Introduction to image classification with TensorFlow

from: https://cs.stanford.edu/people/karpathy/sfmltalk.pdf

Automated image captioning

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

from: https://deepdreamgenerator.com/ddream/9e14fc746e

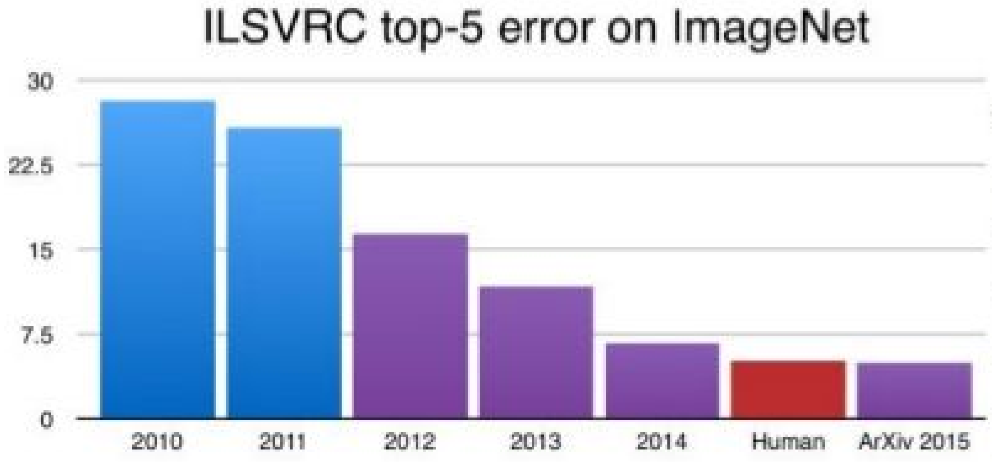
DeepDream

from: http://cs.stanford.edu/people/karpathy/cnnembed/cnn_embed_1k.jpg

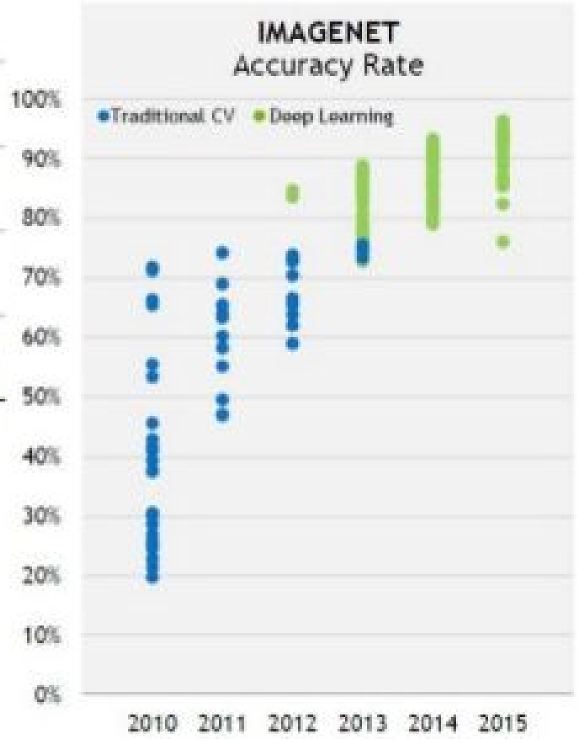
ImageNet

from: https://www.slideshare.net/grigorysapunov/deep-learning-cases-text-and-image-processing

