Deep Learning and Machine Learning with TensorFlow

Trainer: Jan Idziak

TINFOTECH

Website:www.tertiarycourses.com.sg
Email: enquiry@tertiaryinfotech.com

Agenda - Day 1

Module 1 Getting Started

- What is Deep Learning
- What is Machine Learning
- What is TensorFlow
- Installing TensorFlow

Module 2 Simple Operations with TensorFlow

- Math Operations with TensorFlow
- TensorFlow Data Structure

Agenda - Day 1

Module 3 Simple Neural Networks with TensorFlow

- Loss Functions
- Logistic Regression
- Linear Regression

Module 4 Neural Networks with TensorFlow

- Activation functions
- Single Layer Perceptron

Agenda - Day 1

Module 5 Tensorboard

- Creating first tensor board
- Results lookup

Agenda - Day 2

Module 6 Multi Layer Perceptron

- What does MLP consists of?
- Implementation of multilayer perceptron

Module 7 CNN - Convolutional Neural Networks

- Convolutional Layers
- Pooling Layers
- Dropout - finding multiple pathways

Agenda - Day 2

Module 8 RNN - Recurrent Neural Networks

- Model of neural network
- LSTM - long short term memory cell
- Comparison with other models

Module 9 Keras - simplifying interface

- CNN for MNIST data
- RNN for IMDB movie reviews dataset Keras

Exercise Files

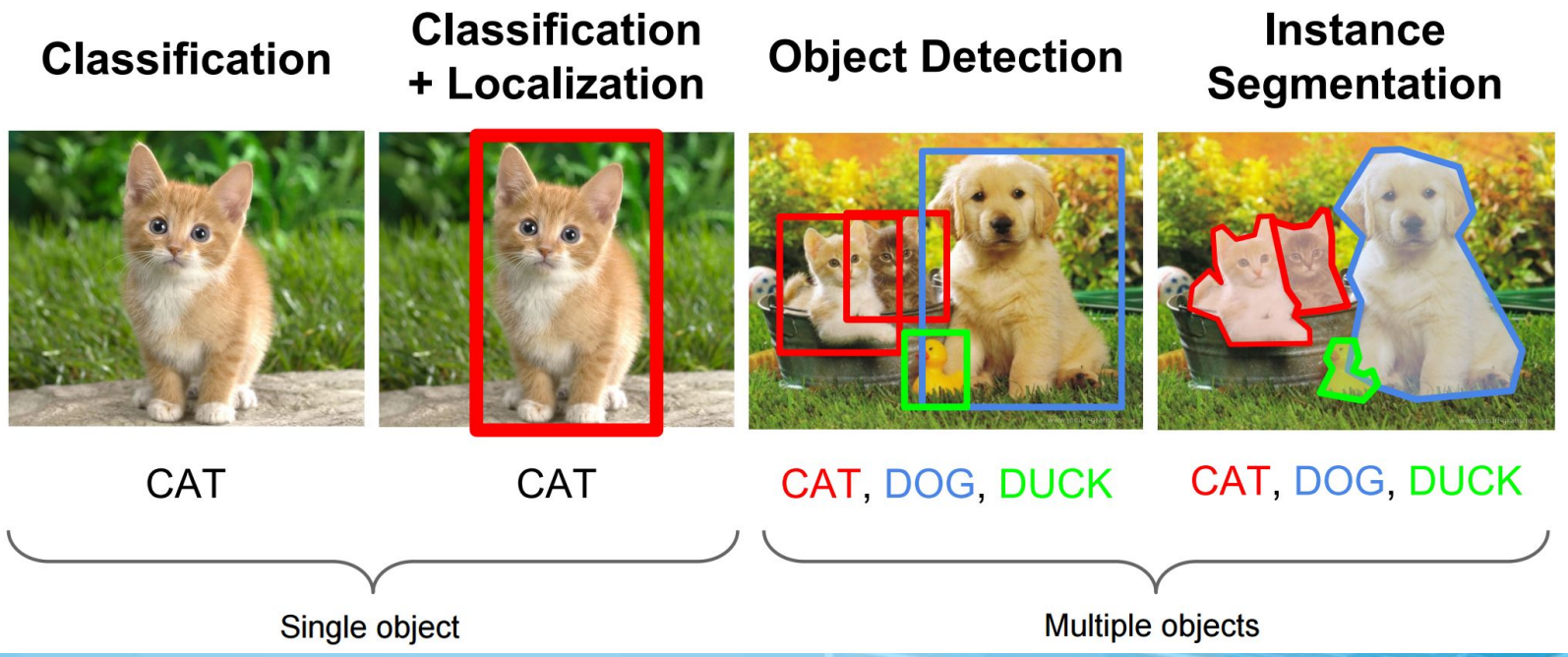
Please download the exercise files from

https://github.com/tertiarycourses/TensorflowTraining

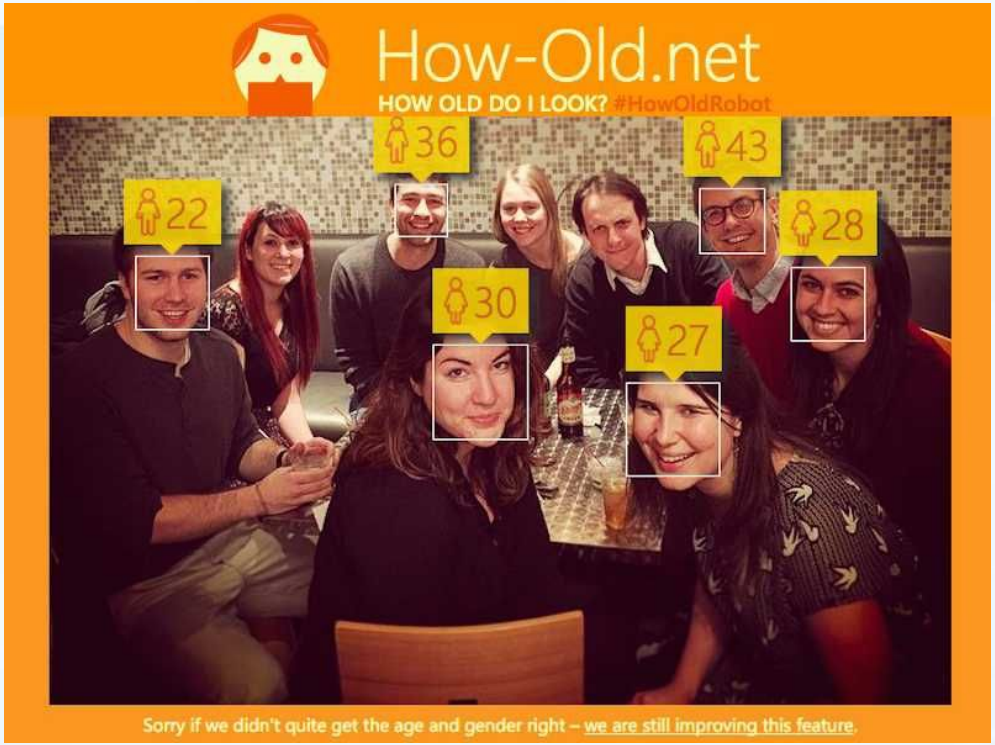
Module 1
Getting Started

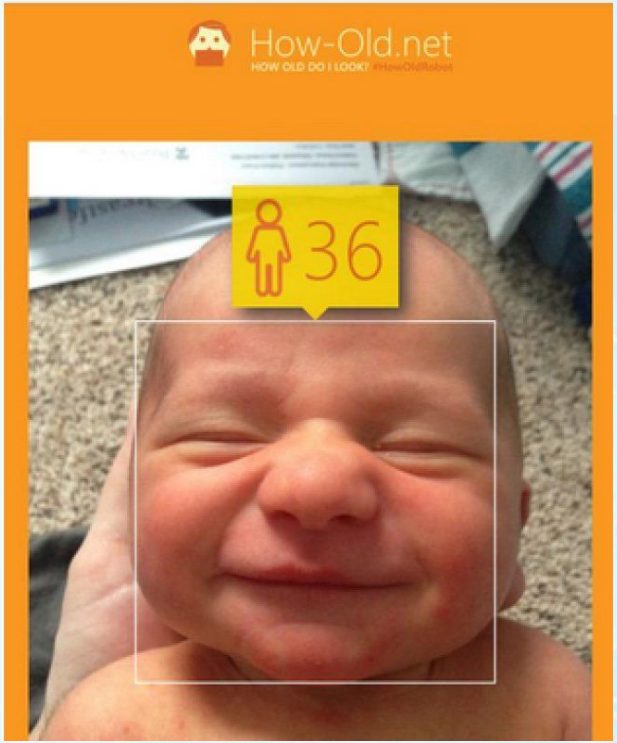
Why Machine Learning?

Computer Vision Tasks

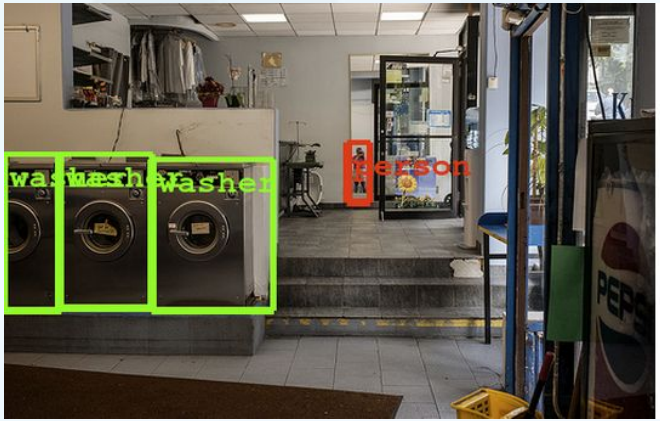


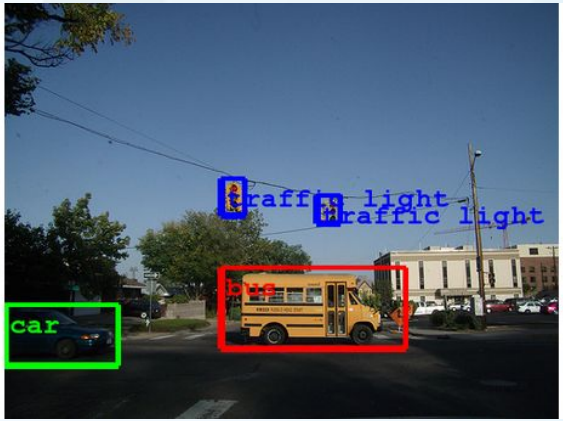
Why Machine Learning?

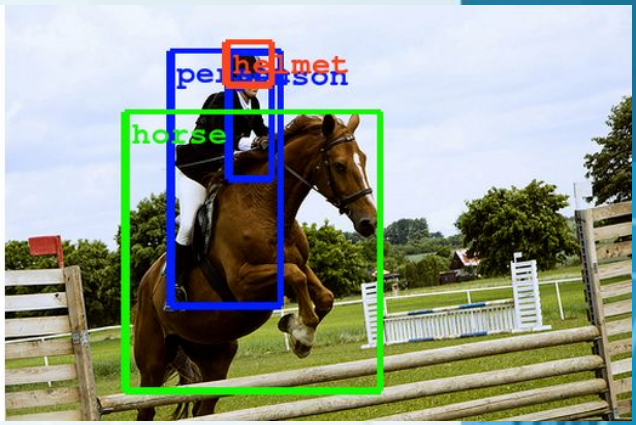




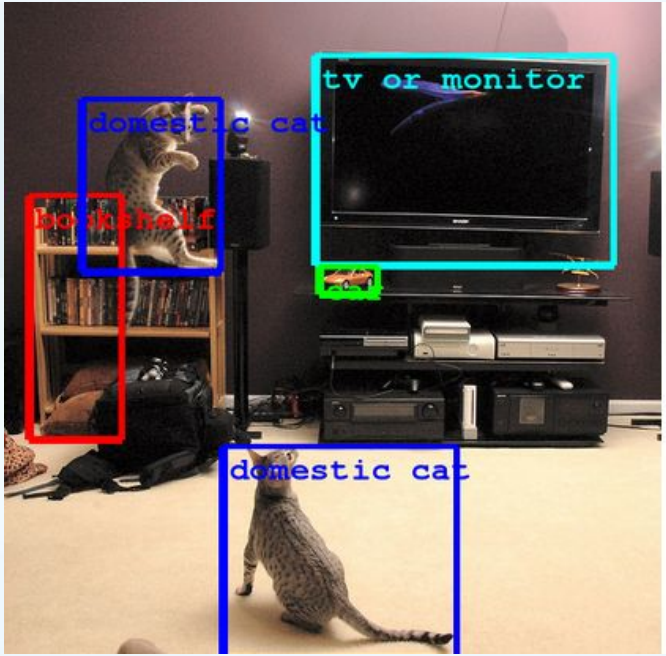
Why Machine Learning?

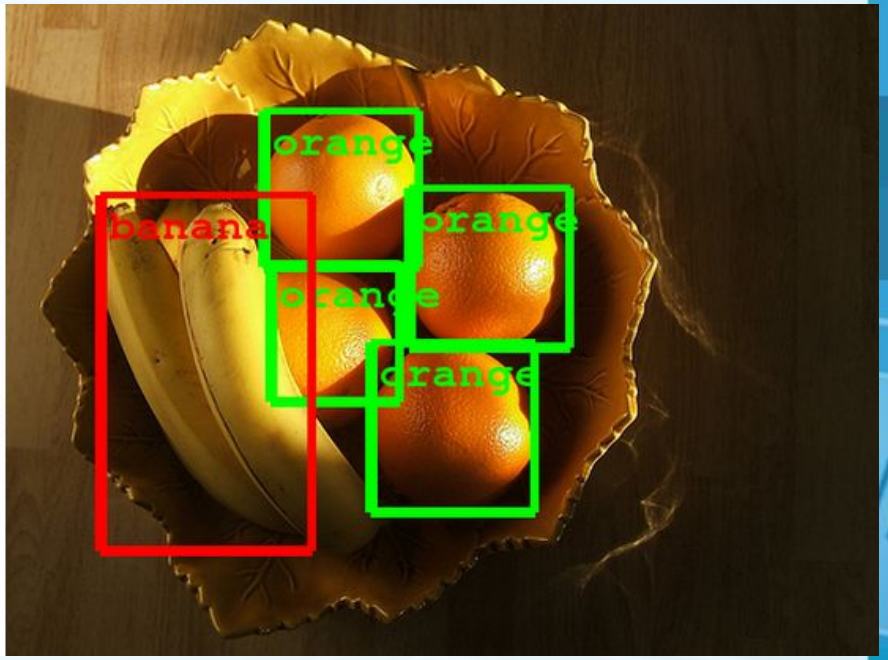






Why Machine Learning?





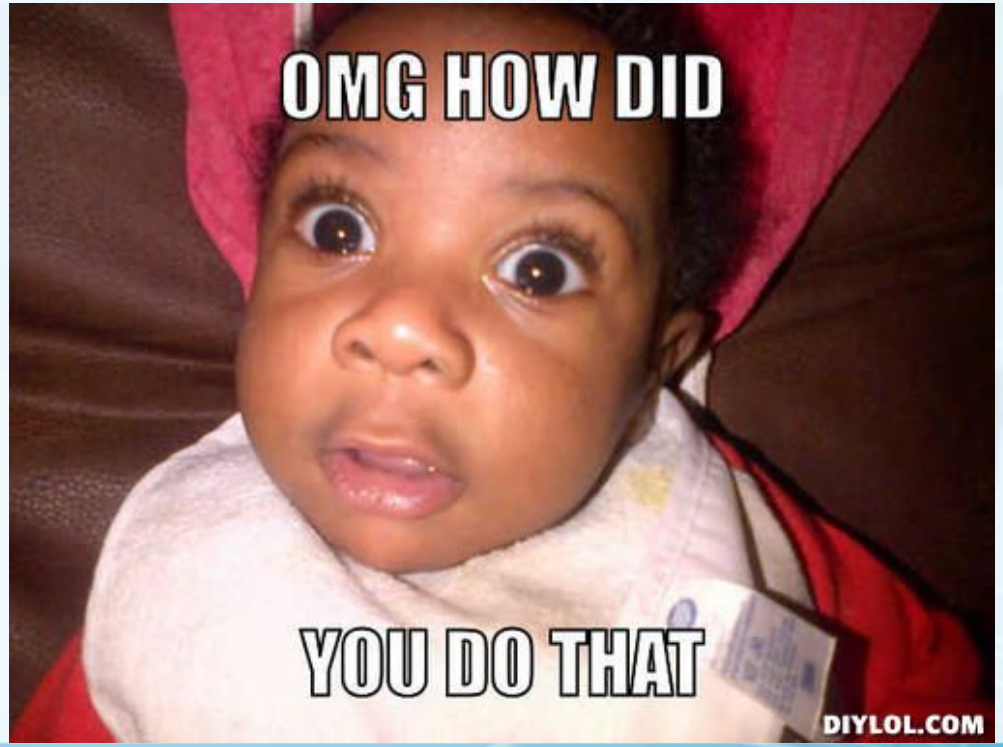
Why to use Machine Learning?

PLAYS : EPOCH 338

LARGER NETWORK...

