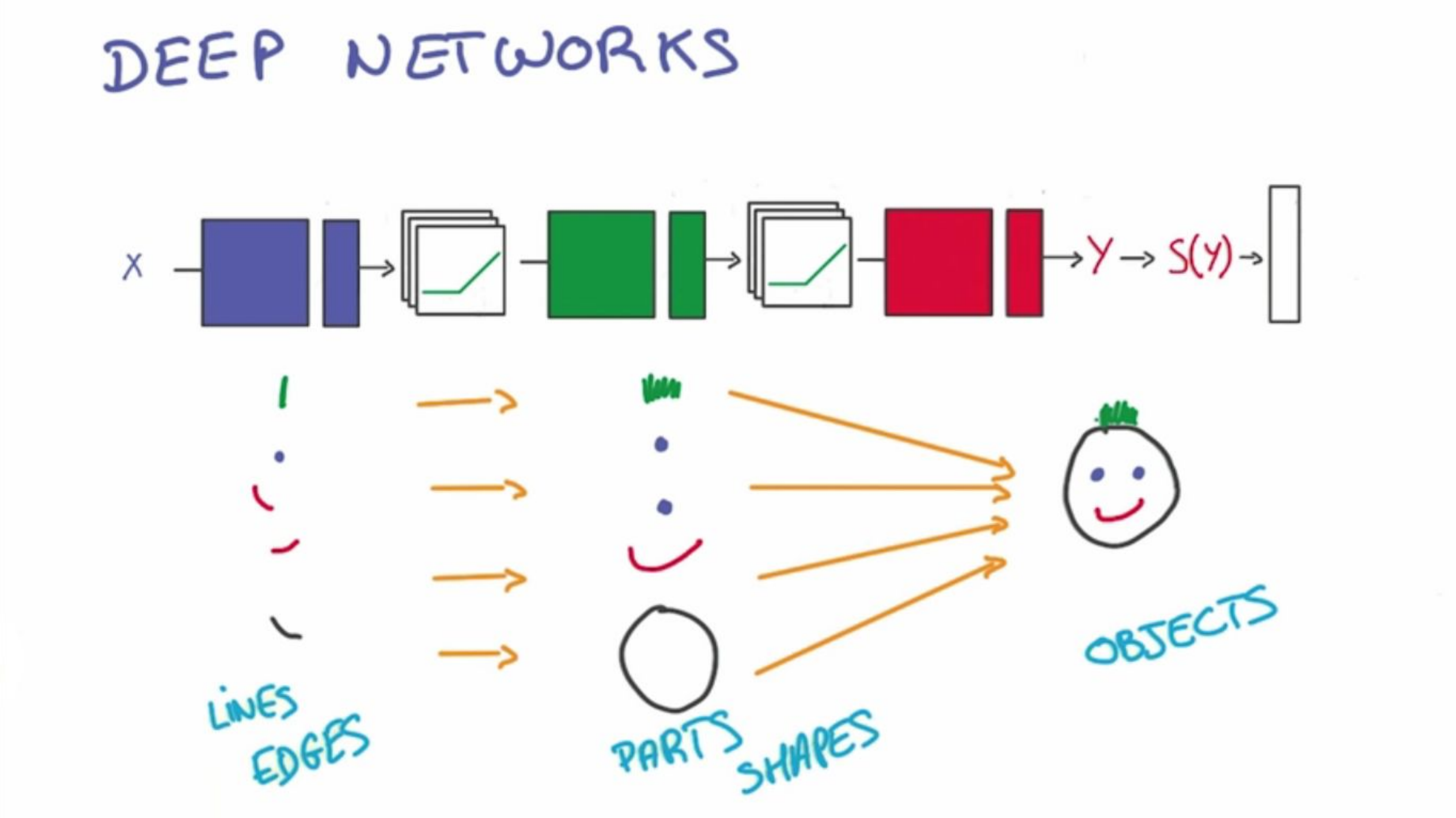
http://playground.tensorflow.org/

Back to Code: Hello Tensor

What is Neural Network?

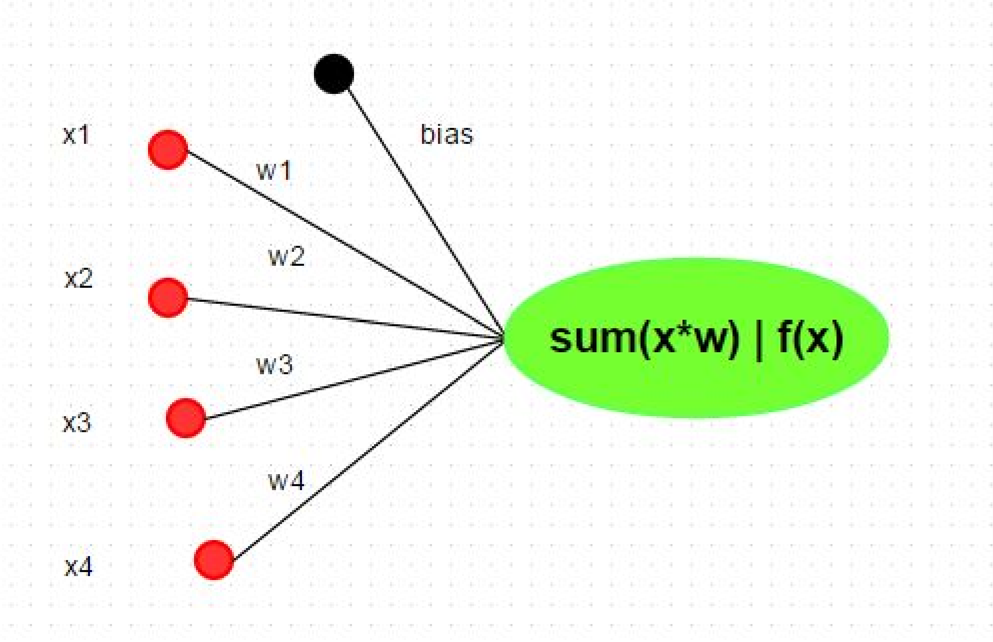
- ❑ Multiple layers
- ❑ Data propagates through layers
- ❑ Transformed by each layer