A bit of history

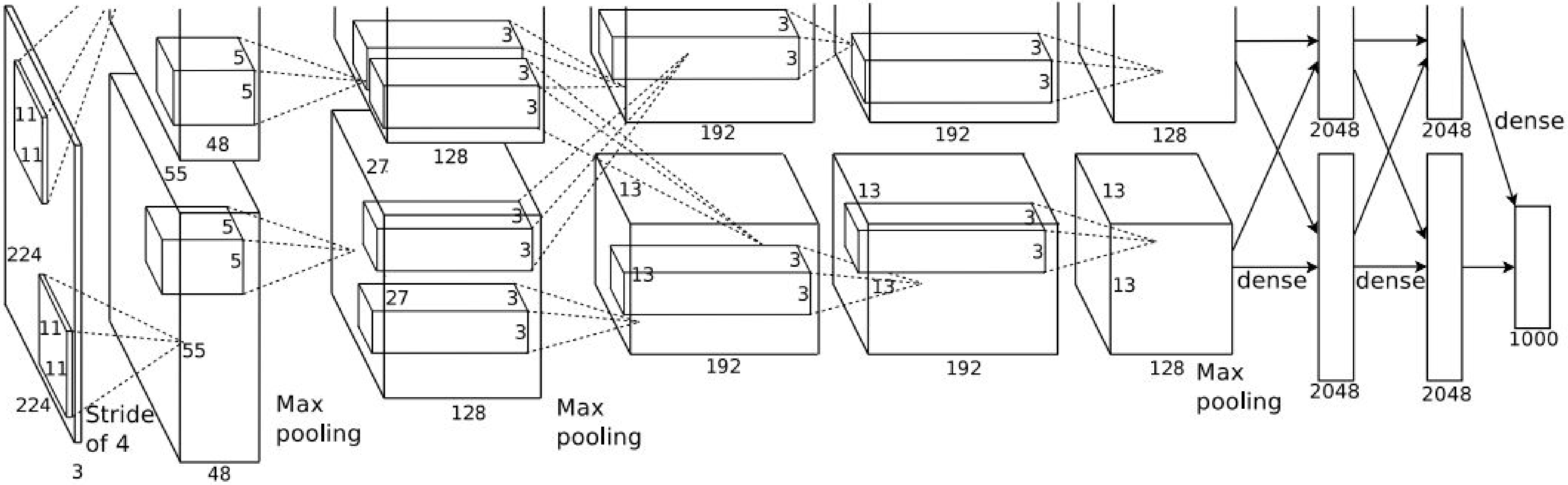


- Blue: Traditional CV
- Purple: Deep Learning
- Red: Human



from: https://people.eecs.berkeley.edu/~shhuang/img/alexnet.png

AlexNet



Our data sets and goals

- MNIST
- CIFAR10
- Simple code, simple models

MNIST

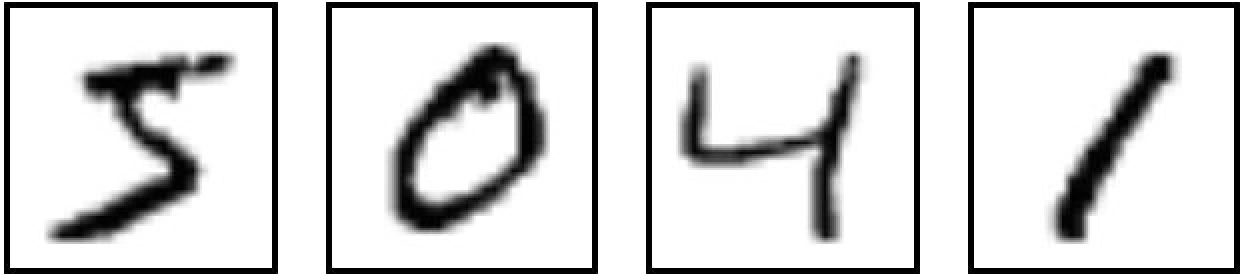


Image from: https://www.tensorflow.org/get_started/mnist/beginners
Data set from: http://yann.lecun.com/exdb/mnist/
Detailed analysis: http://colah.github.io/posts/2014-10-Visualizing-MNIST/

MNIST: key facts

- 60,000 + 10,000 images
- 10 classes
- 28x28x1
- Best error rate: 0.21%

CIFAR10

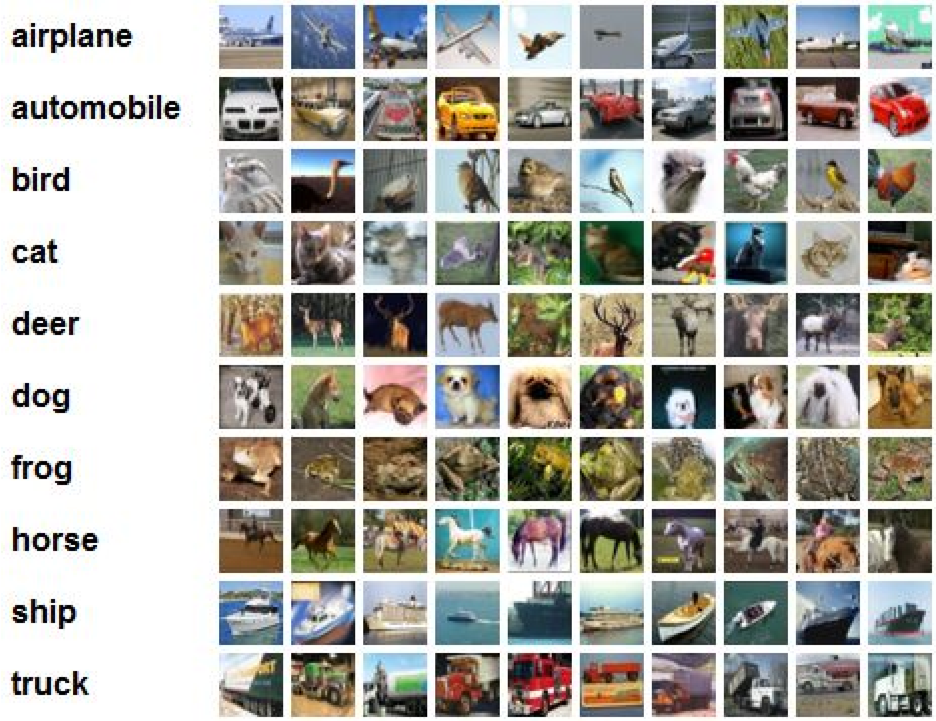


Image from:
http://karpathy.github.io/assets/cifar_preview.png
Data set from:
https://www.cs.toronto.edu/~kriz/cifar.html