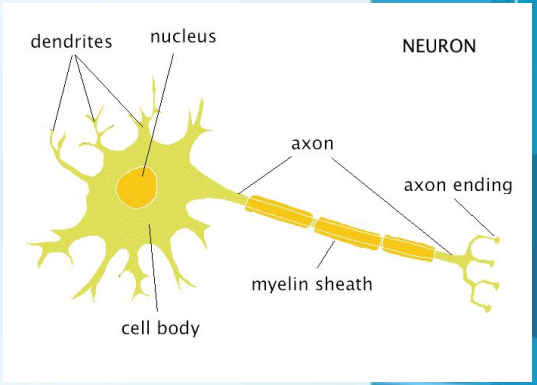
```
DEDENIUS         Why shoulmeying to to wife,
        And thou say: and wall you teading for
        that struke you down as sweet one.
        With be more bornow, bly unjout on the account:
        I duked you did four conlian unfortuned drausing-
        to sicgia stranss, or not sleepplins his arms
        Gentlemen? as write lord; gave sold.

AENEMUUNS        Met that will knop unhian, where ever have
        of the keep his jangst?icks he I love hide,
        Jach heard which offen, sir!'

        [Exit PATIIUS, MARGARUS arr       [Enter CLOTHUR]
```

Why Machine Learning?



Why Neural Networks?

- The human brain can be considered to be one of the best processors. (Estimated to contain ~10^{11} neurons.)
- Studies show that our brain can process the equivalent of ~20Mb/sec just through the optical nerves.
- If we can copy this design, maybe we can solve the “hard for a computer – easy for humans” problems.

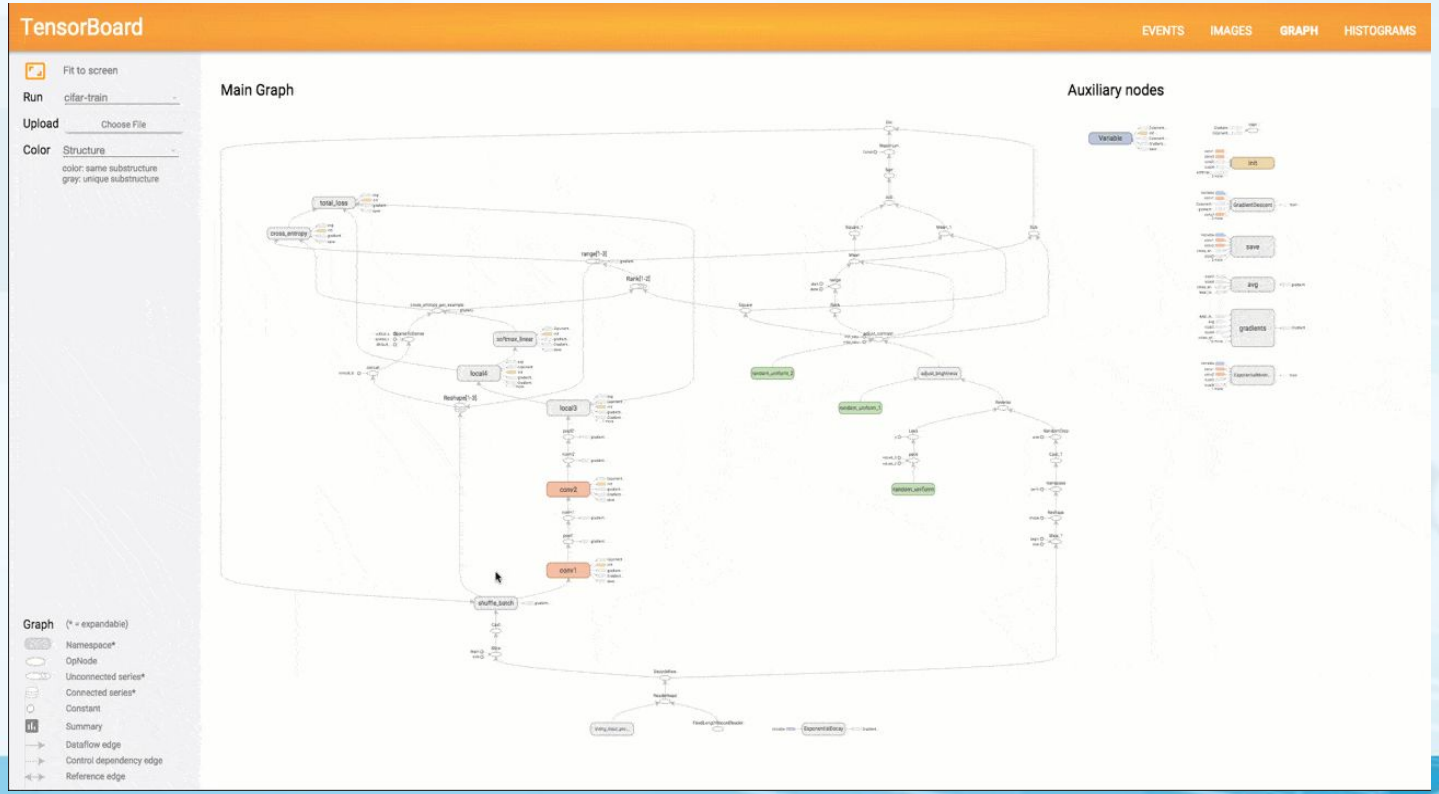


- Speech recognition
- Facial identification
- Reading emotions
- Recognizing images
- Sentiment analysis
- Driving a vehicle
- Disease diagnosis

What is TensorFlow?

- An open source framework for creating computational graphs.
- Computational graphs are actually extremely flexible.
  - Linear Regression (regular, multiple, logistic, lasso, ridge, elasticnet,...)
  - Support Vector Machines (any kernel)
  - Nearest Neighbor methods
  - Neural Networks
  - ODE/PDE Solvers

What is TensorFlow?



TensorFlow vs Scikit Learn

- Scikit-learn is a higher-level library that includes implementations of several machine learning algorithms, so you can define a model object in a single line or a few lines of code, then use it to fit a set of points or predict a value.
- TensorFlow is a low-level library that allows you to build machine learning models (and other computations) using a set of simple operators, like "add", "matmul", "concat", etc.

Tensorflow vs Numpy

- NumPy is the fundamental package for scientific computing with Python
- TensorFlow™ is an open source software library for numerical computation using data flow graphs

What is TensorFlow

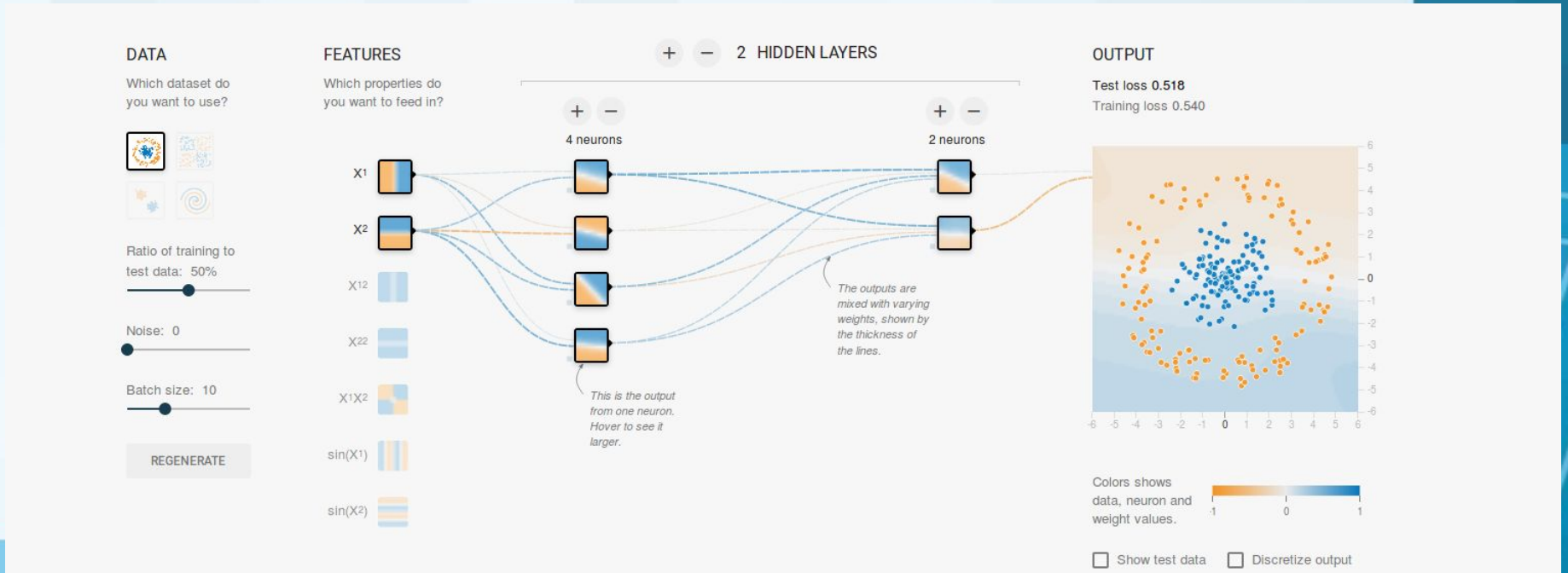
- Tensors: n-dimensional arrays
  - Vector: 1-D tensor
  - Matrix: 2-D tensor
- Deep learning process are flows of tensors
  - A sequence of tensor operations may represent ML algorithms

Other NN Frameworks

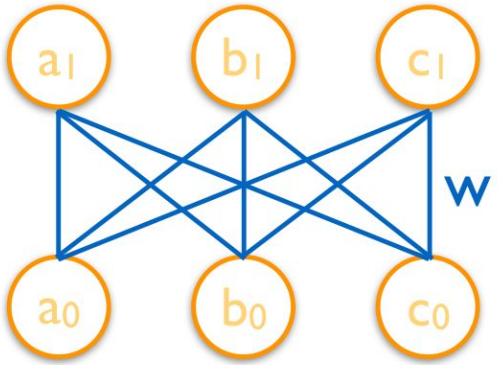
- Theano
- Torch
- Caffe
- Nervana
- H2O
- DL4J
- Keras

TensorFlow Playground

http://playground.tensorflow.org/



Math Operations



$a_1 = a_0 w_{a,a} + b_0 w_{b,a} + c_0 w_{c,a}$
$b_1 = a_0 w_{a,b} + b_0 w_{b,b} + c_0 w_{c,b}$
$c_1 = a_0 w_{a,c} + b_0 w_{b,c} + c_0 w_{c,c}$
Apply relu(...) on a_1 , b_1 , c_1

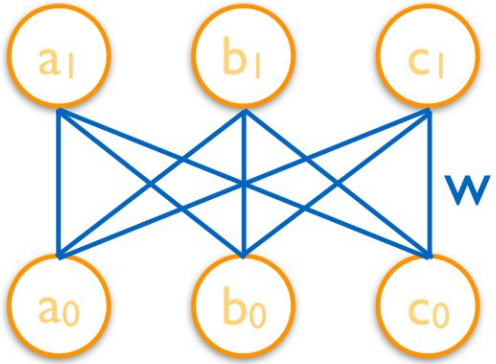
Slower approach:

Per-neuron operation

More efficient approach:

Matrix operation

Math Operations

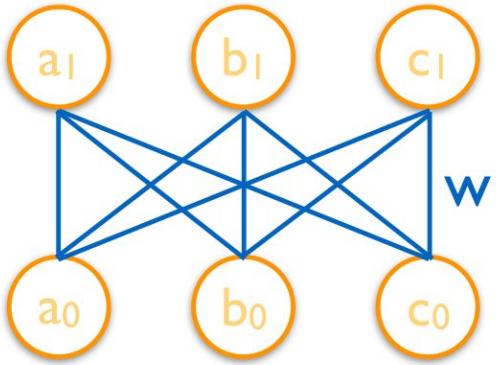


Matrix operation

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

$$a_1 = \text{relu}(a_1)$$
$$b_1 = \text{relu}(b_1)$$
$$c_1 = \text{relu}(c_1)$$

Matrix Operations



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

$$\begin{bmatrix} a_1 \end{bmatrix} = \text{relu}\left(\begin{bmatrix} a_1 \end{bmatrix} \right)$$

$$\begin{bmatrix} b_1 \end{bmatrix} = \text{relu}\left(\begin{bmatrix} b_1 \end{bmatrix} \right)$$

$$\begin{bmatrix} c_1 \end{bmatrix} = \text{relu}\left(\begin{bmatrix} c_1 \end{bmatrix} \right)$$

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

Matrix Operationsw

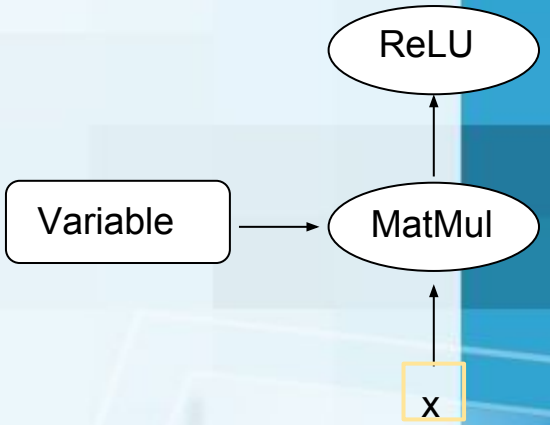
```
import tensorflow as tf
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
```

Relu Operations

```
import tensorflow as tf
sess = tf.Session()
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
result = sess.run(relu_out)
```

Variables

```
import tensorflow as tf
sess = tf.Session()
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
result = sess.run(relu_out)
```



Variable Initializer

```
import tensorflow as tf
sess = tf.Session()
x = tf.placeholder(“float”, [1,3])
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
sess.run(tf.global_variables_initializer())
result = sess.run(relu_out)
```

Placeholder

```
import tensorflow as tf
sess = tf.Session()
x = tf.placeholder(“float”, [1,3])
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
sess.run(tf.global_variables_initializer())
result = sess.run(relu_out)
```

Feed_Dict

```
import tensorflow as tf
import numpy as np
sess = tf.Session()
x = tf.placeholder(“float”, [1,3])
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
sess.run(tf.global_variables_initializer())
result = sess.run(softmax, feed_dict ={x:np.array([[1.0, 2.0, 3.0]])})
```

Softmax

```
import tensorflow as tf
import numpy as np
sess = tf.Session()
x = tf.placeholder(“float”, [1,3])
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
softmax = tf.nn.softmax(relu_out)
sess.run(tf.global_variables_initializer())
result = sess.run(softmax, feed_dict ={x:np.array[[1.0, 2.0, 3.0]])
```

Prediction

```
import tensorflow as tf
import numpy as np
sess = tf.Session()
x = tf.placeholder(“float”, [1,3])
w = tf.Variable(tf.random_normal([3, 3]), name='w')
y = tf.matmul(x, w)
relu_out = tf.nn.relu(y)
softmax = tf.nn.softmax(relu_out)
sess.run(tf.global_variables_initializer())
answer = np.array([[0.0, 1.0, 0.0]])
result = answer - sess.run(softmax, feed_dict ={x:np.array([[1.0, 2.0, 3.0]])})
```

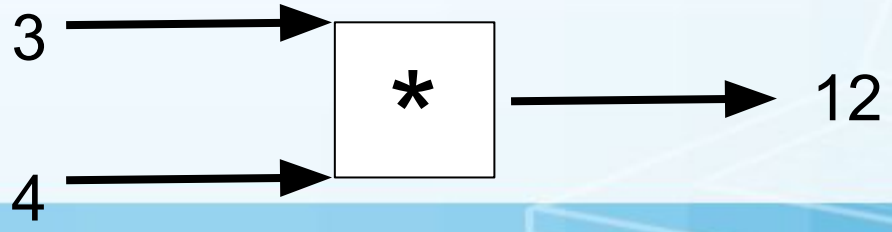
Module 2
Basic Operations

TensorFlow Operations

Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, ...
Array operations	Concat, Slice, Split, Constant, Rank, Shape, Shuffle, ...
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, ...
Stateful operations	Variable, Assign, AssignAdd, ...
Neural-net building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, ...
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

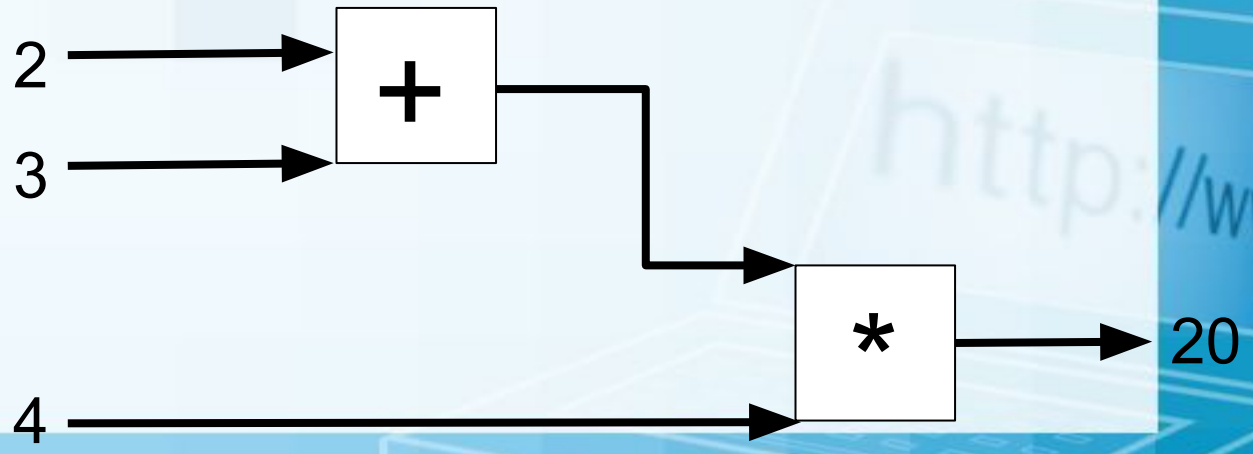
Single Gate Operation

- How can we change the inputs to decrease the output?
- The intuitive answer is to decrease 3 and/or decrease 4.
- Mathematically, the answer is to use the derivative…
  - We know if we decrease the (3) input by 1, the output goes up by a total of 4. (4*2=8)