Possible Reasoning behind effectiveness of NN's



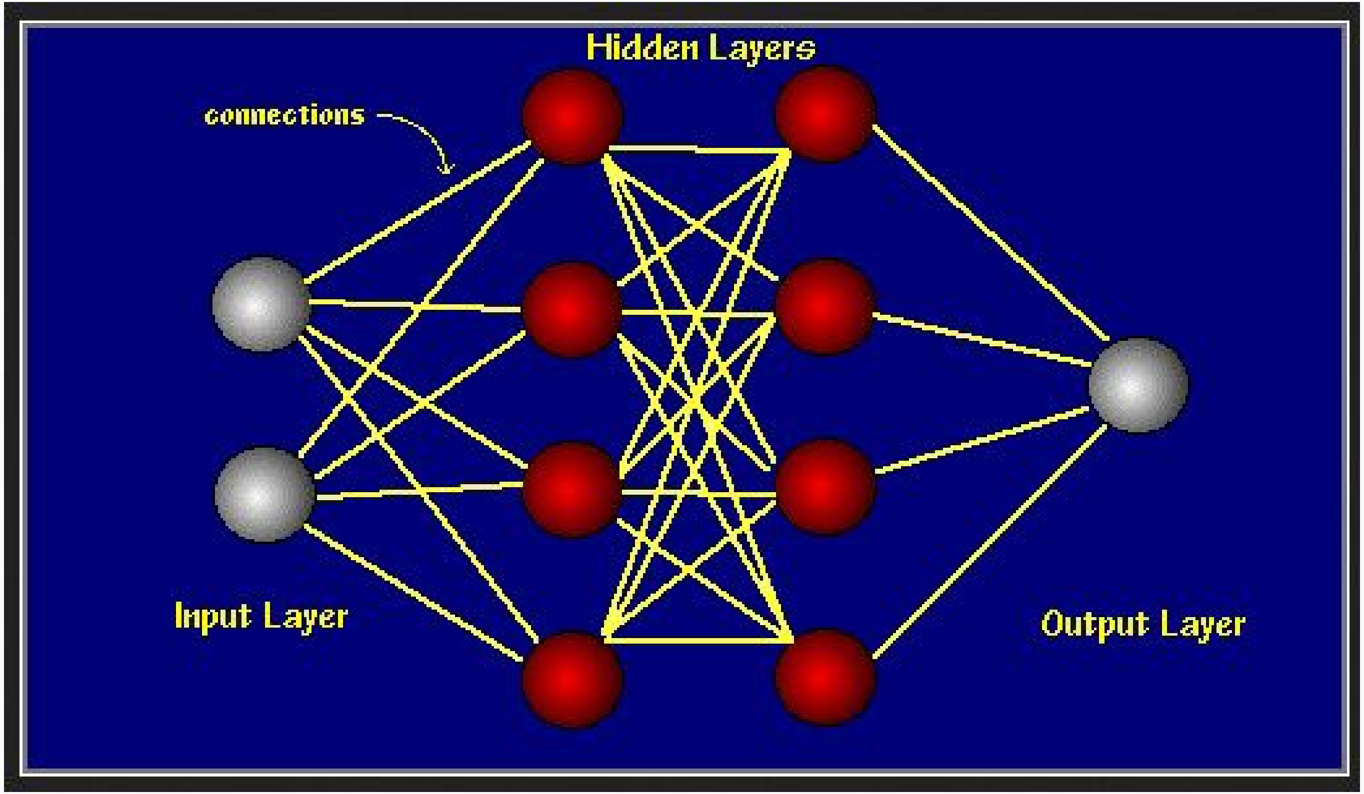
DeepLearning
udacity.com

A Perceptron: (Logistic Regression)