CIFAR10: key facts

- 50,000 + 10,000 images
- 10 classes
- 32x32x3
- Best error rate: 3.47%

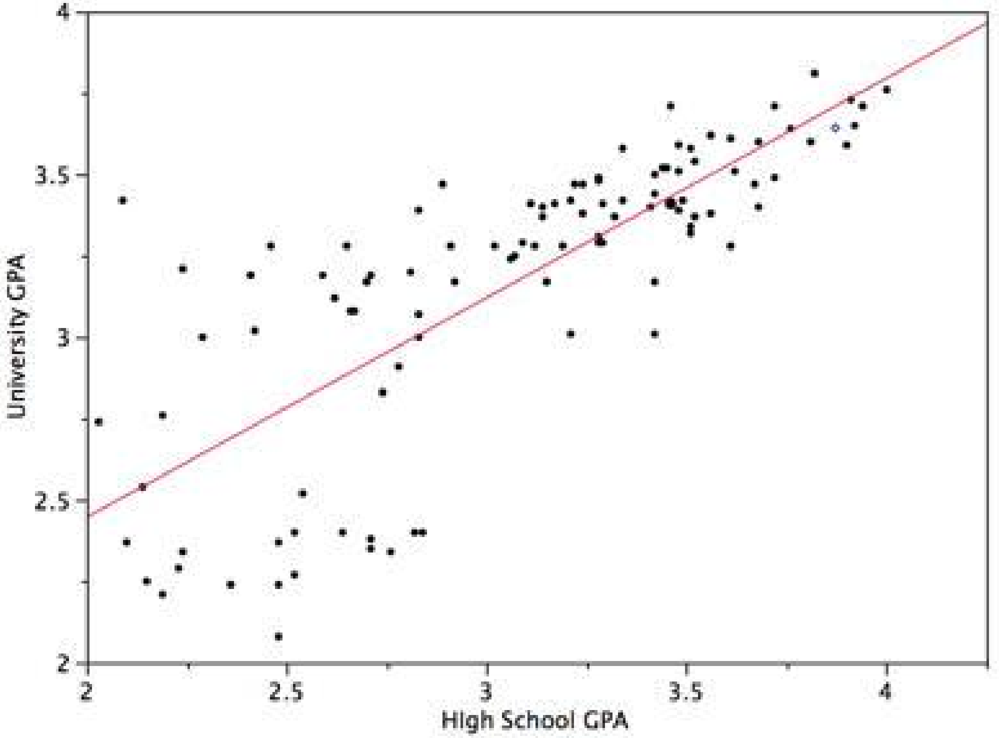
TensorFlow

- Open source library for building and executing computation graphs.
- Cross platform and quite scalable.
- Provides high-level APIs and tools for typical ML tasks.

- Only Python APIs for training (so far)
- Some APIs still not stable
- Some features require building TF from sources

Note: in context of TF, tensor == multidimensional array

Linear regression



Linear function

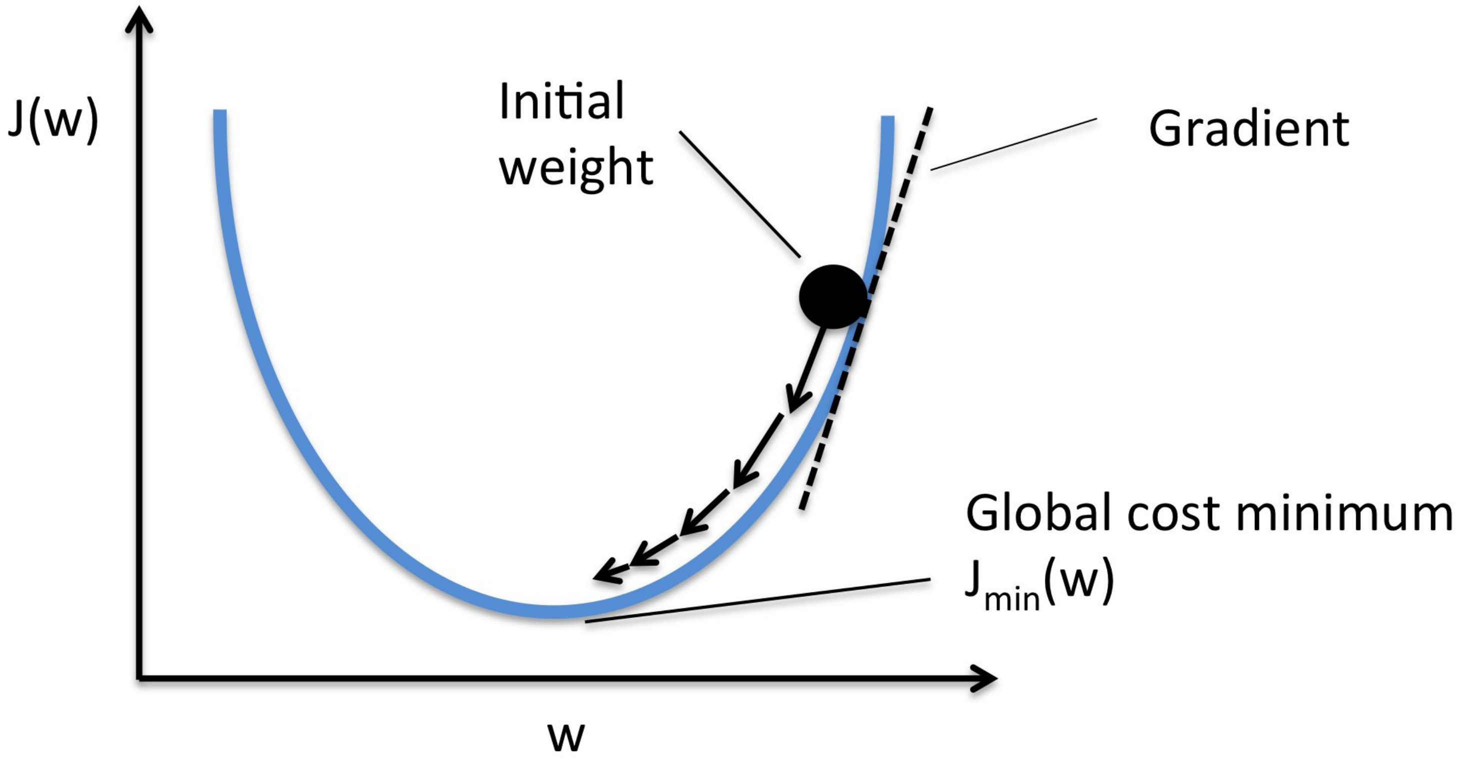
```
with tf.Session() as sess:
    # Parameters
    W = tf.Variable([2.0], tf.float32)
    b = tf.Variable([1.0], tf.float32)
    # Input
    x = tf.placeholder(tf.float32)
    y = W * x + b
```