Multiple Gate Operations

- F(x, y, z) = (x + y) * z
- If G(a, b) = a + b, then F(x,y,z) = G(x, y) *
- Mathematically, we will use the “chain rule” to get the



Math Operations

```
#Simple daffinition of the tensor, constant, and variable
tensor = tf.zeros([1, 10])
print(tensor)
constant = tf.constant([2, 3])
print(constant)
variable = tf.Variable(tf.zeros([1, 10]))
print(variable)
```

Challenge

- Create placeholder x of a shape (2, 2)
- y = x**2 + x
- Create random matrix of a shape (2, 2)
- Feed x into placeholder
- Print y as an output

Print results using single initializer

Challenge - Hint

```
x = tf.placeholder(tf.float32, shape=(2, 2))
#x**2 + x
y = tf.add(tf.multiply(x, x), x)
rand_array_2 = np.random.rand(2, 2)
print(sess.run(y, feed_dict ={x: rand_array_2}))
```

Declaration of Elements

```
#Identity
identity_matrix = tf.diag(np.ones(3))
#Constant
const_matrix = tf.fill([2, 3], -1.0)
#Random Uniform
uniform_matrix = tf.random_uniform([3,2])
#Convert numpy to matrix
converted_matrix =  tf.convert_to_tensor(np.array([[1., 2., 3.], [-3., -7., -1.], [0., 5., -2.]]))
```

Math Operations

```
#Add (substract) two martices (tensors)
print(sess.run(identity_matrix + identity_matrix))
print(sess.run(tf.add(identity_matrix, identity_matrix)))
print(sess.run(tf.sub(identity_matrix, identity_matrix)))
#Matrix multiplication and transpose
uni_times_const = tf.matmul(uniform_matrix, const_matrix)
print(sess.run(tf.transpose(uni_times_const)))
#Inverse
two_identity = tf.add(identity_matrix, identity_matrix)
print(sess.run(tf.matrix_inverse(two_identity)))
```

Math Operations

```
#Cross product
print(sess.run(tf.cross([1, 0, 0], [0, 1, 0])))
#Square, squareroot
print(sess.run(tf.square(2)))
#Trigonometric
print(sess.run(tf.sin(3.1416)))
print(sess.run(tf.tan(3.1416)))
#Equal
print(sess.run(tf.equal(tf.cos(3.1416),-1.0)))
```

Math Operations

```
##Helper for exercises
#For loop
for i in range(15):
    print(i)

#Definition of function
def square_plus_two(x):
    return x**2 + 2
```

Random Numbers

- tf.random_normal
- tf.truncated_normal
- tf.random_uniform
- tf.random_shuffle
- tf.random_crop
- tf.multinomial
- tf.random_gamma
- tf.set_random_seed

Challenge

- Create constant_matrix with dim n_row = 3, ncol = 2
- Create rn_matrix with random_normal values with dim
- Create diag_matrix dim 3x3 with 4 on the diagonal
- Multiply constant matrix and rn_matrix
- Add diag_matrix
- Print results

Challenge - Hint

```
const_matrix = tf.fill([2, 3], 50.0)
print(sess.run(const_matrix))
```

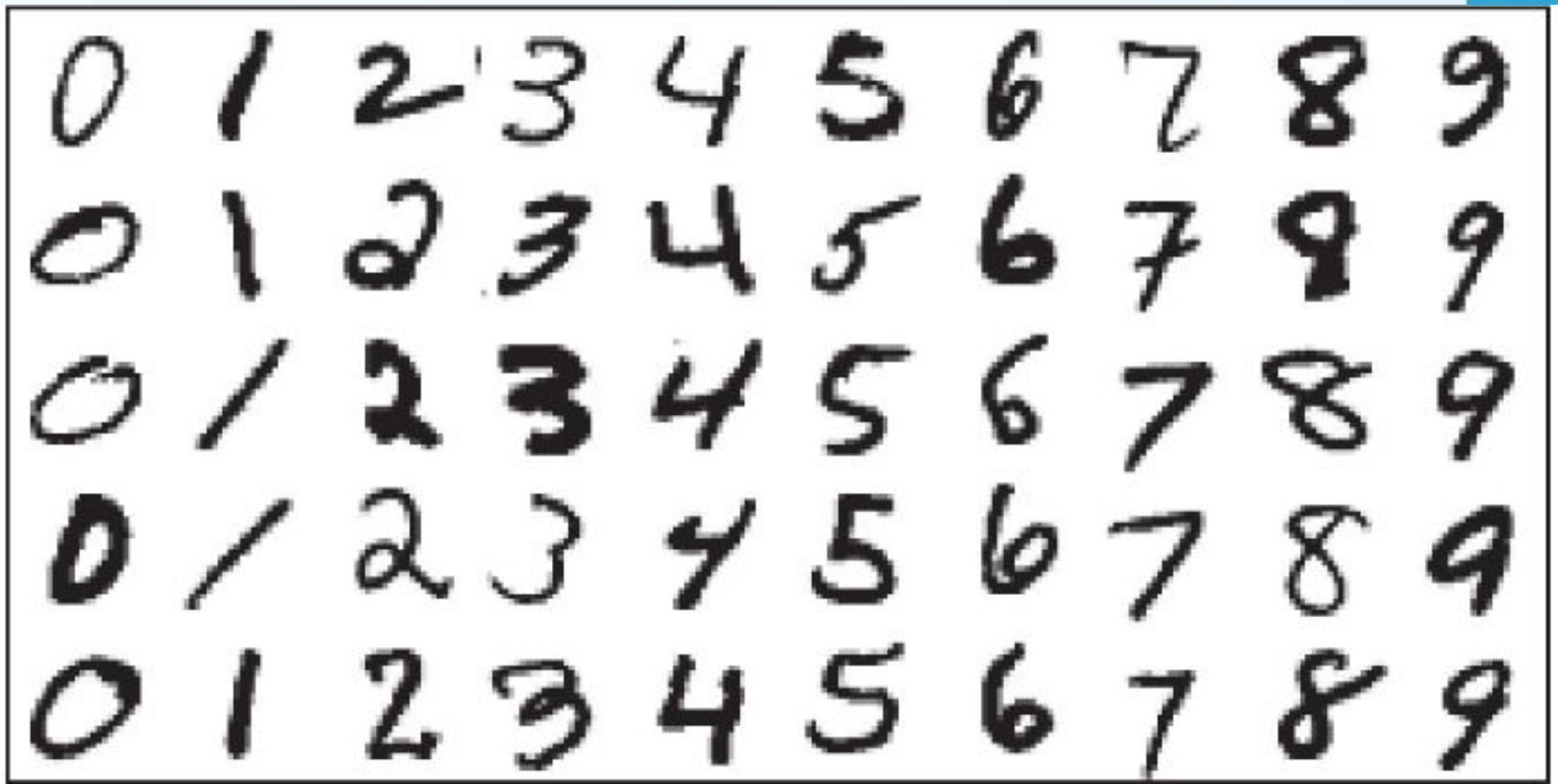
```
nrow = 2, n_col = 3
normal_matrix = tf.random_normal([3,2])
print(sess.run(normal_matrix))
```

Challenge - Hint

```
diag_matrix = 4*tf.diag(np.ones(3))
print(sess.run(diag_matrix))

mm = tf.matmul(const_matrix, normal_matrix)
print(sess.run(mm))
```

Datasets

MNIST Handwritten Digits



MNIST Datasets

```
#MNIST Handwriting Data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/",
  one_hot=True)
print(len(mnist.train.images))
print(len(mnist.test.images))
print(len(mnist.validation.images))
print(mnist.train.labels[1,:])
```

Iris Flowers

Iris Flowers Dataset

```
#Iris data
iris = datasets.load_iris()
print(len((iris.data)))
print(set(iris.target))
print(iris.data[0])
print(iris.feature_names)
```

Movie Review Data

http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz

Sentiment Analysis

A Baseline Algorithm

Movie Review Dataset

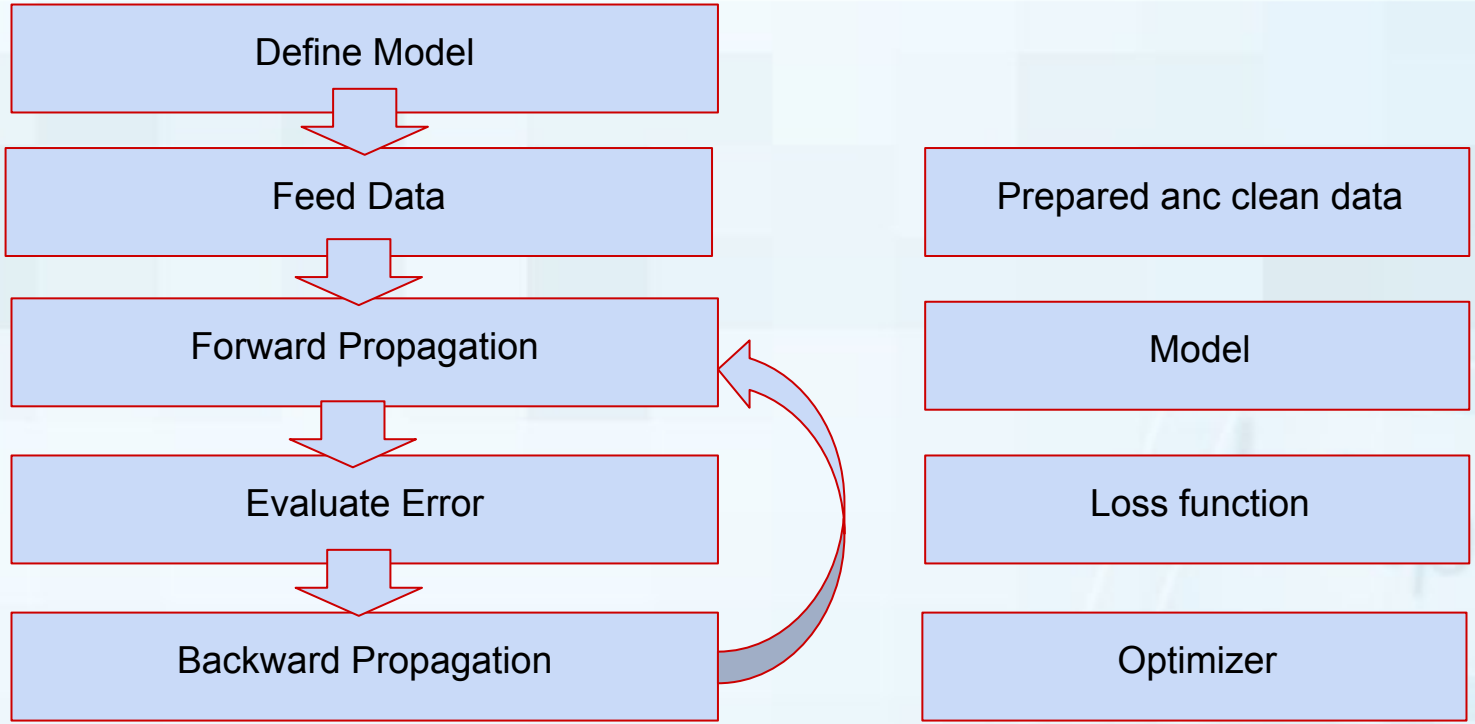
```
# Movie Review Data
import requests
import io
import tarfile
movie_data_url =
'http://www.cs.cornell.edu/people/pabo/movie-review-data/rt-polaritydata.tar.gz'
r = requests.get(movie_data_url)
# Stream data into temp object
stream_data = io.BytesIO(r.content)
(...)
```

Module 3
Simple models with Tensorflow

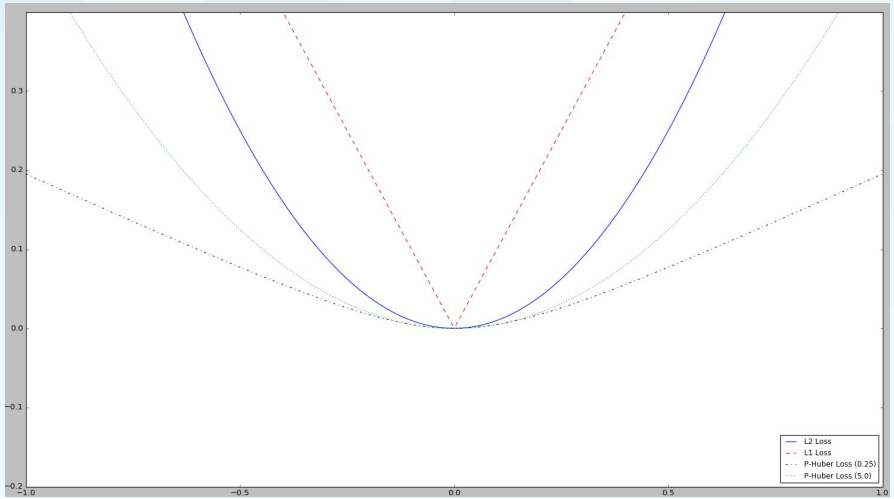
Loss functions

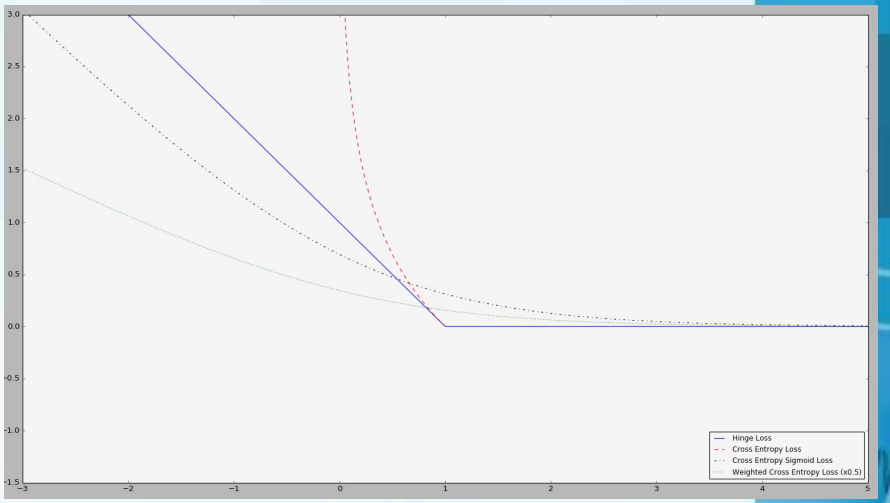
- You can’t learn unless you know how bad/good your past results was.
- This is designated by a ‘Loss function’.
- The idea is we have a labeled data set and we look at each data point. We run the data point forward through the network and then see if we need to increase or decrease the output.
- We then change parameters according to the derivatives in the network.
- We loop through this many times until we reach the lowest error in our training.