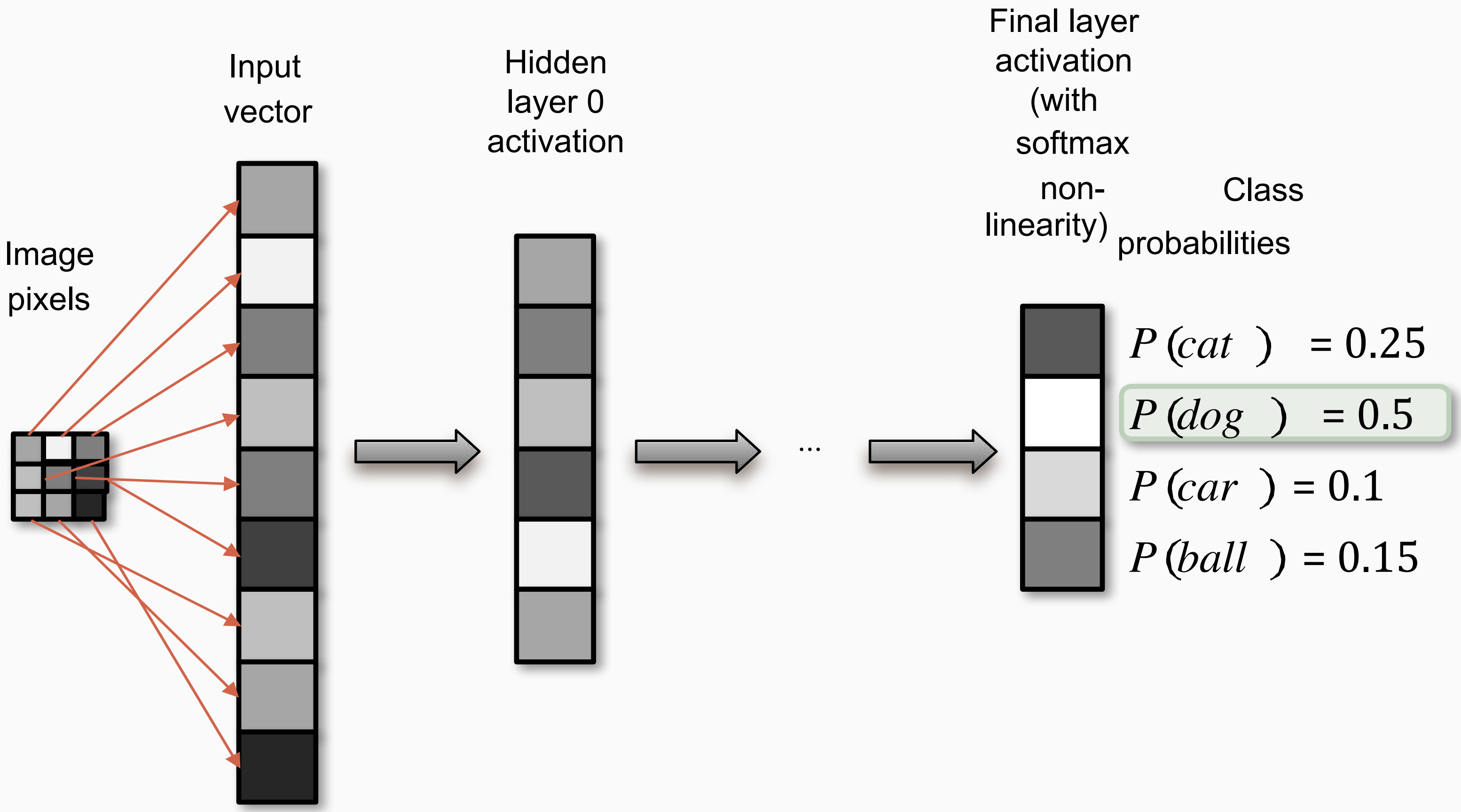
http://www.parallelr.com/wp-content/uploads/2016/02/neuron.png