Loss function

```
# correct output
y_ = tf.placeholder(tf.float32)
# loss
loss = tf.reduce_sum(tf.square(y - y_))
```

Image from: https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html

Gradient descent



Optimizer

```
optimizer = tf.train.GradientDescentOptimizer(0.01)

train = optimizer.minimize(loss)
```

Train

```
# training data
x_train = [1,2,3,4]
y_train = [0,-1,-2,-3]
# training loop
init = tf.global_variables_initializer()
sess.run(init)
for i in range(1000):
  sess.run(train, {x:x_train, y:y_train})
```

Evaluate

```
W_v, b_v, loss_v = sess.run(
    [W, b, loss], {x:x_train, y:y_train})
print("W: {} b: {} loss: {}".format(W_v, b_v, loss_v))
```

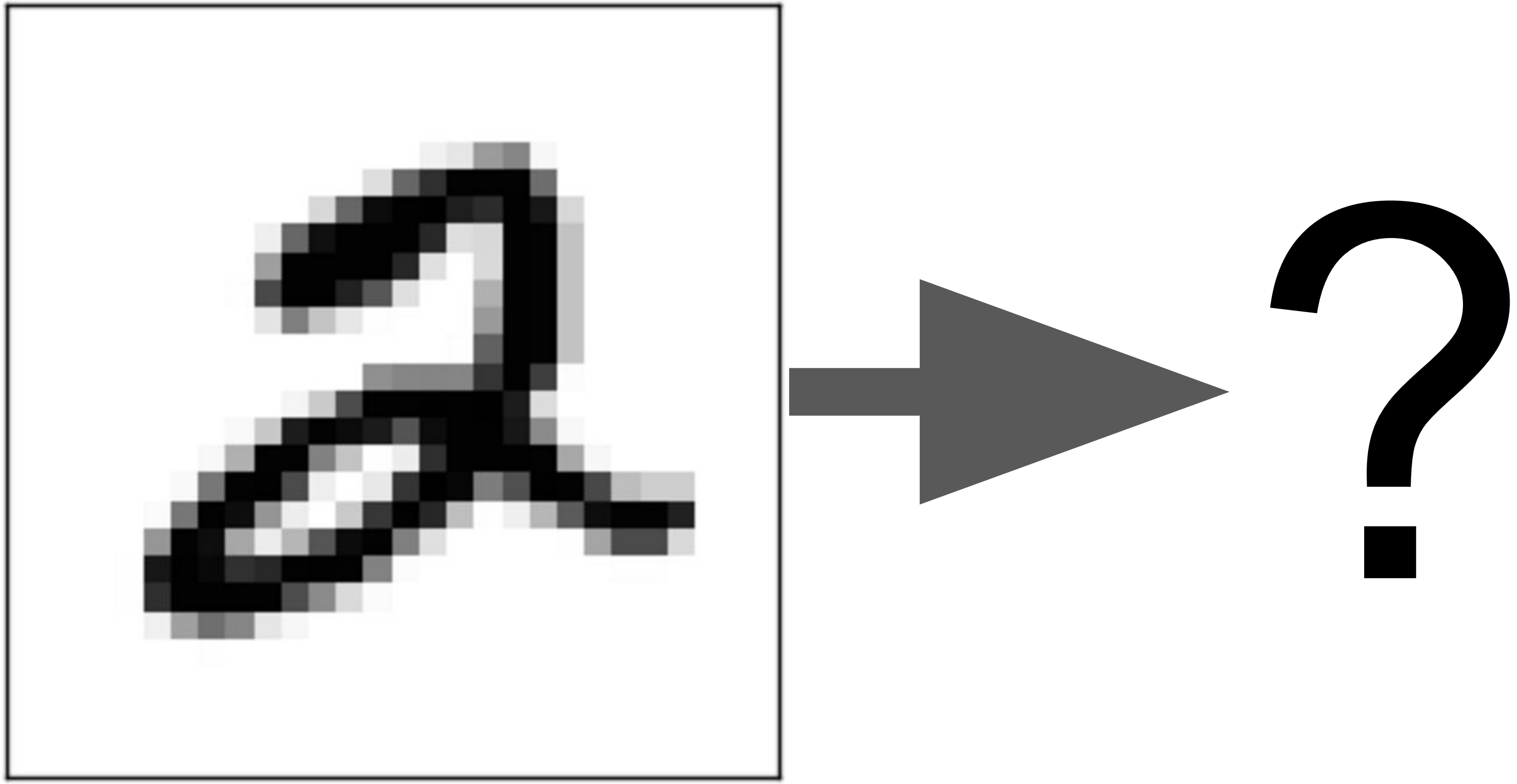
Example output:

W: [-0.99999464] b: [0.9999842] loss: 1.66839e-10

Estimator API

```
features = [tf.contrib.layers.real_valued_column( "x", dimension=1)]

estimator = tf.contrib.learn.LinearRegressor( feature_columns=features)

x = np.array([1., 2., 3., 4.])

y = np.array([0., -1., -2., -3.])

input_fn = tf.contrib.learn.io.numpy_input_fn({ "x":x}, y, batch_size=4,
num_epochs=1000)

estimator.fit(input_fn=input_fn, steps=1000)

estimator.evaluate(input_fn=input_fn)
```