Training scheme

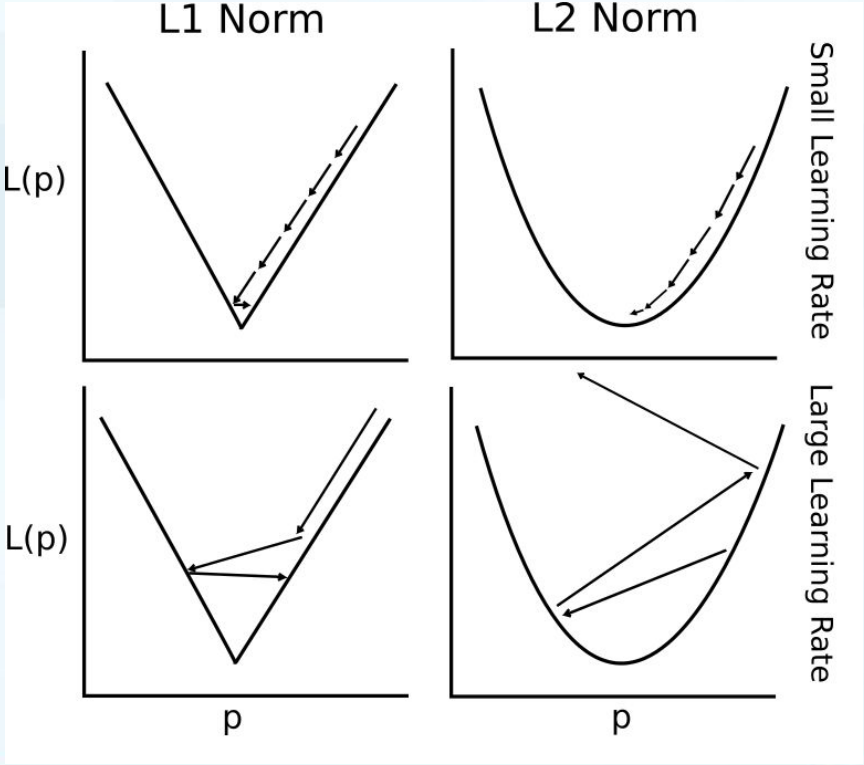


Loss functions

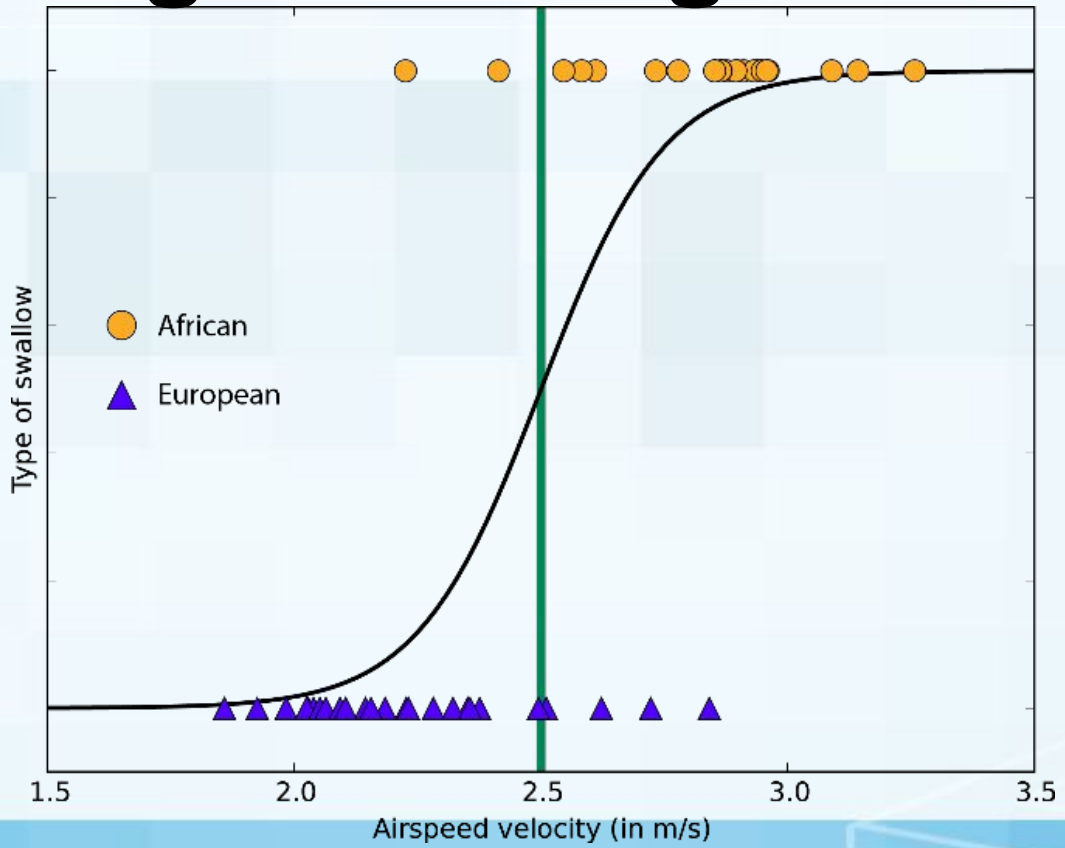




Learning Curve



Logistics Regression

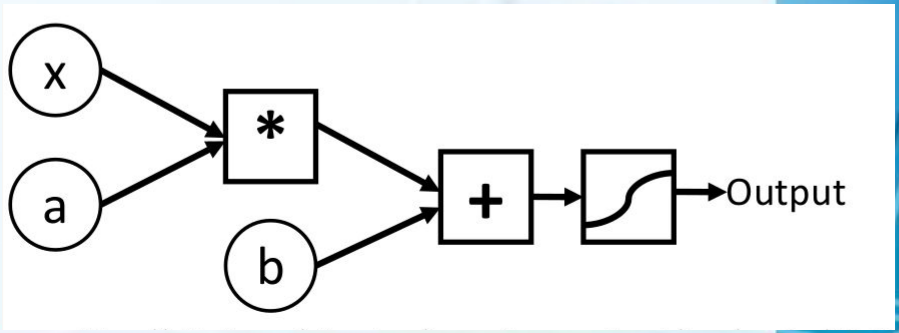


Logistic Regression with NN

- A neural network is very similar to what we have been doing, except that we bound the outputs between 0 and 1… with a sigmoid function. (Called an activation function)
- The sigmoid function is nice, because it turns out that the derivative is related to itself:
- We can use this fact in our 'chain rule' for derivatives.

$$F(x, a, b) = \sigma(ax + b) \qquad \sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\frac{d\sigma}{dx} = \sigma(x) \cdot (1 - \sigma(x))$$



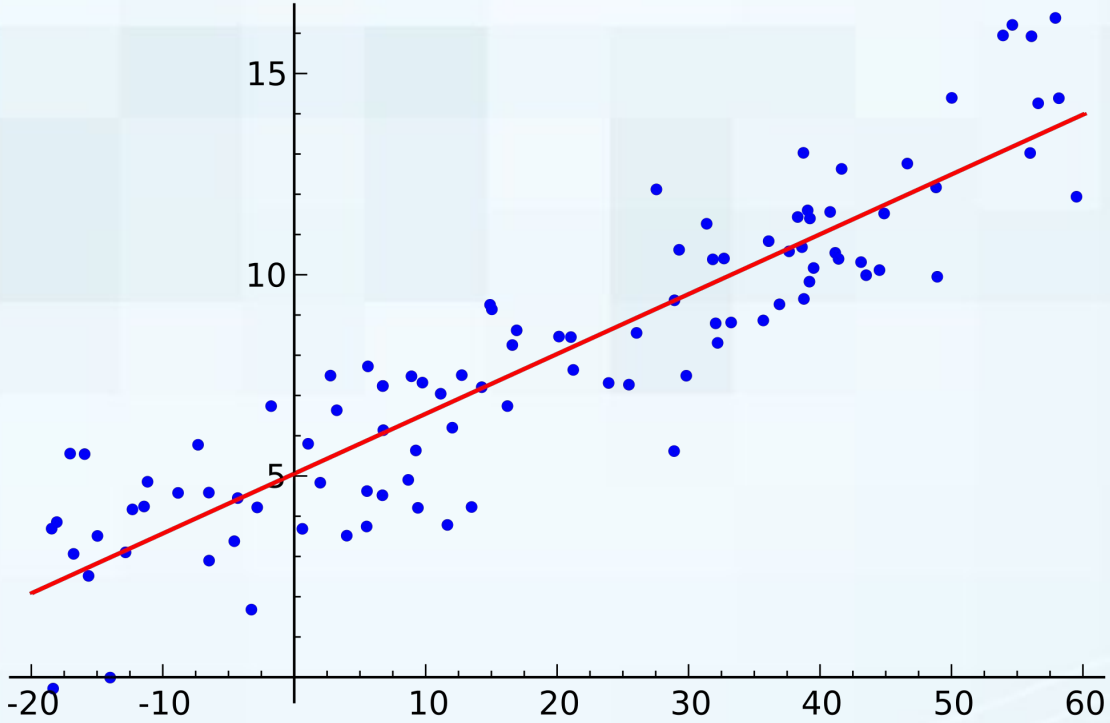
Logistics Regression Modeling

```
def get_iris_data():
    iris   = datasets.load_iris()
    data   = iris["data"]
    target = iris["target"]
    N, M  = data.shape
    all_X = np.ones((N, M + 1))
    all_X[:, 1:] = data
    num_labels = len(np.unique(target))
    all_Y = np.eye(num_labels)[target]  # One liner trick!
    return train_test_split(all_X, all_Y, test_size=0.33,
random_state=RANDOM_SEED)
```

Logistics Regression Modeling

```
def forwardprop(X, w_1):
    """
    Forward-propagation.
    """
    yhat = tf.nn.sigmoid(tf.matmul(X, w_1))  # The \sigma function
    return yhat
```

Linear Regression



Challenge

Prepare script for linear regression for the Iris data

We will use the iris data, specifically:

y = Sepal Length

x = Pedal Length, Petal Width, Sepal Width

Use functions

- init_weights,
- forwardprop -> modify for linear regression,
- get_iris_data -> modify for data entry train/target

Challenge - Hint

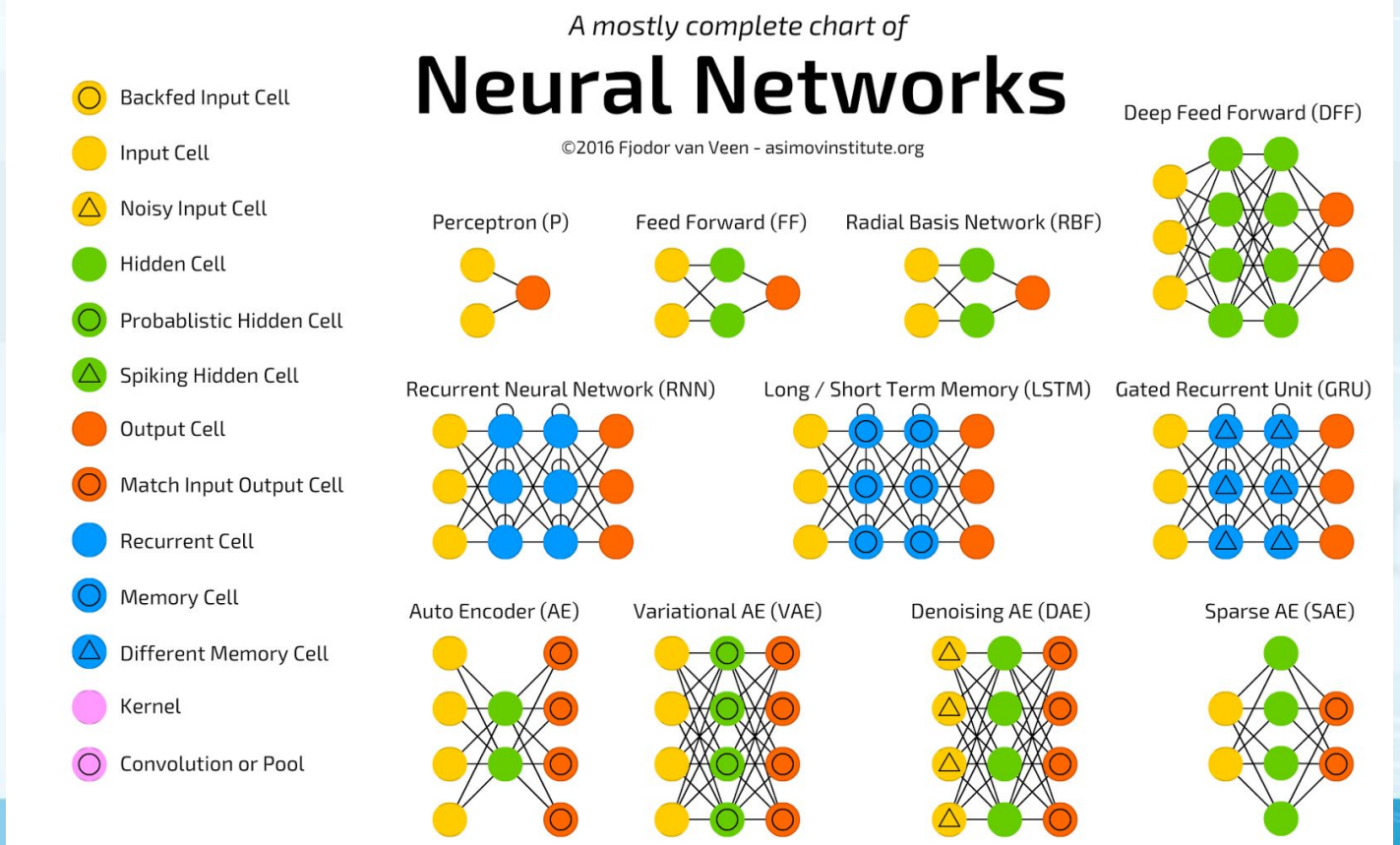
```
def forwardprop(X, w_1):
    """
    Forward-propagation.
    """
    yhat = tf.matmul(X, w_1)
    return yhat
```

Challenge - Hint

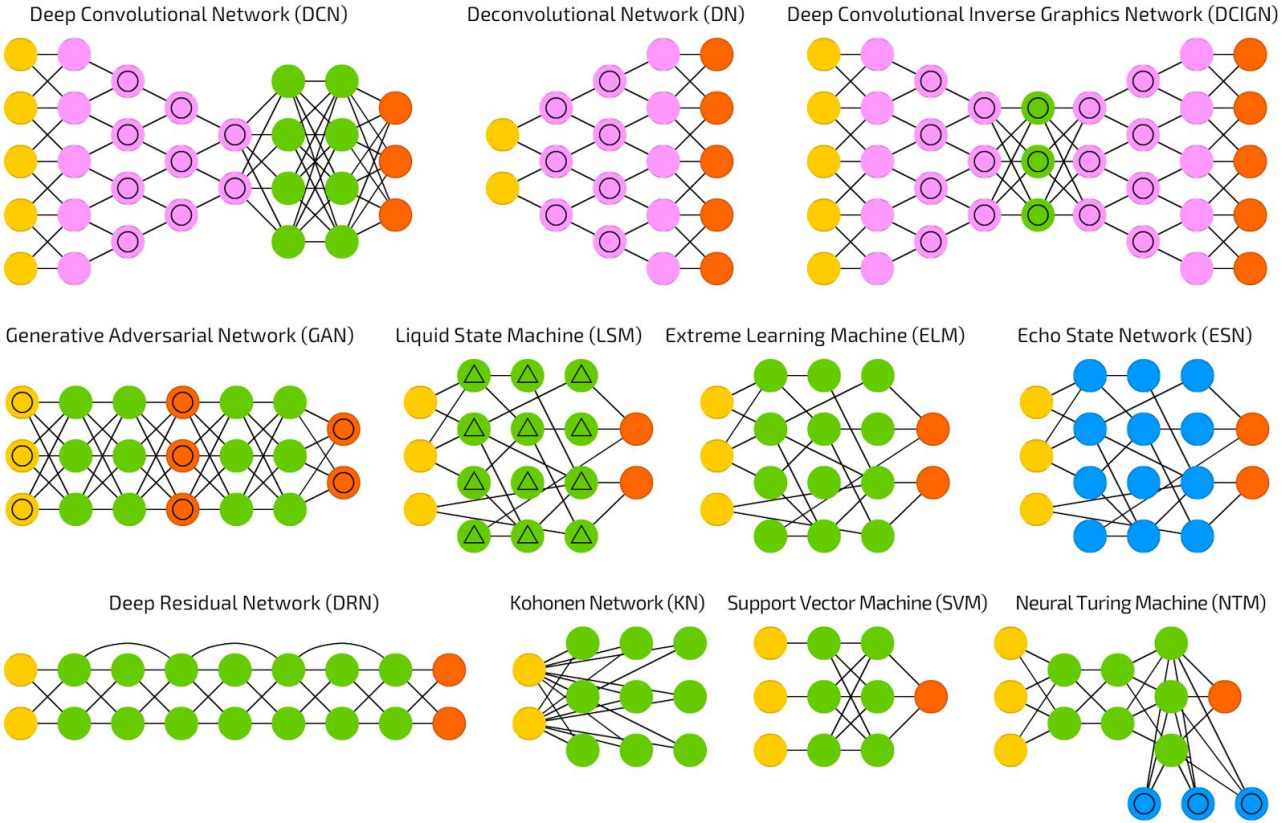
```
def get_iris_data():
    iris   = datasets.load_iris()
    data   = np.array([[x[1], x[2], x[3]] for x in iris.data])
    target = np.array([y[0] for y in iris.data])
    # Prepend the column of 1s for bias
    N, M  = data.shape
    all_X = np.ones((N, M + 1))
    all_X[:, 1:] = data
    all_Y = target  # One liner trick!
    return train_test_split(all_X, all_Y, test_size=0.33,
random_state=RANDOM_SEED)
```

Module 4
Single Layer Perceptron

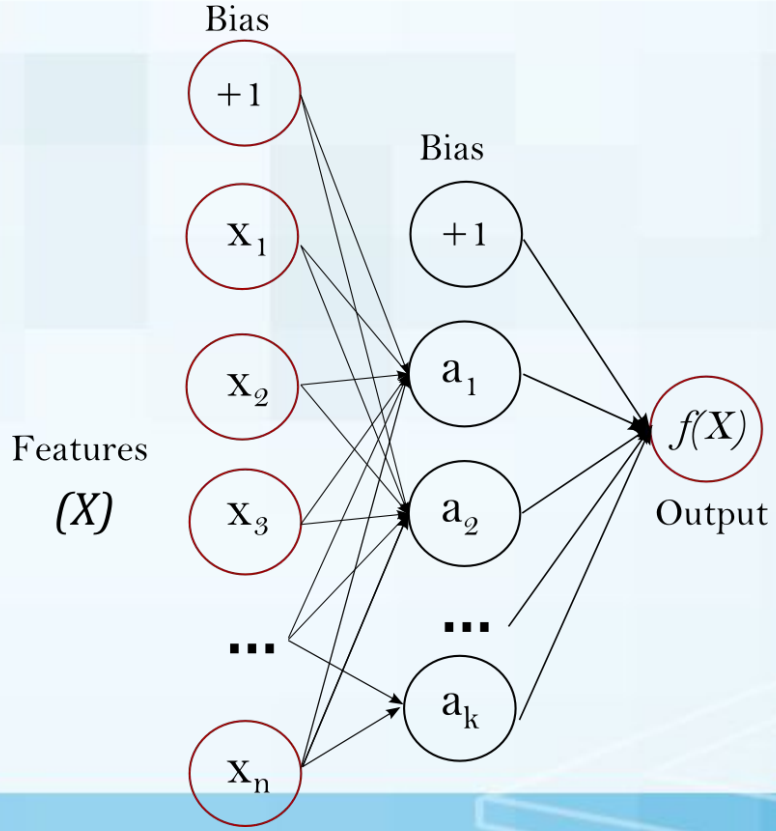
Basic Neutral Networks (NN)