Increasing The classifiers

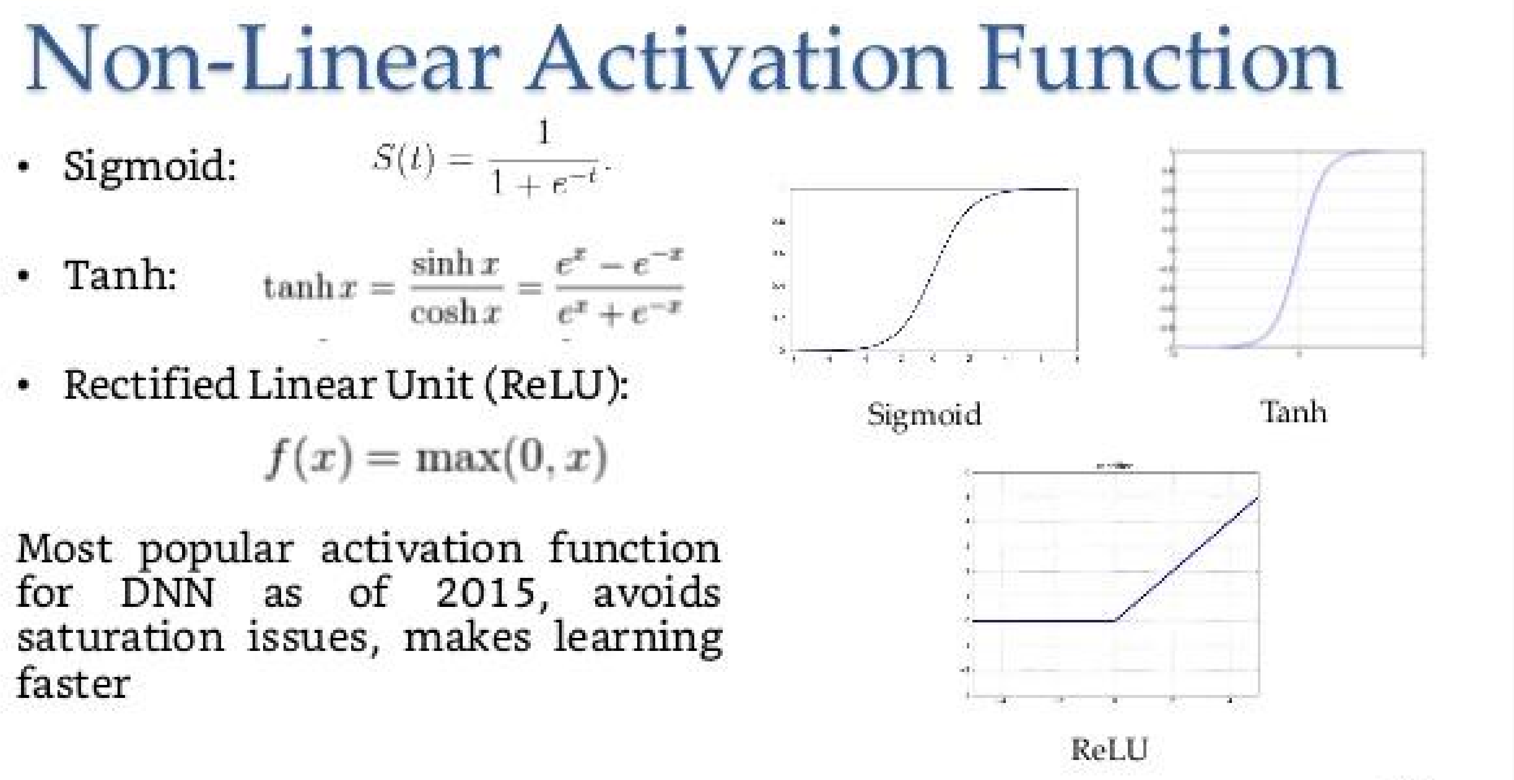


http://pages.cs.wisc.edu/~bolo/shipyard/neural/nn_schematic.gif

As a classifier



Types of Nonlinearities

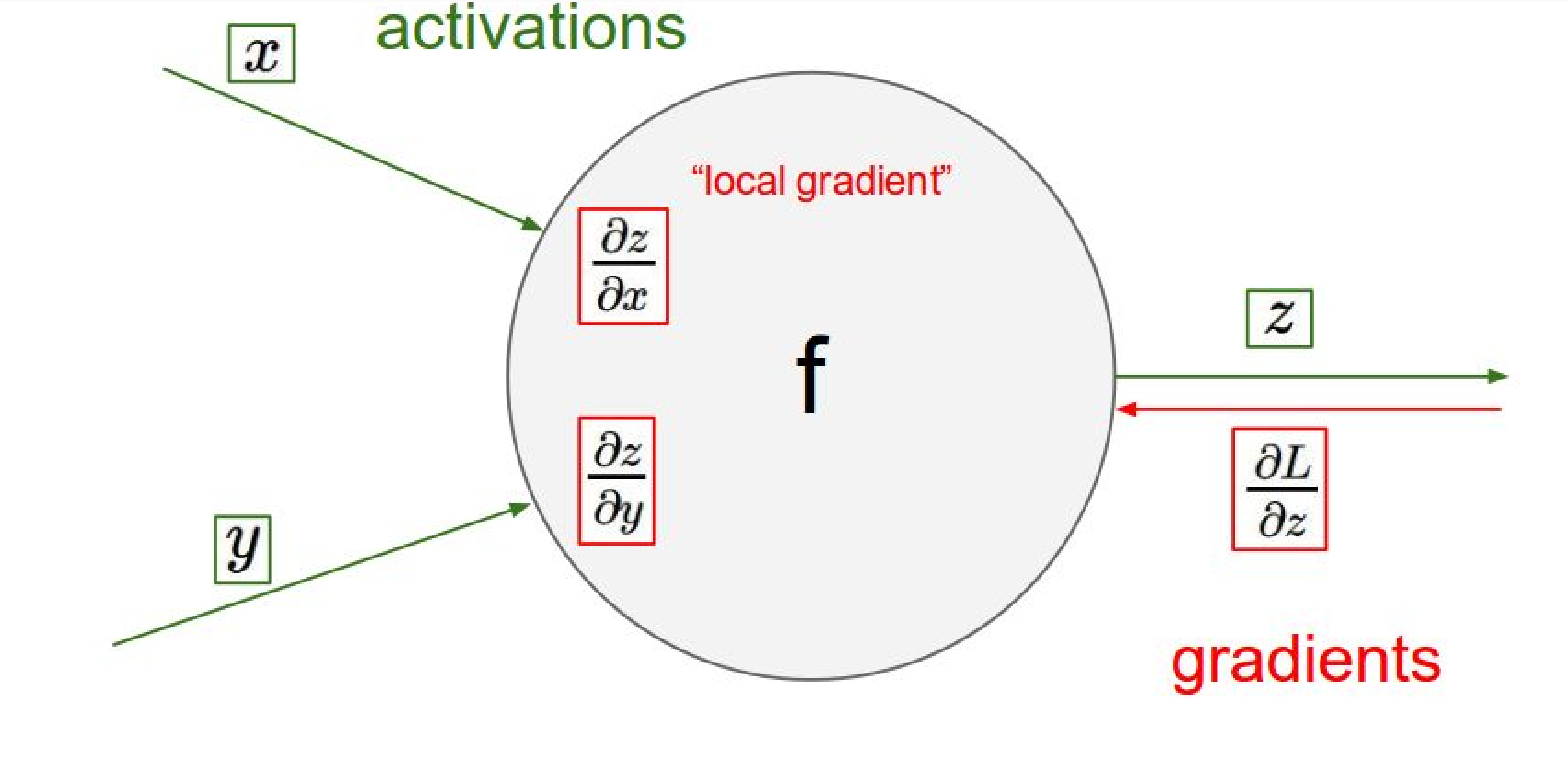


http://image.slidesharecdn.com/cnn-toupload-final-151117124948-lva1-app6892/95/convolutional-neural-networks-cnn-44-638.jpg?cb=1455889178

Training a Network

- Forward Propagation
  - Multiply Weights by inputs and add them up
  - Pass the result through a nonlinearity (ReLU, Sigmoid)
  - Repeat for next layer until the end layer
- Backward Propagation
  - Each node contributes to a certain degree of error
  - Final errors is a combination of everything
  - Disperse the error to respective nodes
  - Until you reach the starting node

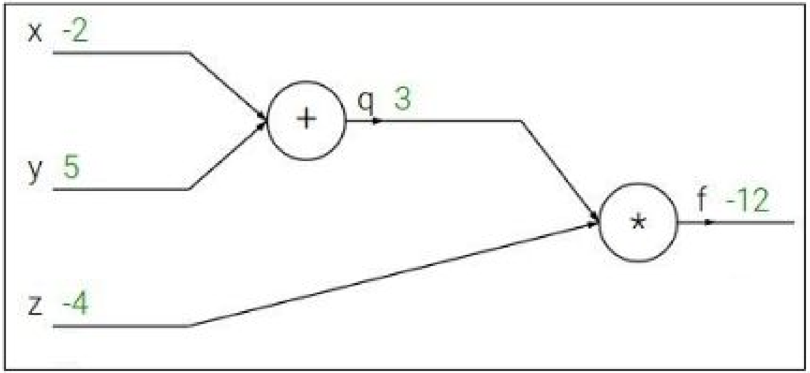
Toy Example Intuition



Toy Example

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

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Toy Example (Cont.)