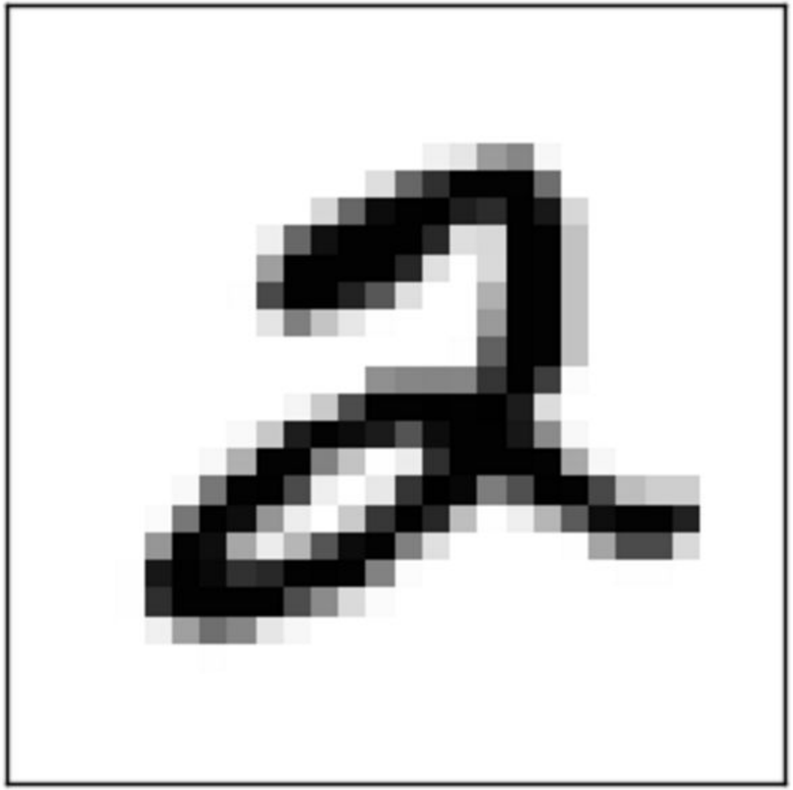
Classification



Input representation

```
x = tf.placeholder(tf.float32, [None, 784])
```

Labels



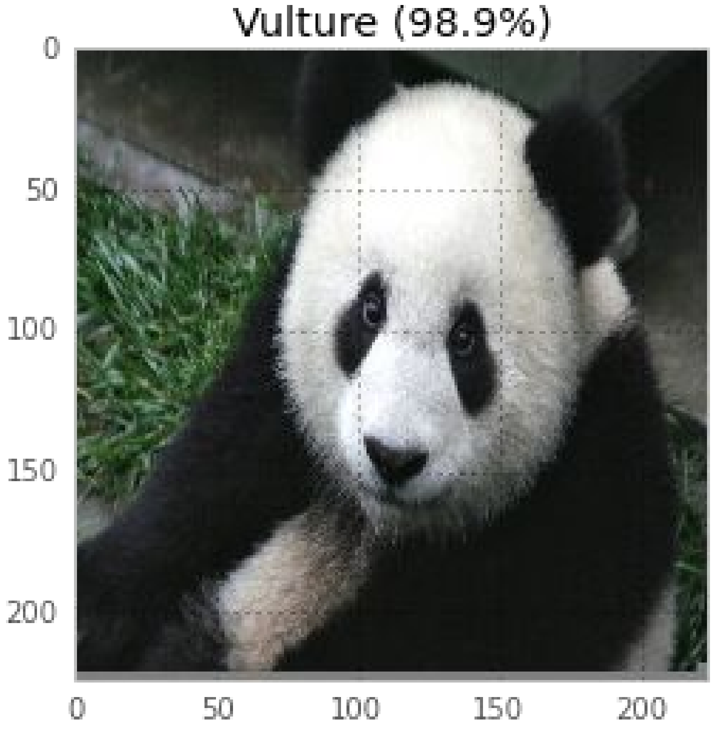
0
0
1
0
0
0
0
0
0
0

Labels in TensorFlow

```
y_ = tf.placeholder(tf.float32, [None, 10])
```

Image from: https://codewords.recurse.com/images/five/neural-networks/panda_vulture.png