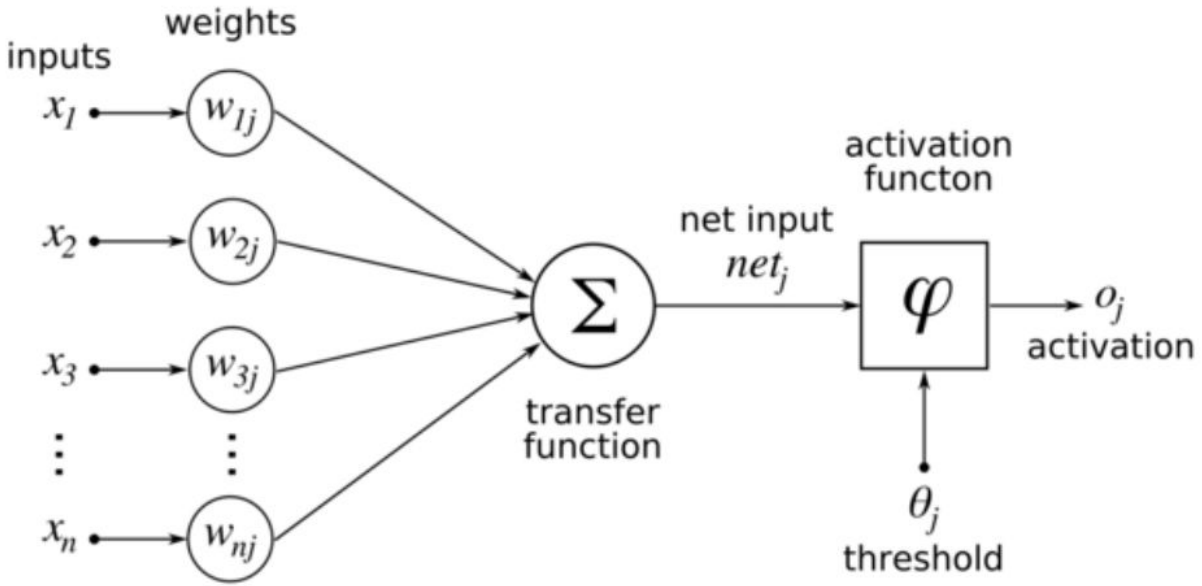
Basic Neutral Networks (NN)



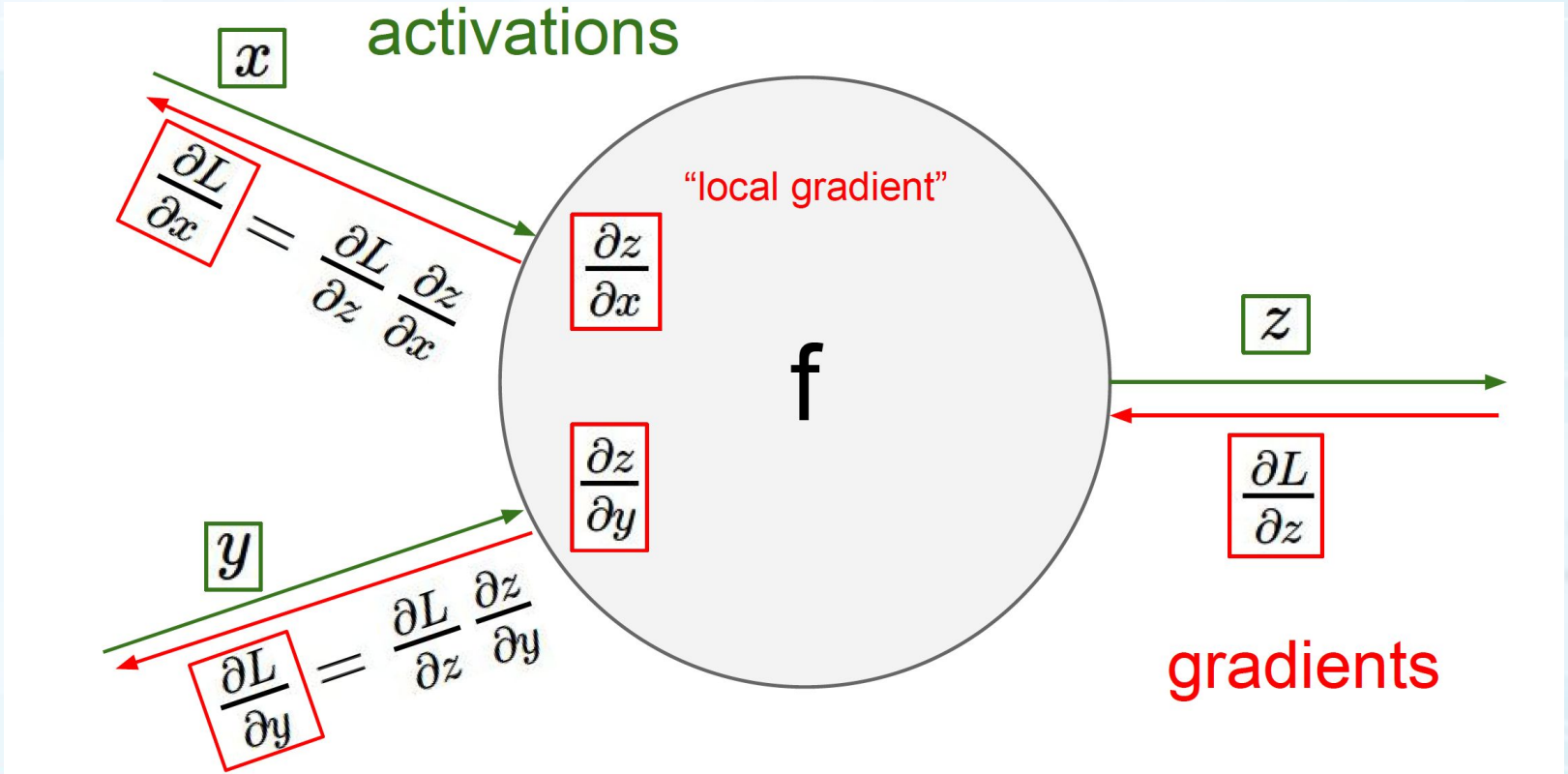
Single Layer Perceptron



Feed forward

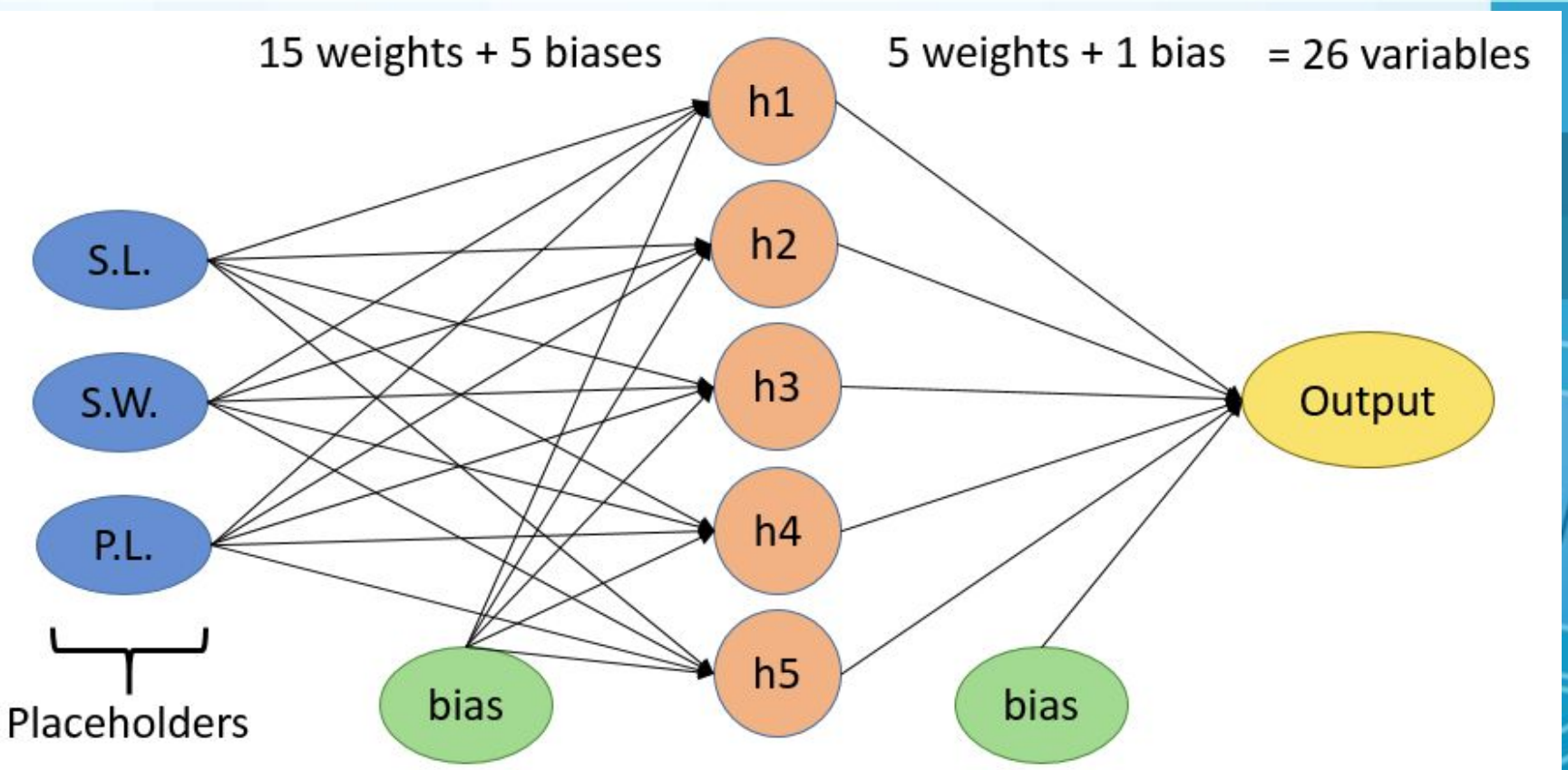


Backward Propagation



Source : http://cs231n.github.io

Basic Neutral Networks (NN)



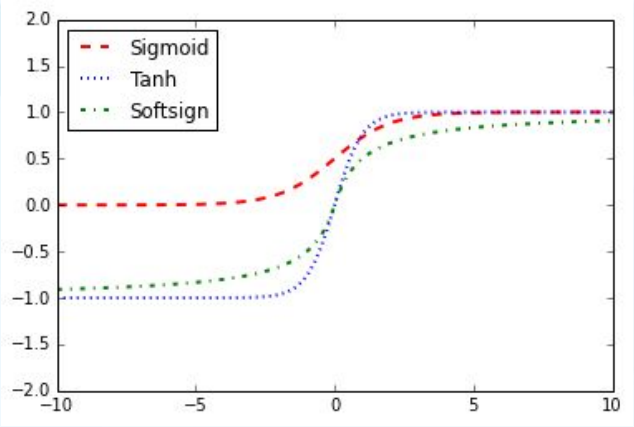
One Layer Neural Networks

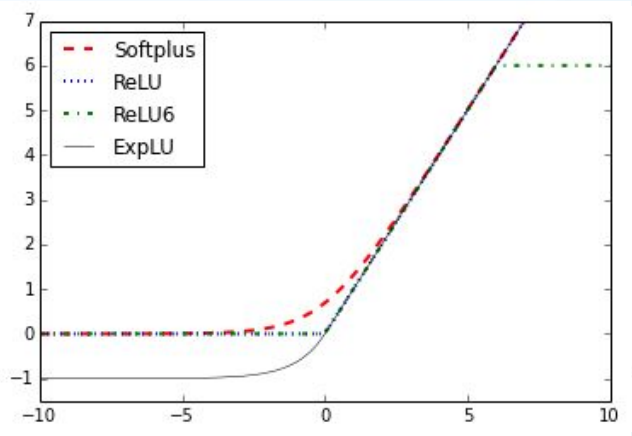
```
def neural_network_model(data):
    hidden_1_layer =
            {'weights':tf.Variable(tf.random_normal([784, n_nodes_hl1])),
                'biases':tf.Variable(tf.random_normal([n_nodes_hl1]))}
    output_layer =
            {'weights':tf.Variable(tf.random_normal([n_nodes_hl1, n_classes])),
                'biases':tf.Variable(tf.random_normal([n_classes])),}
    l1 = tf.add(tf.matmul(data,hidden_1_layer['weights']), hidden_1_layer['biases'])
    l1 = tf.nn.relu(l1)
    output = tf.matmul(l1,output_layer['weights']) + output_layer['biases']
    return output
```

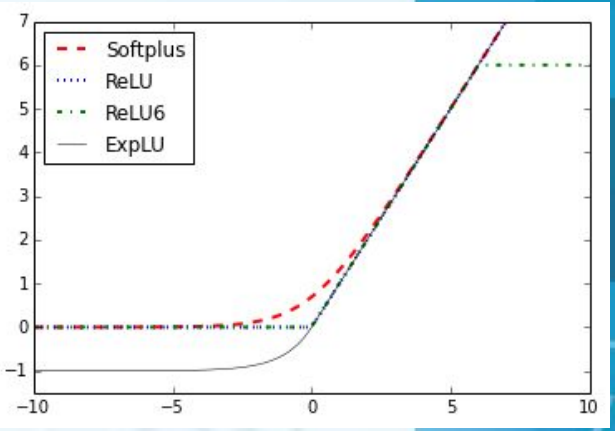
Activation Functions

- tf.nn.relu(features, name=None)
- tf.nn.relu6(features, name=None)
- tf.nn.elu(features, name=None)
- tf.nn.softplus(features, name=None)
- tf.nn.softsign(features, name=None)
- tf.nn.dropout(x, keep_prob, noise_shape=None, seed=None, name=None)
- tf.nn.bias_add(value, bias, data_format=None, name=None)
- tf.sigmoid(x, name=None)
- tf.tanh(x, name=None)

Activation functions







Activation Functions

- Sigmoid function

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

- Rectified Linear Unit (ReLU):

$$\sigma(x) = \max(x, 0)$$

- Softplus:

$$\sigma(x) = \ln(1 + e^{x})$$

- Leaky ReLU:

$$\sigma(x) = \max(x, ax)$$

Challenge

Prepare one layer perceptron neural network for the Iris data.

Use:

- get_data() function from Logistic Regression script.
- neural_network_model() -> modify it for iris data
- train_regression() -> modify it for Single Layer Perceptron

Challenge - Hint

```
RANDOM_SEED = 42
tf.set_random_seed(RANDOM_SEED)
n_nodes_hl1 = 100


n_classes = 3


X = tf.placeholder('float', [None, 5])
y = tf.placeholder('float')
```

Challenge - Hint

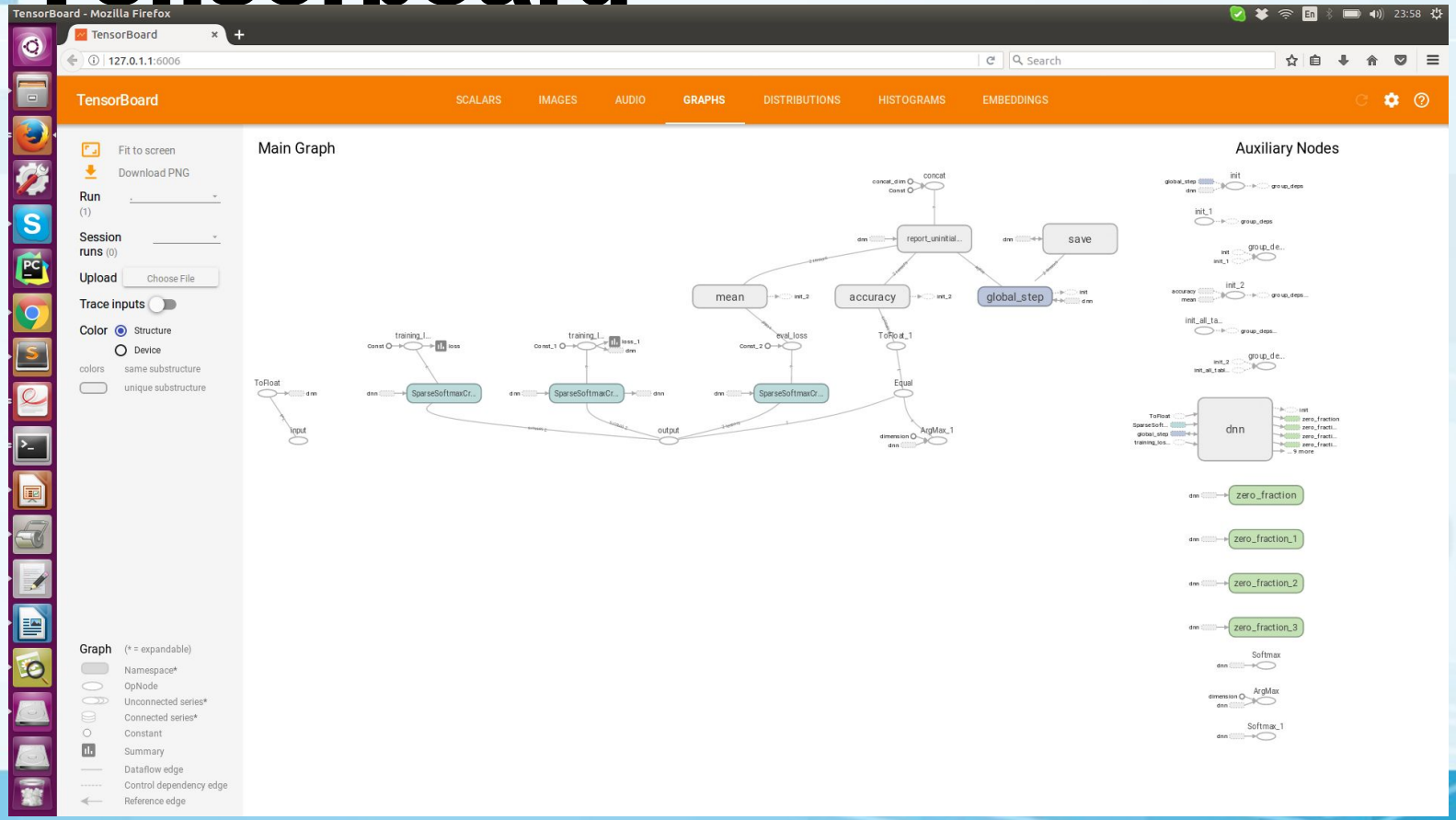
```
def neural_network_model(data):
    hidden_1_layer = {'weights':tf.Variable(tf.random_normal([5, n_nodes_hl1])),
                      'biases':tf.Variable(tf.random_normal([n_nodes_hl1]))}
    output_layer = {'weights':tf.Variable(tf.random_normal([n_nodes_hl1, n_classes])),
                    'biases':tf.Variable(tf.random_normal([n_classes])),}
    l1 = tf.add(tf.matmul(data,hidden_1_layer['weights']), hidden_1_layer['biases'])
    l1 = tf.nn.relu(l1)

    output = tf.matmul(l1,output_layer['weights']) + output_layer['biases']
    return output
```