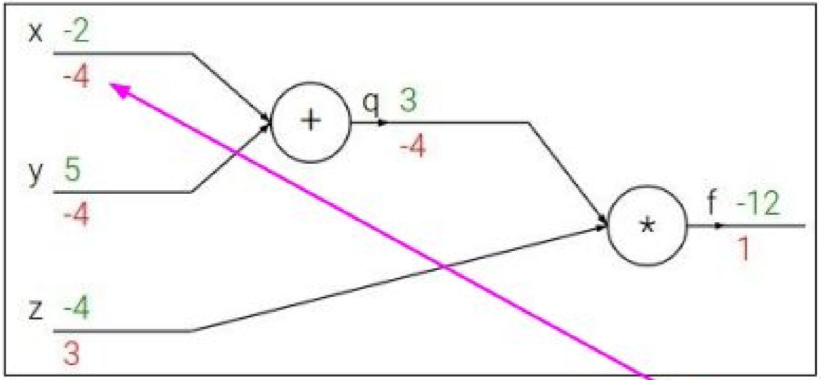
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\boxed{\frac{\partial f}{\partial x}}$$

Chain rule:

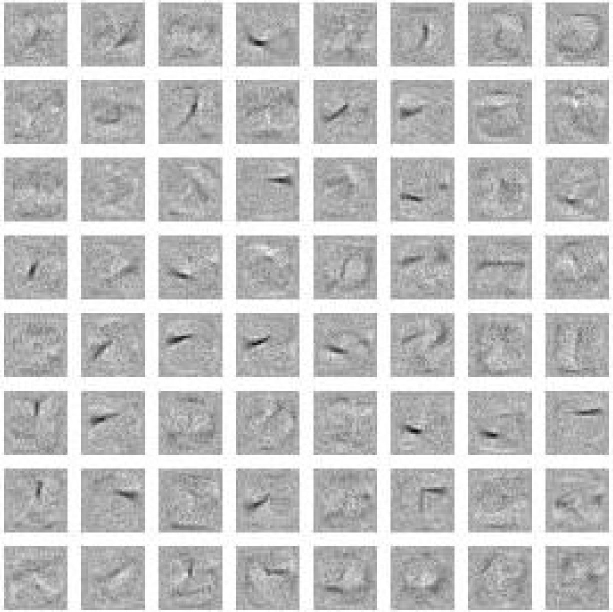
$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

Break (10 Min)

Hands On After That

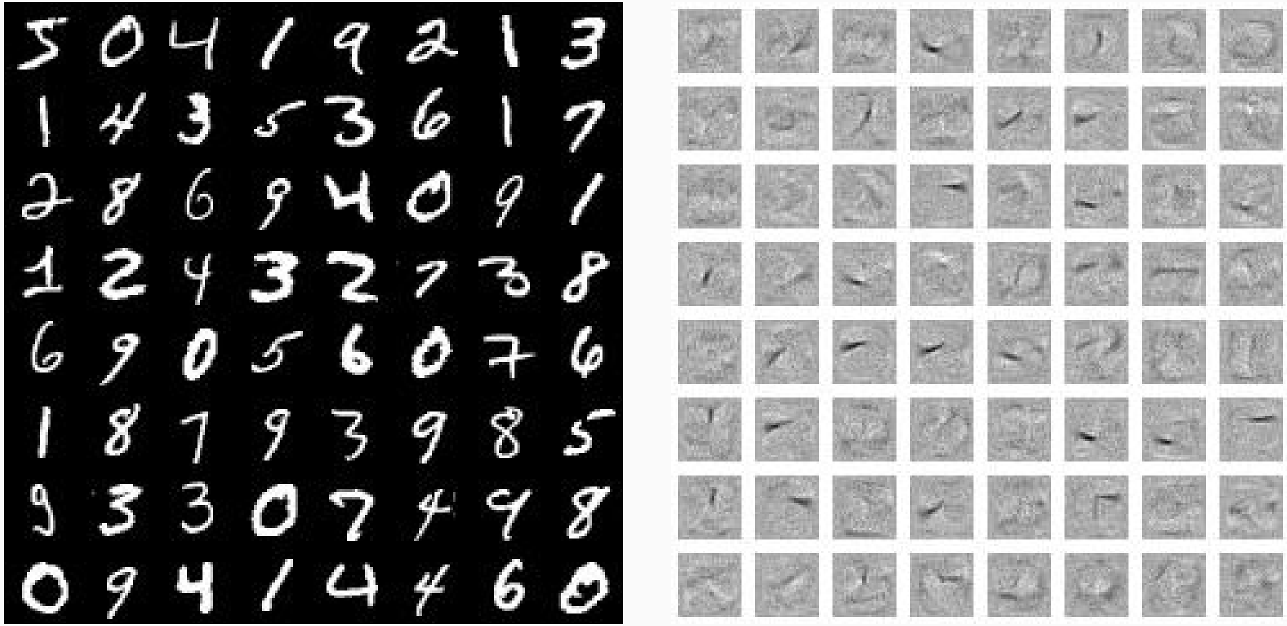
A Problem

Visualising the learned weights can be educational



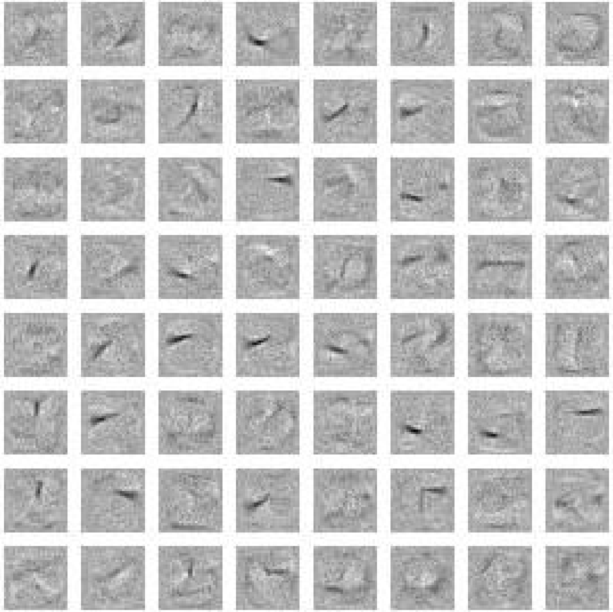
Each image visualises the weights connecting pixels to a specific unit in the first hidden layer.