Accuracy

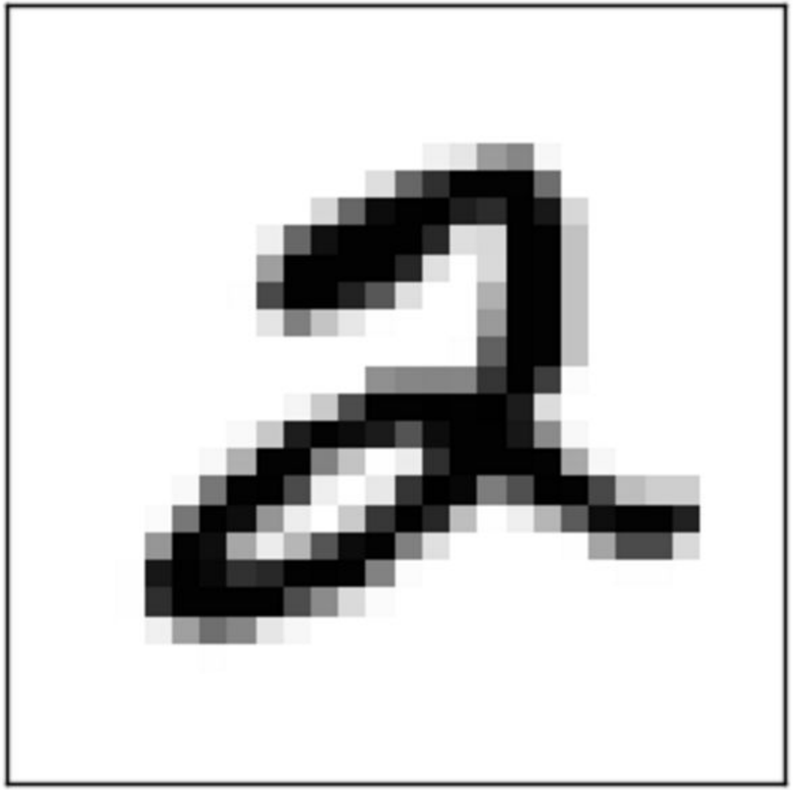


Accuracy

- Rate of correctly classified examples
- Good metric for balanced classes

- Can't be easily used as loss function

Use regression for each class?

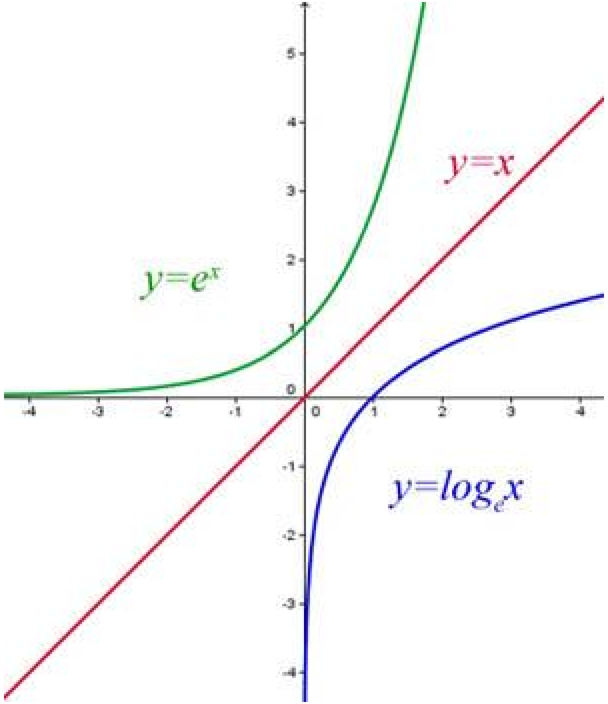


1.5
0.7
100
4
-3
4
32
6
56
49

Linear Regression (again)

```
W = tf.Variable(tf.zeros([784, 10]))

b = tf.Variable(tf.zeros([10]))

y = tf.matmul(x, W) + b
```

Exponential and logarithmic functions



Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Softmax in TensorFlow

```
y_softmax = tf.nn.softmax(y)
```

Cross-entropy loss

$$H(p, q) = -\sum_{x} p(x)\, \log q(x)$$

More on information theory: http://colah.github.io/posts/2015-09-Visual-Information/

Cross-entropy loss in TensorFlow

```
cross_entropy =
    tf.reduce_mean(
        -tf.reduce_sum(y_ * tf.log(y_softmax),
                      reduction_indices=[1]))
```

Cross-entropy loss in TensorFlow

```
cross_entropy = tf.reduce_mean(
    tf.nn.softmax_cross_entropy_with_logits(labels=y_,
                                            logits=y))
```

Getting the data

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

Training

```
optimizer = tf.train.AdagradOptimizer(0.5)
train_step = optimizer.minimize(cross_entropy)
sess = tf.InteractiveSession()
tf.global_variables_initializer().run()

for _ in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

Evaluation