Challenge - Hint

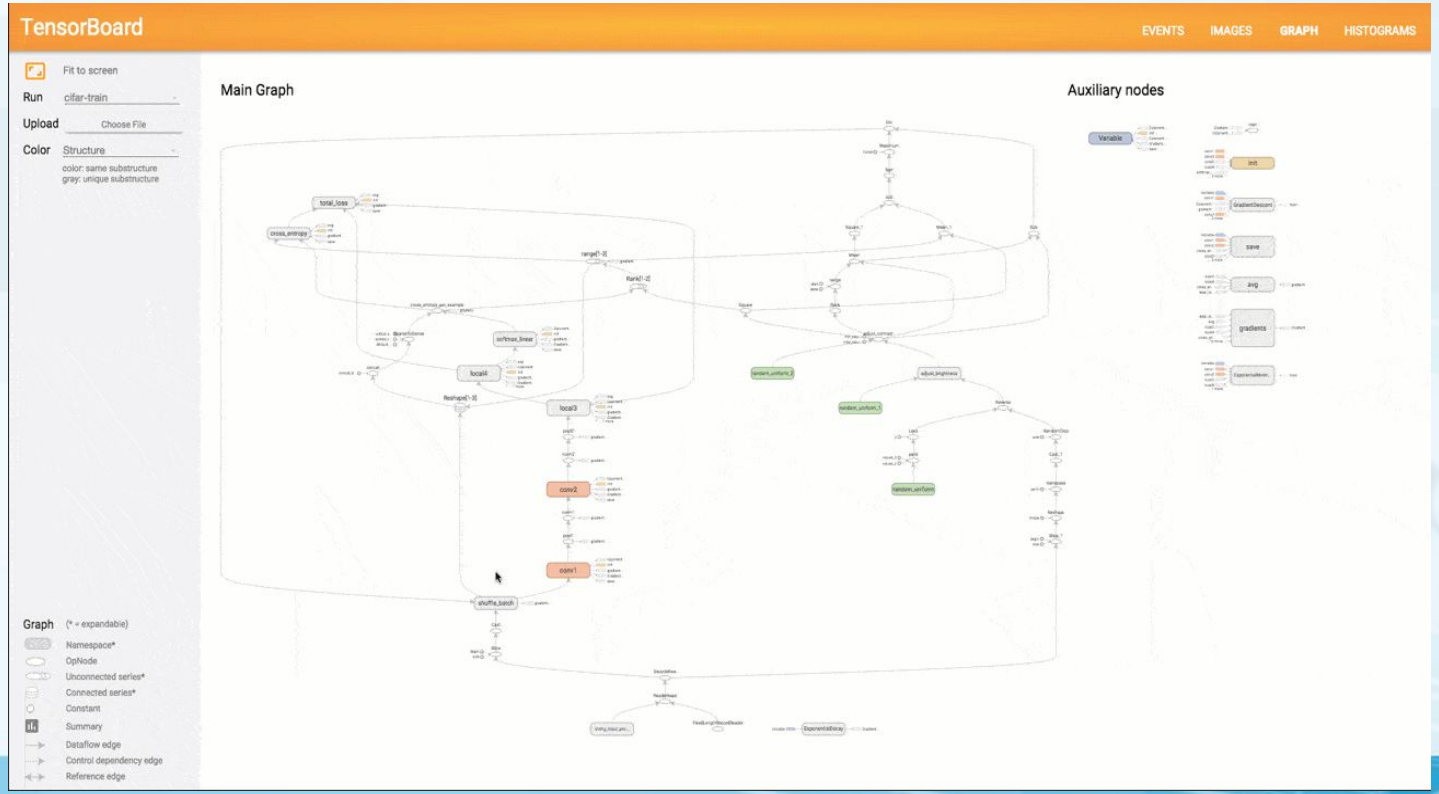
```
def train_regression():
    train_X, test_X, train_y, test_y = get_iris_data()
    # Layer's sizes
    x_size = train_X.shape[1]   # Number of input nodes: 4 features and 1 bias
    y_size = train_y.shape[1]   # Number of outcomes (3 iris flowers)
    # Symbols
    X = tf.placeholder("float", shape=[None, x_size])
    y = tf.placeholder("float", shape=[None, y_size])
    # Forward propagation
    yhat    = neural_network_model(X)
    predict = tf.argmax(yhat, axis=1)   (...)
```

Module 5
Tensorboard

Tensorboard



What is TensorFlow?



Tensorboard naming

```
with tf.name_scope('Model'):
    pred = tf.nn.softmax(tf.matmul(x, W) + b) # Softmax
with tf.name_scope('Loss'):
    cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(pred), reduction_indices=1))
with tf.name_scope('SGD'):
    optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
with tf.name_scope('Accuracy'):
    acc = tf.equal(tf.argmax(pred, 1), tf.argmax(y, 1))
    acc = tf.reduce_mean(tf.cast(acc, tf.float32))
```

Tensorboard summary

```
# Create a summary to monitor cost tensor
tf.summary.scalar("loss", cost)
# Create a summary to monitor accuracy tensor
tf.summary.scalar("accuracy", acc)
# Merge all summaries into a single op
merged_summary_op = tf.summary.merge_all()
```

Tensorboard naming

```
logs_path = '/tmp/tensorflow_logs/example'
with tf.Session() as sess:
    sess.run(init)

    # op to write logs to Tensorboard
    summary_writer = tf.summary.FileWriter(logs_path,
       graph=tf.get_default_graph())
```

Tensorboard

By pasting tensorboard --logdir=/tmp/my_model into your terminal the session of tensorboard would be created

Look at your computational graph at: http://127.0.1.1:6006

Challenge

Prepare Tensorboard for previous model (Module 4_1)

Add scope for the:

- model
- cost
- optimizer
- accuracy
- name placeholders and variables
- make sure you include summary into your code

Challenge - Hint

```
def train_neural_network(x):
    with tf.name_scope('Model'):
        prediction = neural_network_model(x)
    with tf.name_scope('Loss'):
        cost = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits  …)
with tf.name_scope('Adam opt'):
        optimizer = tf.train.AdamOptimizer().minimize(cost)
    with tf.name_scope('Accuracy'):
        correct = tf.equal(tf.argmax(prediction, 1), tf.argmax(y, 1))
        accuracy = tf.reduce_mean(tf.cast(correct, 'float'))
```

Challenge - Hint

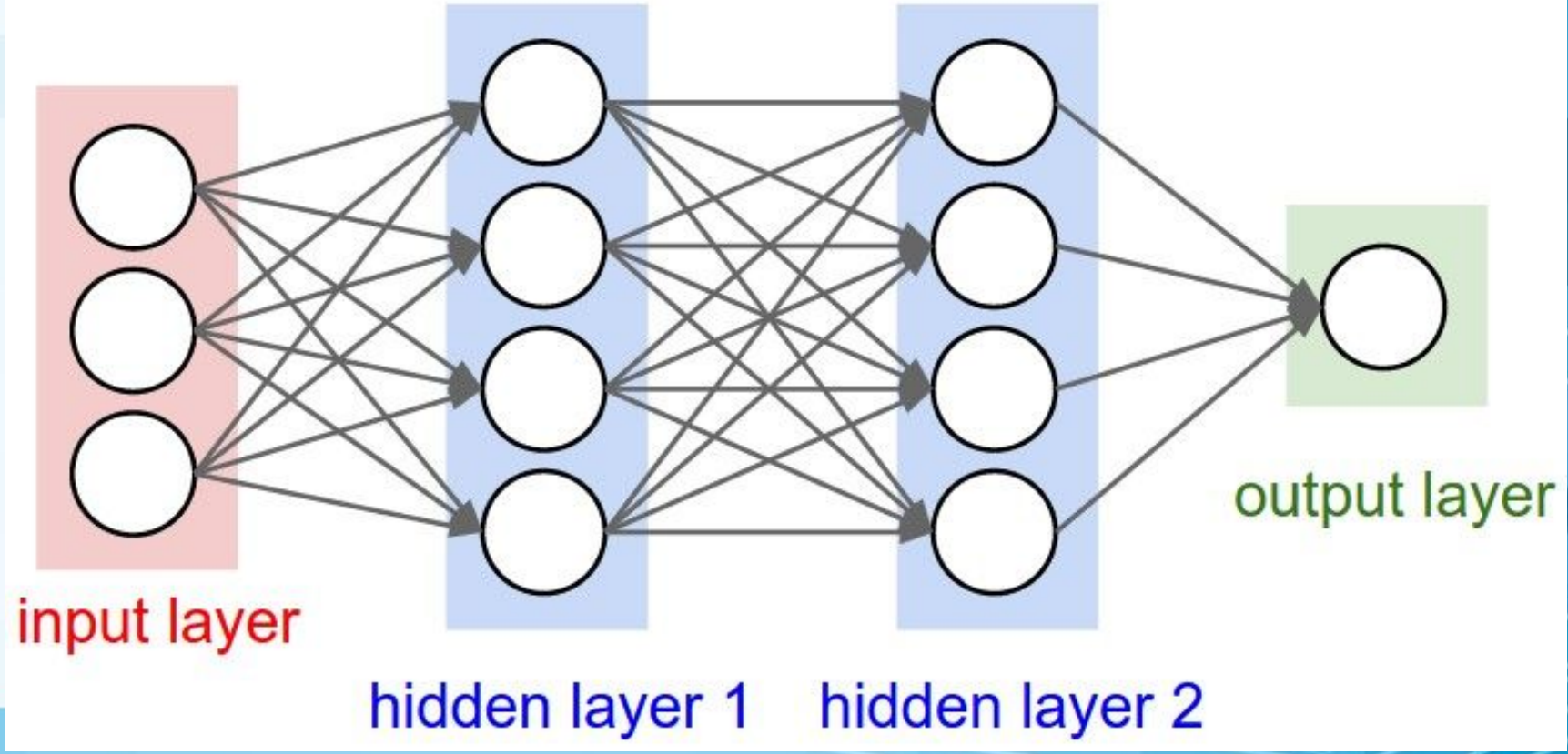
```
# Create a summary to monitor cost tensor
tf.summary.scalar("loss", cost)
# Create a summary to monitor accuracy tensor
tf.summary.scalar("accuracy", accuracy)
# Merge all summaries into a single op
merged_summary_op = tf.summary.merge_all()
```

Challenge - Hint

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    summary_writer = tf.summary.FileWriter(logs_path, graph=tf.get_default_graph())
    for epoch in range(hm_epochs):
        epoch_loss = 0
        total_batch = int(mnist.train.num_examples/batch_size)
        for i in range(total_batch):
            epoch_x, epoch_y = mnist.train.next_batch(batch_size)
            _, c, summary = sess.run([optimizer, cost, merged_summary_op],
                    feed_dict={x: epoch_x, y: epoch_y})
            summary_writer.add_summary(summary, epoch * total_batch + i)
            epoch_loss += c
```

Module 6
Multiple Layer Neural Network

Multi Layer Perceptron



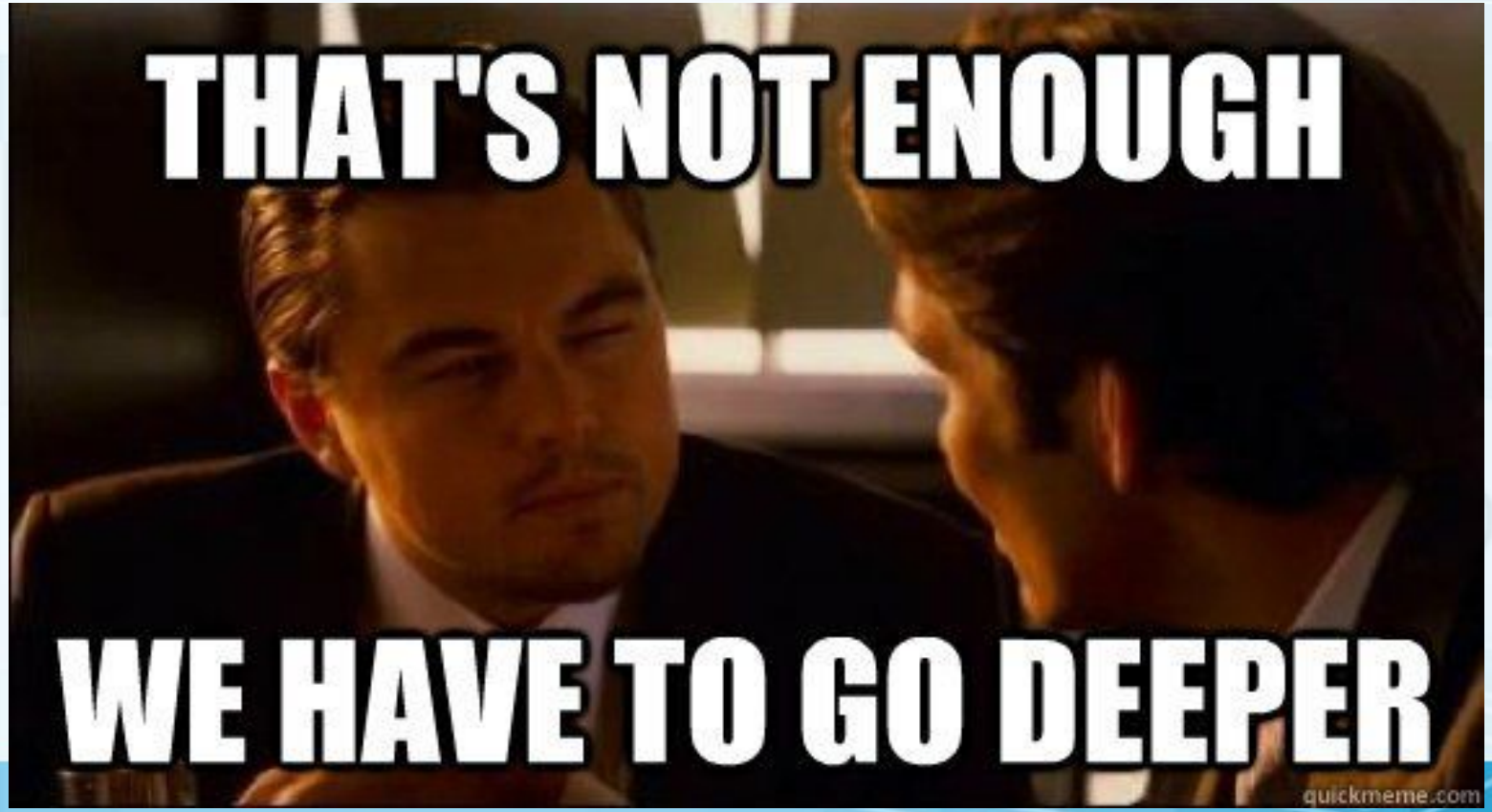
Multiple Layers Neural Network