Note the stroke features detected by the various units



The fully connected networks so far have a weakness:

No translation invariance; learned features are position dependent



How do you solve this?

For more general imagery:

- Requires a training set large enough to see all features in all possible positions…
- Requires network with enough units to represent this…

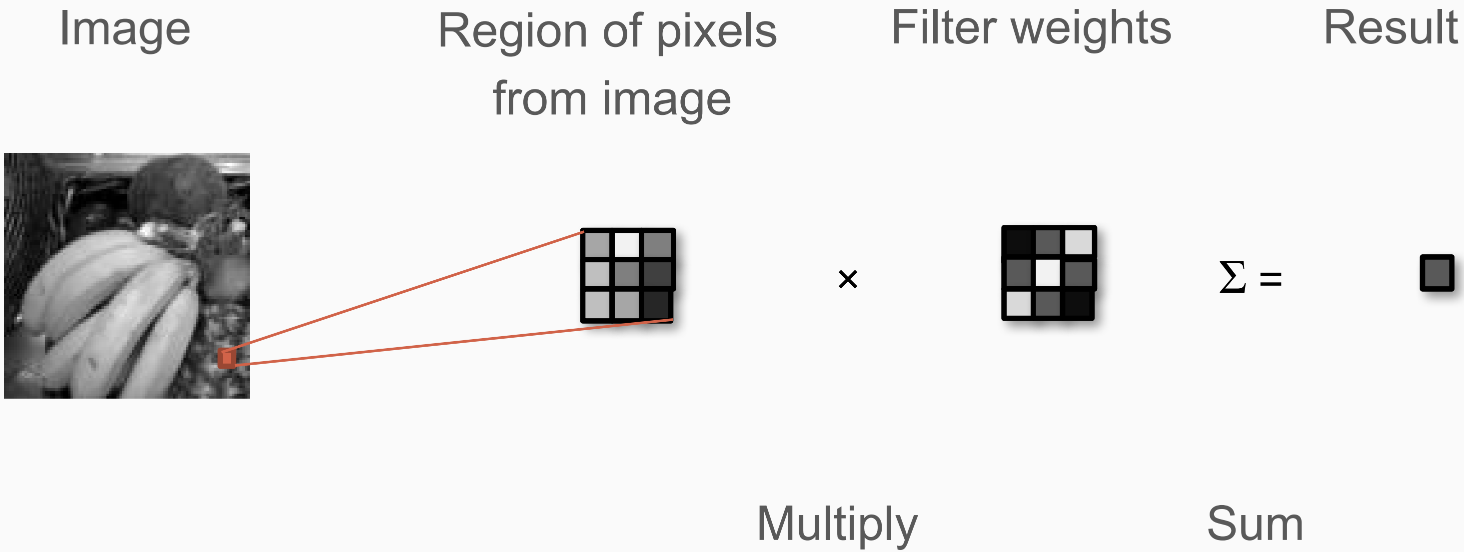
Convolutional Networks

Convolution

Often used for feature detection

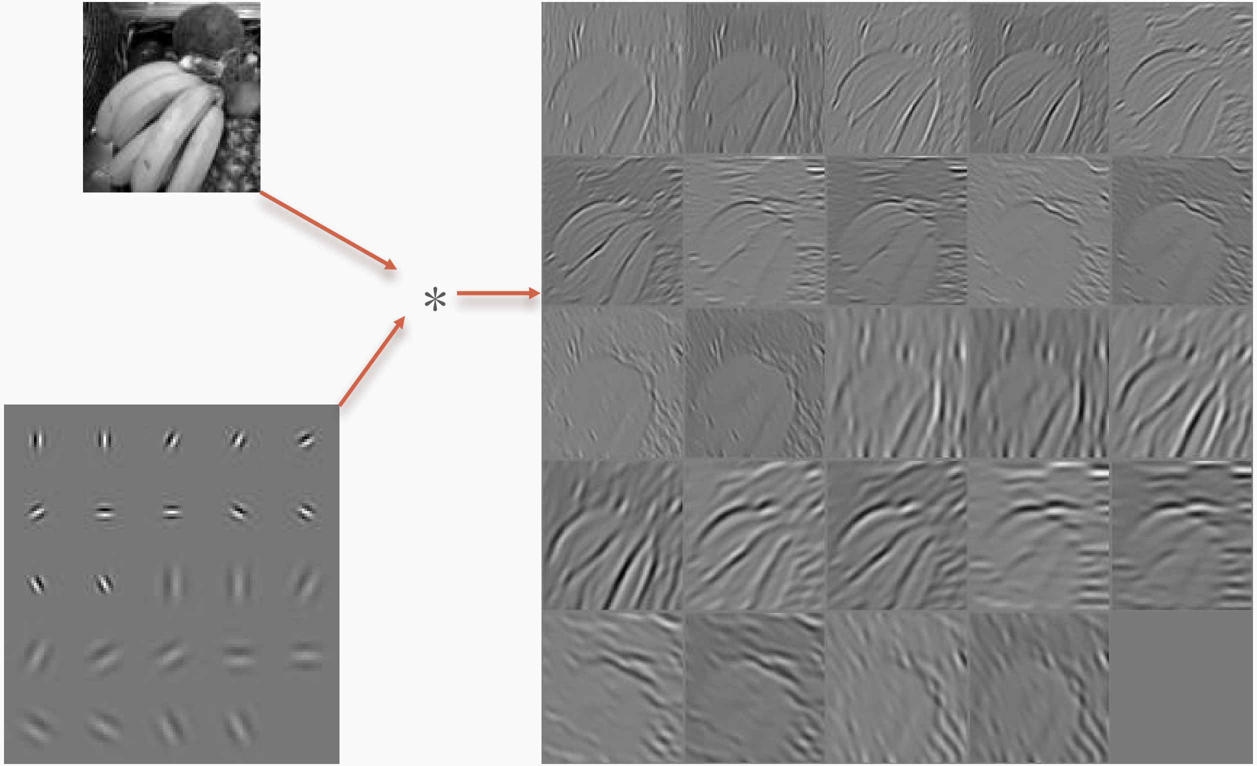
Slide a convolutional filter over an image...