```
correct_prediction = tf.equal(tf.argmax(y,1),
                              tf.argmax(y_,1))
accuracy = tf.reduce_mean(
                 tf.cast(correct_prediction, tf.float32))
print(sess.run(accuracy,
               feed_dict={
                   x: mnist.test.images,
                   y_: mnist.test.labels}))
```

Accuracy

- MNIST: 0.9144
- CIFAR10: 0.279

Overfitting

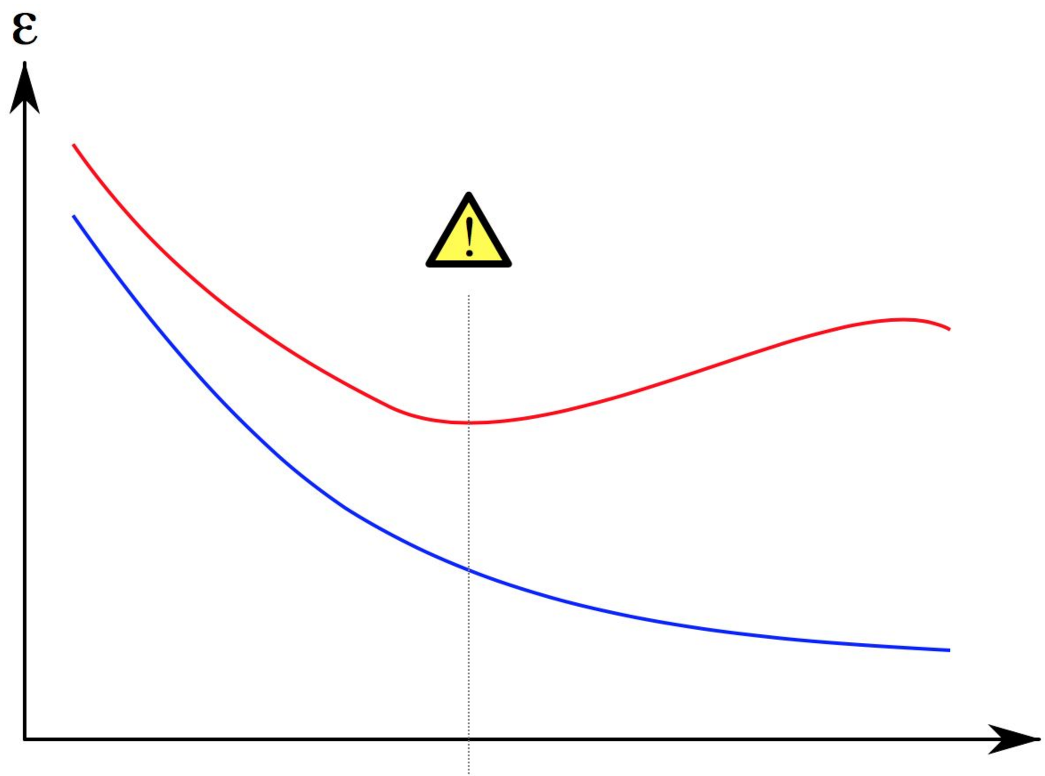


Image from: https://www.tensorflow.org/get_started/mnist/beginners

Model introspection: MNIST

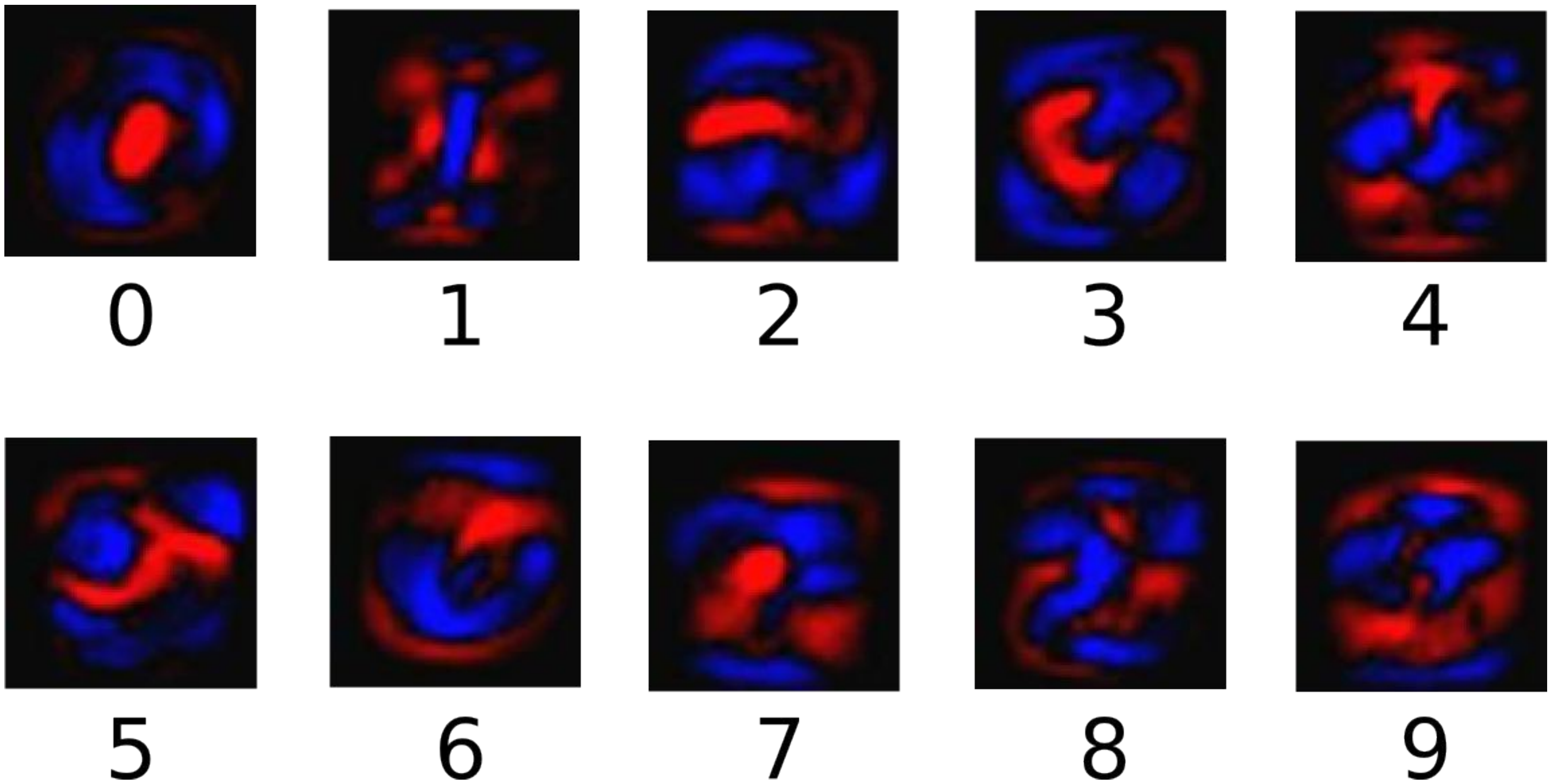