```
def neural_network_model(data):
    hidden_1_layer =
            {'weights':tf.Variable(tf.random_normal([784, n_nodes_hl1])),
                'biases':tf.Variable(tf.random_normal([n_nodes_hl1]))}
    hidden_2_layer =
            {'weights':tf.Variable(tf.random_normal([n_nodes_hl1, n_nodes_hl2])),
                'biases':tf.Variable(tf.random_normal([n_nodes_hl2]))}
    output_layer =
            {'weights':tf.Variable(tf.random_normal([n_nodes_hl3, n_classes])),
                'biases':tf.Variable(tf.random_normal([n_classes])),}
```

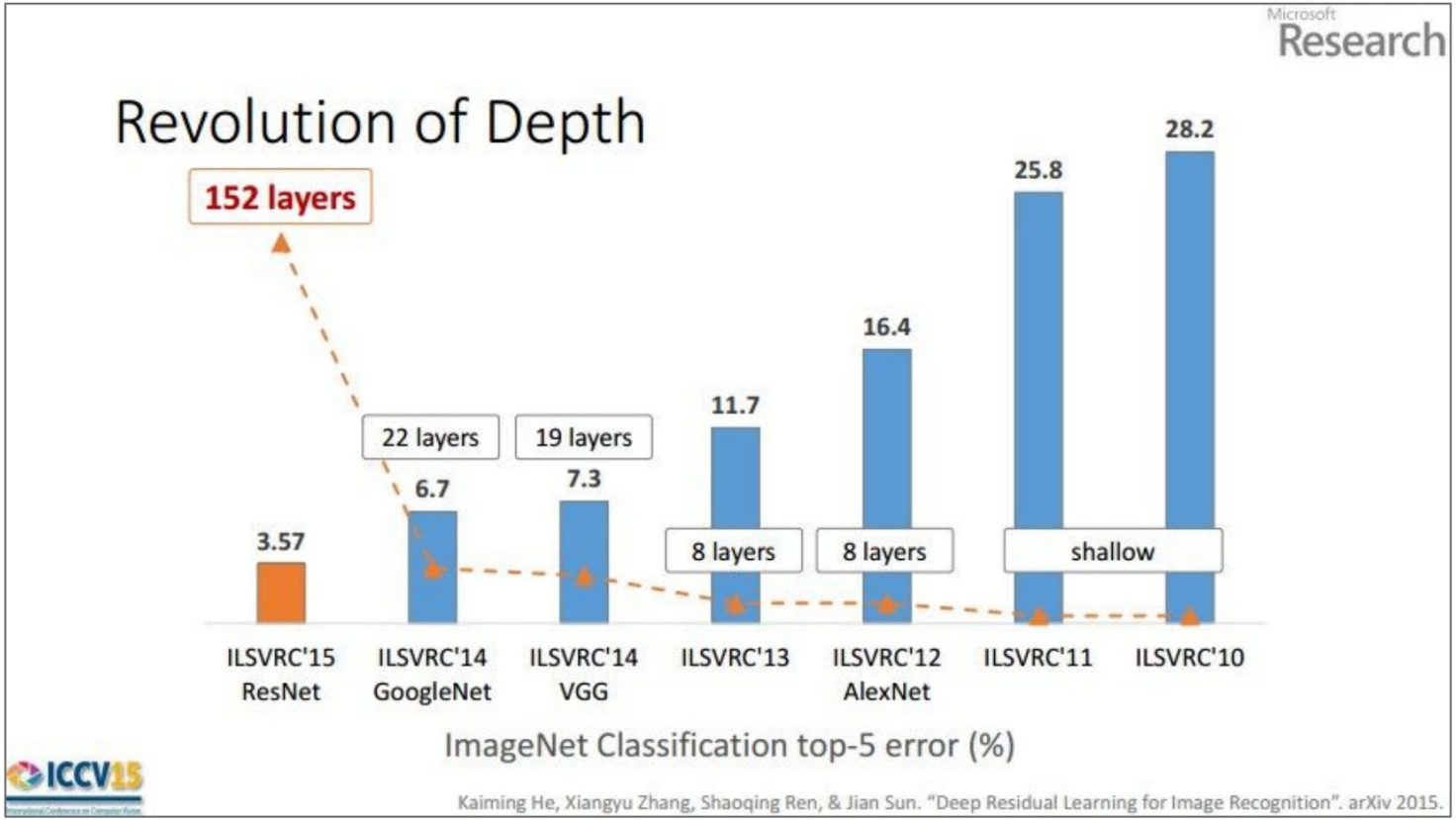
Multiple Layers Neural Network

```
(...)
l1 = tf.add(tf.matmul(data,hidden_1_layer['weights']),
    hidden_1_layer['biases'])
 l1 = tf.nn.relu(l1)

 l2 = tf.add(tf.matmul(l1,hidden_2_layer['weights']), hidden_2_layer['biases'])
 l2 = tf.nn.relu(l2)

 output = tf.matmul(l2,output_layer['weights']) + output_layer['biases']

 return output
```

Multi Layer Perceptron

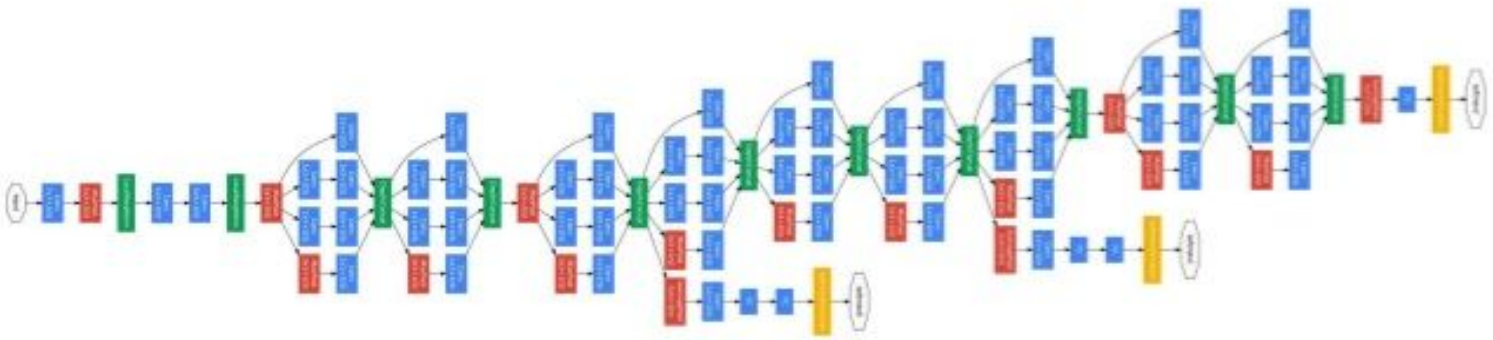


Depth Revolution



Deep Neural Networks

The Inception Architecture (GoogLeNet, 2014)

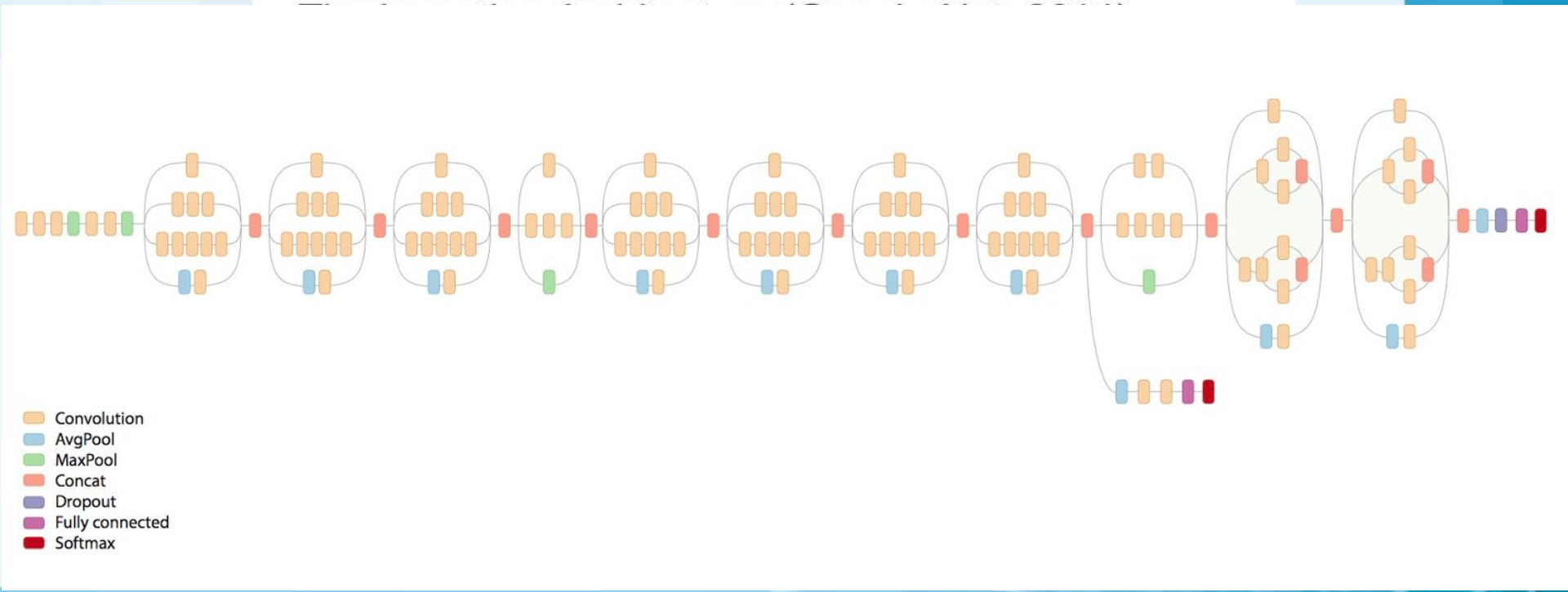


Going Deeper with Convolutions

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

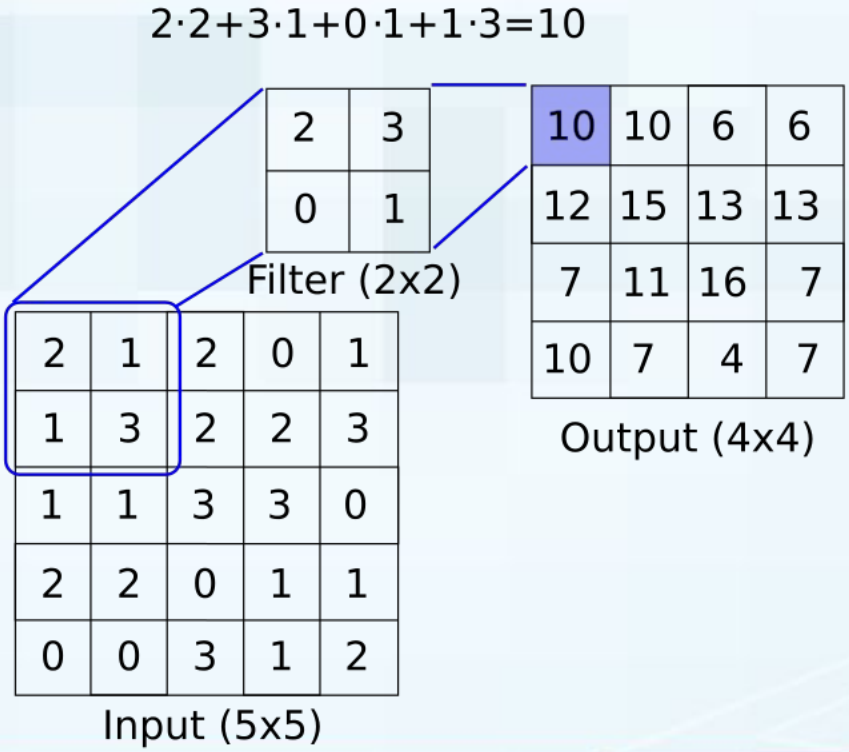
ArXiv 2014, CVPR 2015

Deep Neural Networks

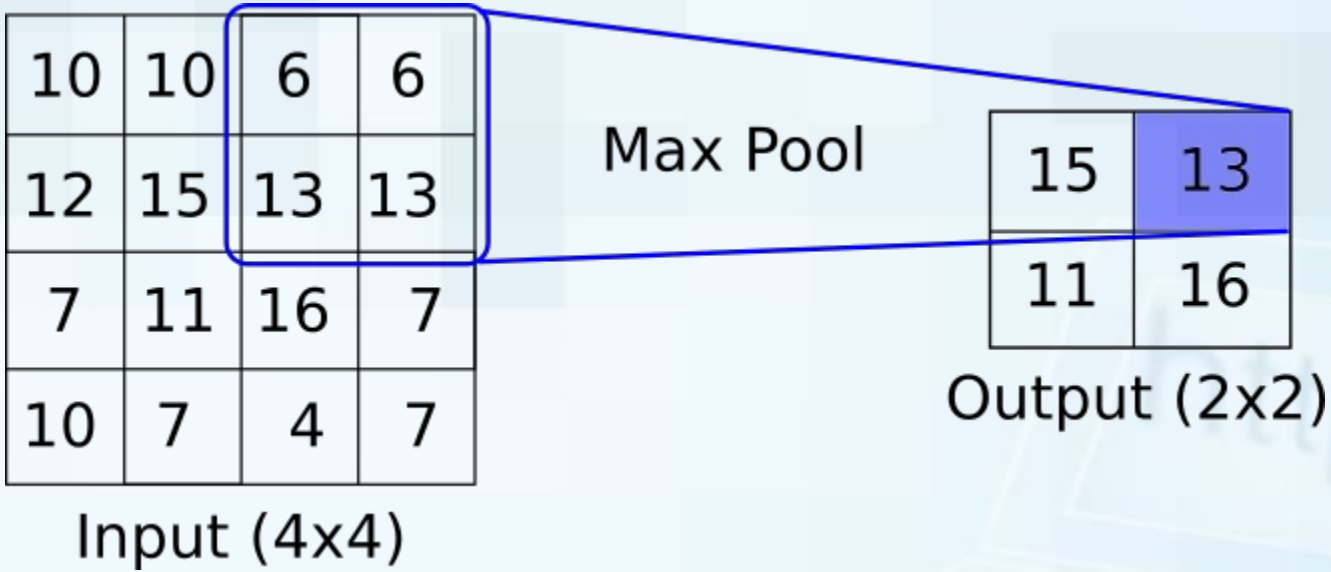


Module 7
Convolutional neural networks

Convolutional NN (CNN)



Pooling


10 10 6 6
12 15 13 13
7 11 16 7
10 7 4 7
Input (4x4)
Max Pool
15 13
11 16
Output (2x2)

Dropout

- Dropout is a way to increase the training strength and regularize the parameters.
- During training, we randomly select activation functions and make the output equal to zero.
- This helps find multiple pathways for prediction.
- This also helps of other parameters in the network.

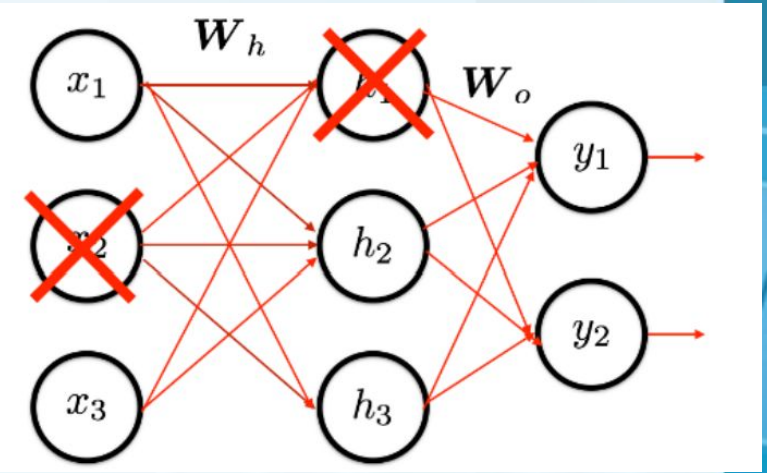
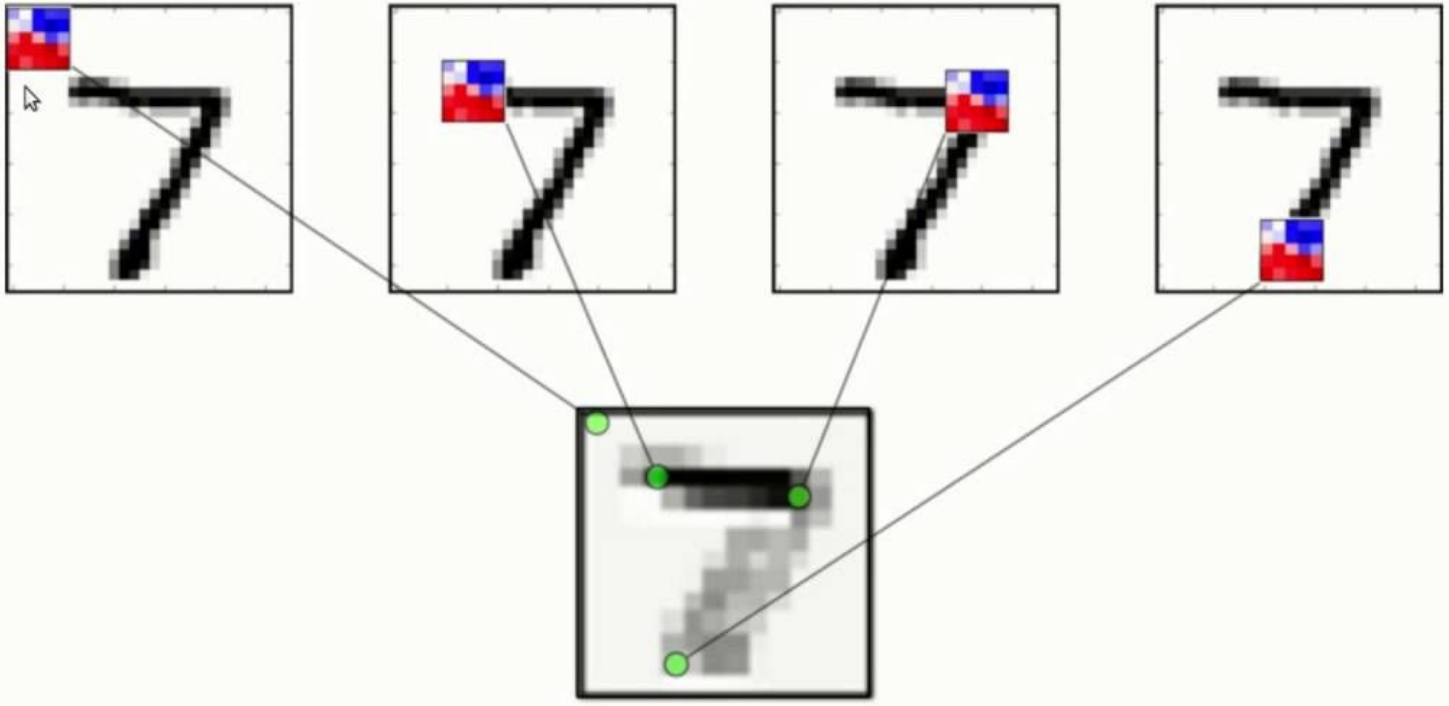


Image Processing with CNN



CNN - Playground

http://redcatlabs.com/2017-03-20_TFandDL_IntroToCNNs/CNN-demo.html#/

CNN with Tensorflow