Multiply image pixels by filter weights and sum



Do this for all possible positions in the image

Convolution: Gabor filters

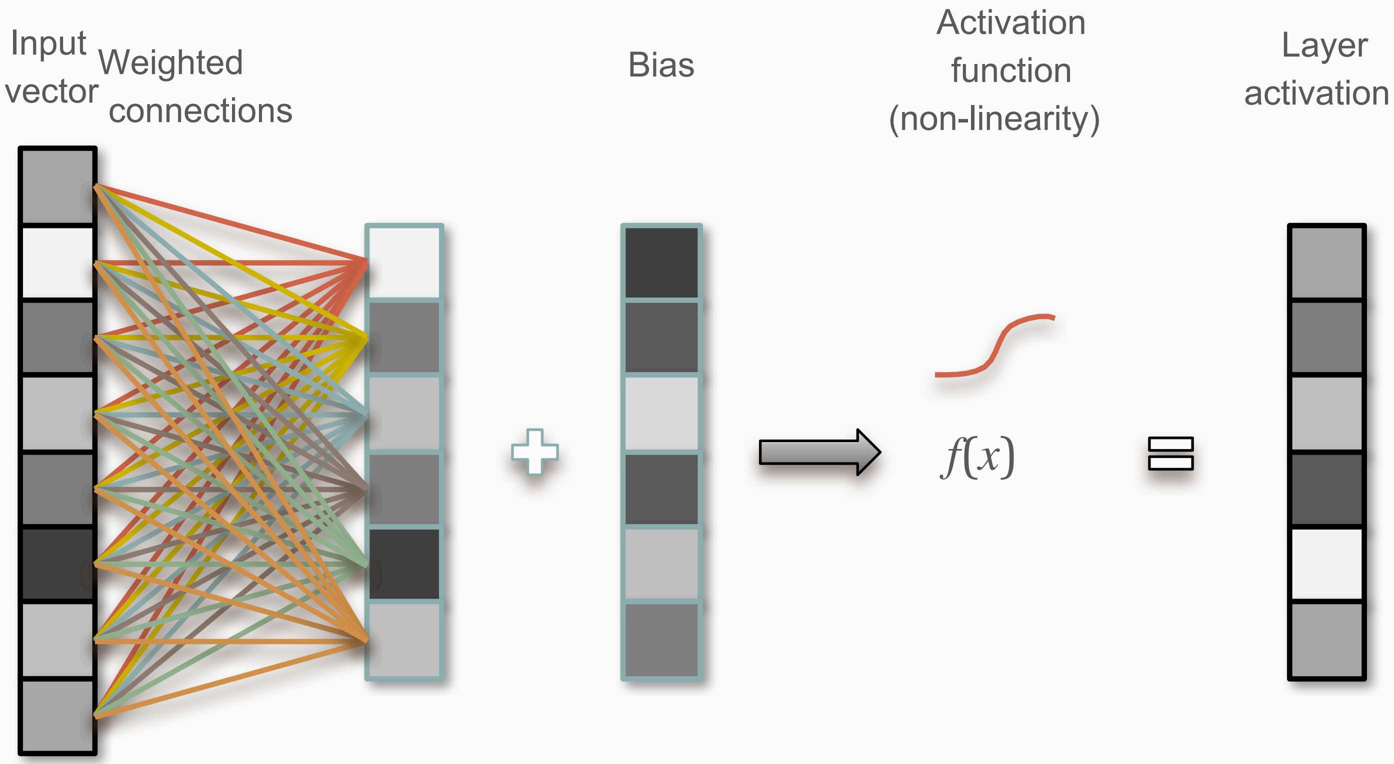


Convolution detects features in a position independent manner

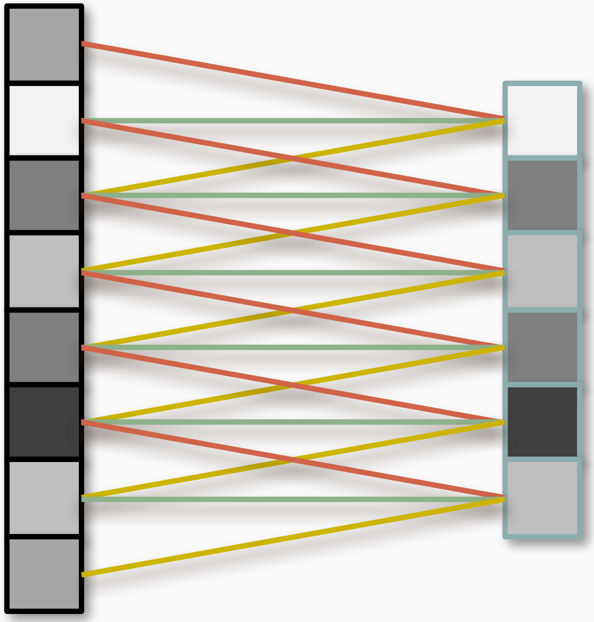
--

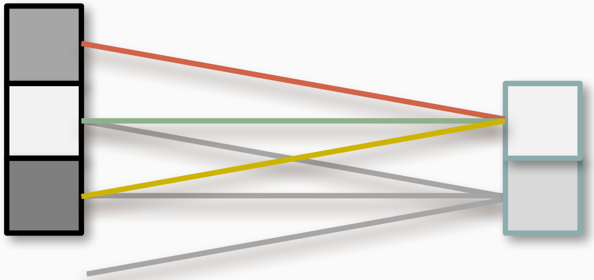
Convolutional neural networks learn position independent filters (feature detectors)