```
def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1,1,1,1], padding='SAME')


def maxpool2d(x):
    #size of window movement of window
    return tf.nn.max_pool(x, ksize=[1,2,2,1], strides=[1,2,2,1],
padding='SAME')
```

CNN with Tensorflow

```
weights = {'W_conv1':tf.Variable(tf.random_normal([5,5,1,32])),
           'W_conv2':tf.Variable(tf.random_normal([5,5,32,64])),
           'W_fc':tf.Variable(tf.random_normal([7*7*64,1024])),
           'out':tf.Variable(tf.random_normal([1024, n_classes]))}

biases = {'b_conv1':tf.Variable(tf.random_normal([32])),
          'b_conv2':tf.Variable(tf.random_normal([64])),
          'b_fc':tf.Variable(tf.random_normal([1024])),
          'out':tf.Variable(tf.random_normal([n_classes]))}
```

CNN with Tensorflow

```
weights = {'W_conv1':tf.Variable(tf.random_normal([5,5,1,32])),
           'W_conv2':tf.Variable(tf.random_normal([5,5,32,64])),
           'W_fc':tf.Variable(tf.random_normal([7*7*64,1024])),
           'out':tf.Variable(tf.random_normal([1024, n_classes]))}

biases = {'b_conv1':tf.Variable(tf.random_normal([32])),
          'b_conv2':tf.Variable(tf.random_normal([64])),
          'b_fc':tf.Variable(tf.random_normal([1024])),
          'out':tf.Variable(tf.random_normal([n_classes]))}
```

CNN with Tensorflow

```
x = tf.reshape(x, shape=[-1, 28, 28, 1])
conv1 = tf.nn.relu(conv2d(x, weights['W_conv1']) + biases['b_conv1'])
conv1 = maxpool2d(conv1)
conv2 = tf.nn.relu(conv2d(conv1, weights['W_conv2']) + biases['b_conv2'])
conv2 = maxpool2d(conv2)

fc = tf.reshape(conv2,[-1, 7*7*64])
fc = tf.nn.relu(tf.matmul(fc, weights['W_fc'])+biases['b_fc'])
fc = tf.nn.dropout(fc, keep_rate)

output = tf.matmul(fc, weights['out'])+biases['out']
```

Challenge

Prepare CNN neural network for the Iris data.

Use:

- Two conwolutional layers

(filter size: first: 5, returns: 16, second: 3, returns:32)

- Two pooling layers

- reshape to appropriate size for the fully connected layer

Train the network for 3 epochs

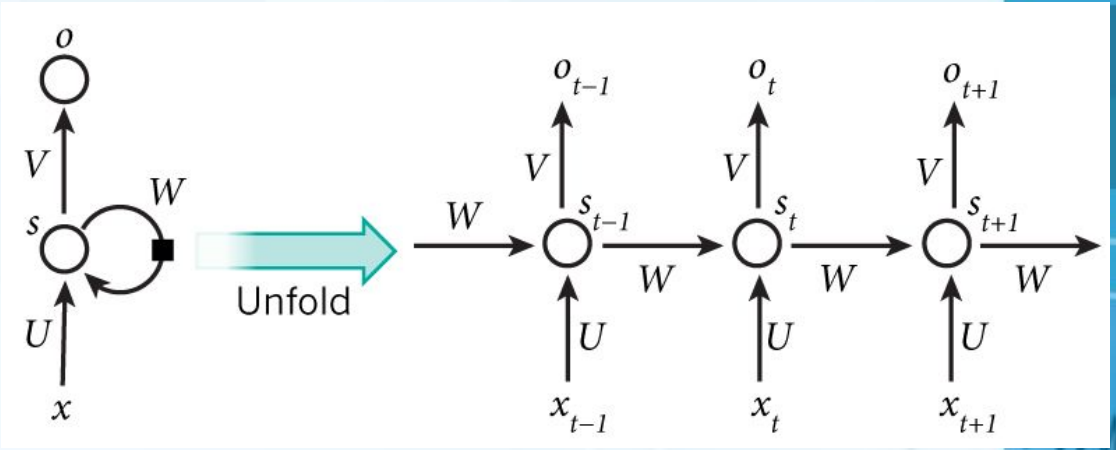
Use bach size of 100

Calculate accuracy for just 5000 first observations from train set

Module 8
Recurent Neural Networks

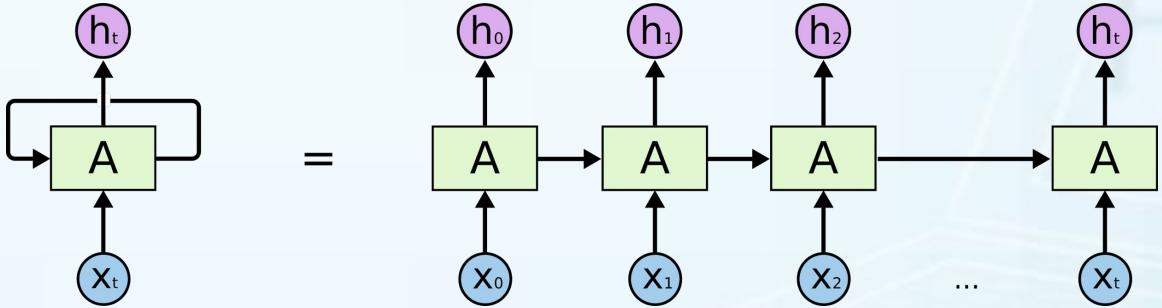
Recurrent Neural Networks (RNN)

- U,V,W = Weights
- S = layer
- X = input
- O = output



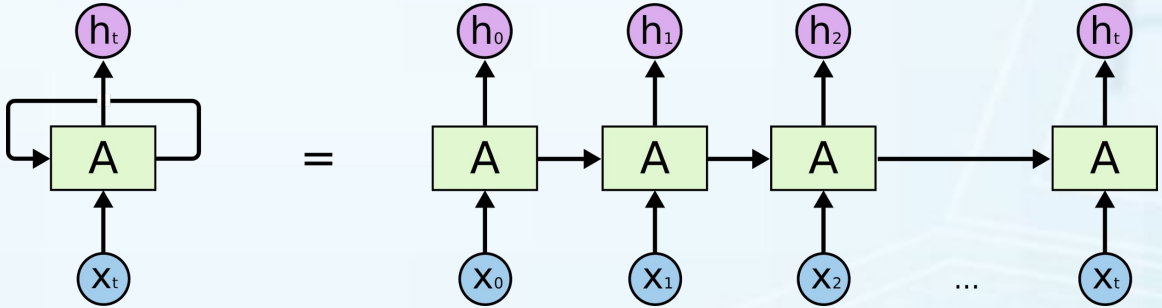
Recurrent Neural Networks (RNN)

- Recurrent Neural Networks (RNN) are networks that can ‘see’ outputs from prior layers. Very helpful for sequences.
- The most common usage is to predict the next letter from the prior letter. We train this network on large text samples.

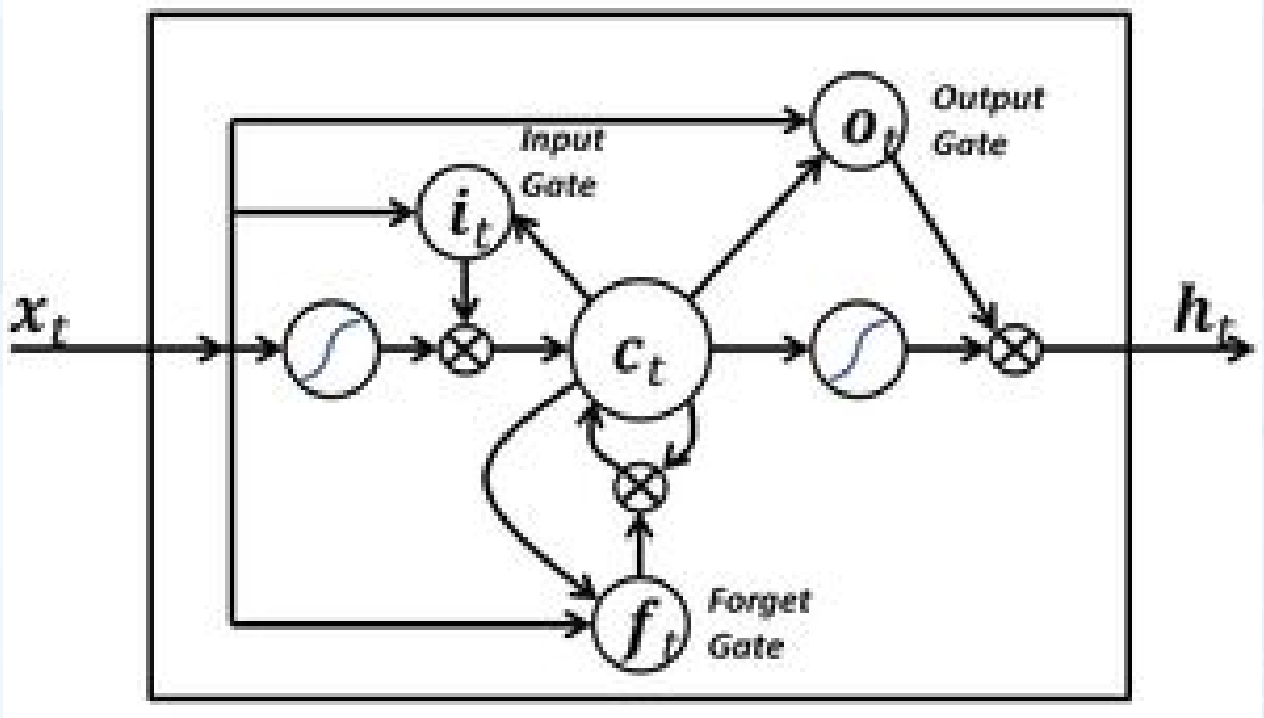


Recurrent Neural Networks (RNN)

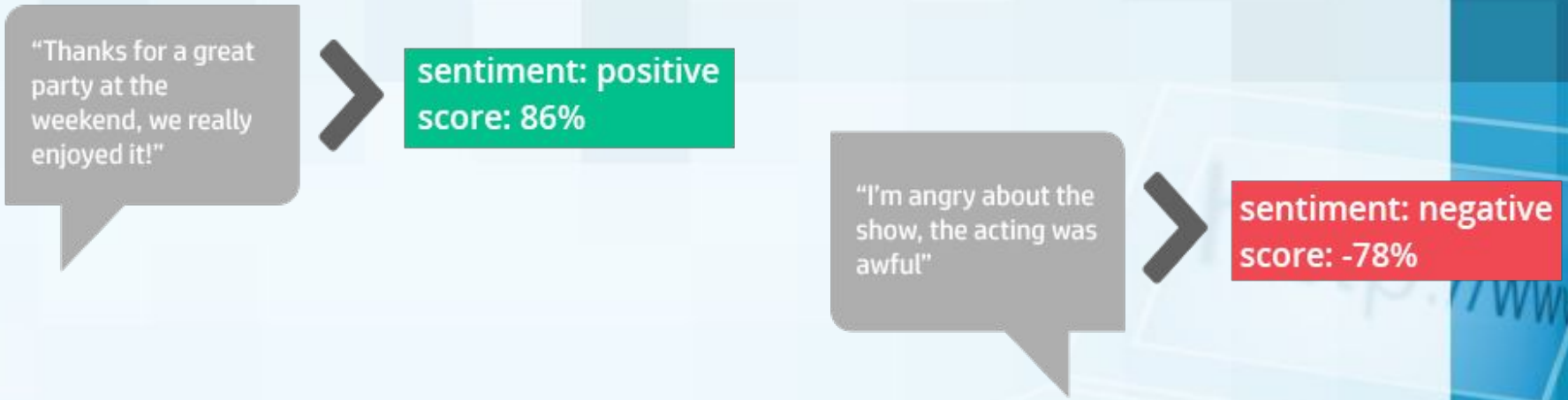
- Recurrent Neural Networks (RNN) are networks that can ‘see’ outputs from prior layers. Very helpful for sequences.
- The most common usage is to predict the next letter from the prior letter. We train this network on large text samples.



LSTM - long short term memory



Language Processing with RNN



RNN with Tensorflow

```
layer = {'weights':tf.Variable(tf.random_normal([rnn_size,n_classes])),
        'biases':tf.Variable(tf.random_normal([n_classes]))}
x = tf.transpose(x, [1,0,2])
x = tf.reshape(x, [-1, chunk_size])
x = tf.split(x, n_chunks, 0)
lstm_cell = rnn.BasicLSTMCell(rnn_size)
outputs, states = rnn.static_rnn(lstm_cell, x, dtype=tf.float32)

output = tf.matmul(outputs[-1],layer['weights']) + layer['biases']
return output
```

Challenge

Prepare RNN neural network for the Iris data.

Use:

- BasicLSTM cell
- Initialize data with random uniform distribution variables
- Put two rows of the picture as a chunk size
- RNN size should be 56
- Use SGD optimizer

Module 9
Keras

SkFlow

SkFlow has been moved to http://github.com/tensorflow/tensorflow into contrib folder specifically located here. The development will continue there. Please submit any issues and pull requests to Tensorflow repository instead.

This repository will ramp down, including after next Tensorflow release we will wind down code here. Please see instructions on most recent installation here.

TFlearn

Deep learning library featuring a higher-level API for TensorFlow.

TFlearn is a modular and transparent deep learning library built on top of Tensorflow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it.

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow or Theano.

It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

CNN with Keras

```
model = Sequential()

model.add(Convolution2D(nb_filters, kernel_size[0], kernel_size[1],
                        border_mode='valid',
                        input_shape=input_shape))
model.add(Activation('relu'))
model.add(Convolution2D(nb_filters, kernel_size[0], kernel_size[1]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=pool_size))
model.add(Dropout(0.25))
```

CNN with Keras

```
model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adadelta',
              metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch,
          verbose=1)
```

CNN with Keras

```
model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='adadelta',
              metrics=['accuracy'])
model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch,
          verbose=1)
```

Challenge

Prepare CNN using KERAS

- copy the architecture from the Module 7_1
- use siftsign and relu6 activation functions where required
- make sure filter sizes are appropriate
- do not use dropout for this case

RNN with Keras

```
model = Sequential()
model.add(Embedding(max_features, 128, dropout=0.2))
model.add(LSTM(128, dropout_W=0.2, dropout_U=0.2))  # try using a GRU instead, for fun
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=batch_size, nb_epoch=2,
          validation_data=(X_test, y_test))
```

How does people use ML?
Want more?

Q&A Session Feedback

Resources

- More on CNNs:

  https://www.reddit.com/r/MachineLearning/

  https://www.reddit.com/r/deepdream/

  http://karpathy.github.io/
- RNNs:

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

  https://www.youtube.com/watch?v=1PGLj-uKT1w
- Determine location of a random image:

  https://www.technologyreview.com/s/600889/google-unveils-neural-network-with-superhuman-ability-to-determine-the-location-of-almost/

  • Stochastic Depth Networks: http://arxiv.org/abs/1603.09382

Resources

- Determine location of a random image: https://www.technologyreview.com/s/600889/google-unveils-neural-network-with-superhuman-ability-to-determine-the-location-of-almost/
- Stochastic Depth Networks: http://arxiv.org/abs/1603.09382

Resources

- https://github.com/tflearn/tflearn
- https://github.com/nfmcclure/tensorflow_cookbook
- https://pythonprogramming.net/machine-learning-tutorial-python-introduction/
- http://learningtensorflow.com
- https://www.tensorflow.org/
- http://playground.tensorflow.org
- http://terrytangyuan.github.io/2016/03/14/scikit-flow-intro/

How to keep up to date with ML

- Machine learning advances happen more and morefrequently.
- Can’t rely on journals that take multiple years to publish.
- Twitter
- http://arxiv.org/
- Local meetup groups AI/R data analysis/Python analysis/...

Ongoing Development

- Predict age from facial photo - https://how-old.net
- Predict dog breed from photo - https://www.what-dog.net
- Inside a self driving car brain: https://www.youtube.com/watch?v=ZJMtDRbqH40
- Memory networks: https://arxiv.org/pdf/1410.3916.pdf
  - Frodo journeyed to Mount-Doom. Frodo dropped the ring there. Sauron died.Frodo went back to the Shire. Bilbo travelled to the Grey-havens.
  - Where is the ring? A: Mount-Doom
  - Where is Bilbo now? A: Grey-havens
- “Synthia”- Virtual driving school for self driving cars:
- http://www.gizmag.com/synthia-dataset-self-driving-cars/43895/

Resources - Latest publications

- June-July (2016):
  - YodaNN: http://arxiv.org/abs/1606.05487
  - CMS-RCNN: http://arxiv.org/abs/1606.05413
  - Zoneout RNN: https://arxiv.org/abs/1606.01305
  - Pixel CNN: http://arxiv.org/abs/1606.05328
  - Memory CNN: http://arxiv.org/abs/1606.05262
  - DISCO Nets: http://arxiv.org/abs/1606.02556
- More last year (2016):
  - Stochastic Depth Net: https://arxiv.org/abs/1603.09382
  - Squeeze Net: https://arxiv.org/abs/1602.07360
  - Binary Nets: http://arxiv.org/abs/1602.02830