Recap: FC (fully-connected) layer



Convolutional layer



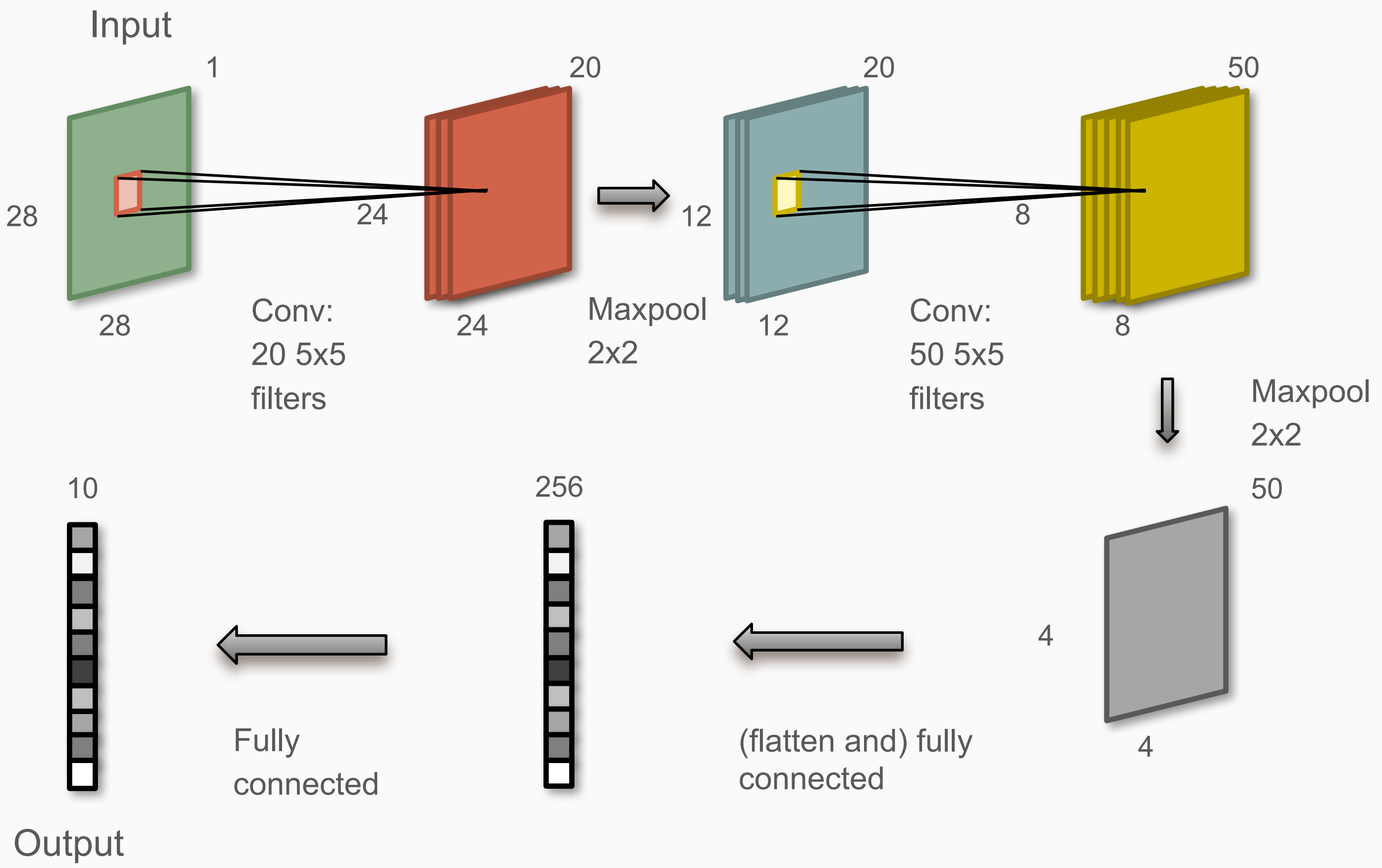


Each unit only connected to units in its neighbourhood

Example:

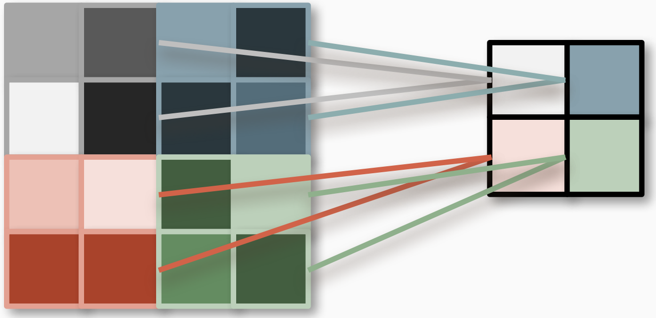
A Simplified LeNet **[LeCun95]** for MNIST digits

Convolutional layer



Max-pooling 'layer' **[Ciresan12]**

Take maximum value from each 2 x 2 pooling region (p x p) in the general case



after 300 iterations over training set:
99.21% validation accuracy

Thanks

Adapted from:

- https://github.com/Britefury
- https://github.com/nlintz/TensorFlow-Tutorials