Image from: http://cs231n.github.io/linear-classify/

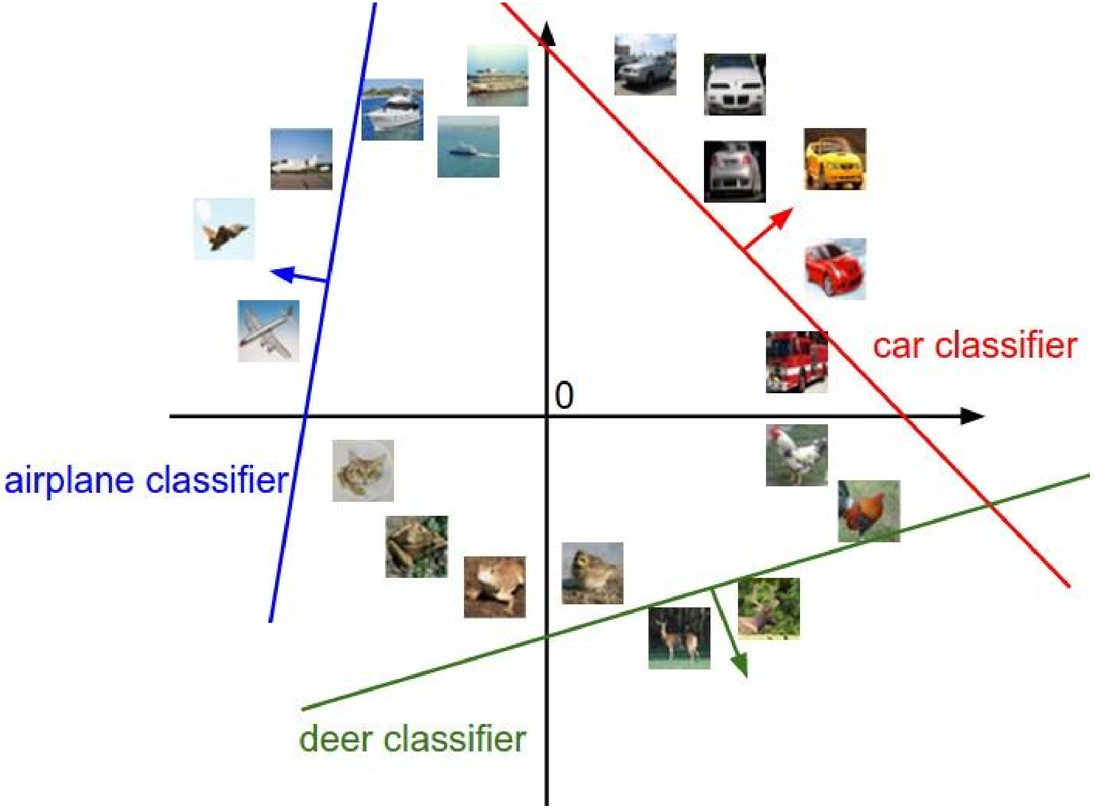
Model introspection: CIFAR10

Further reading

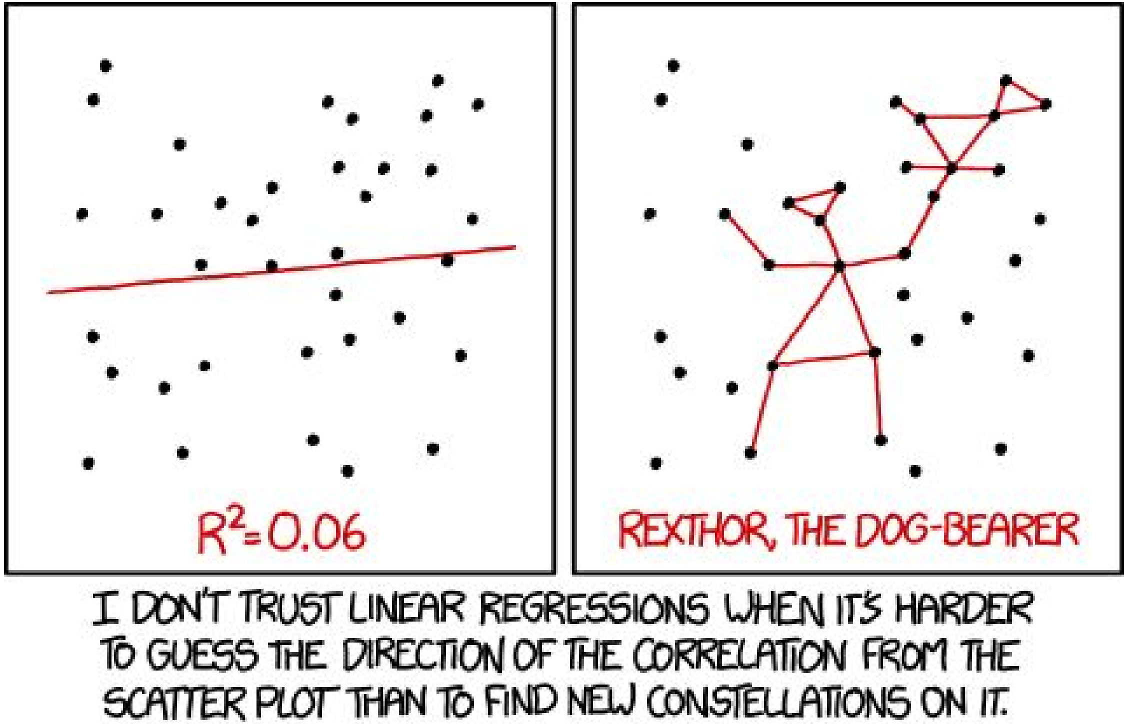
- Linear classification for MNIST: https://www.tensorflow.org/get_started/mnist/beginners
- Linear classification for CIFAR10: http://cs231n.github.io/linear-classify/

Image from: http://cs231n.github.io/linear-classify/

Linear separability



The world is not linear



Feature engineering

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal lines

-1	2	-1
-1	2	-1
-1	2	-1

Vertical lines

-1	-1	2
-1	2	-1
2	-1	-1

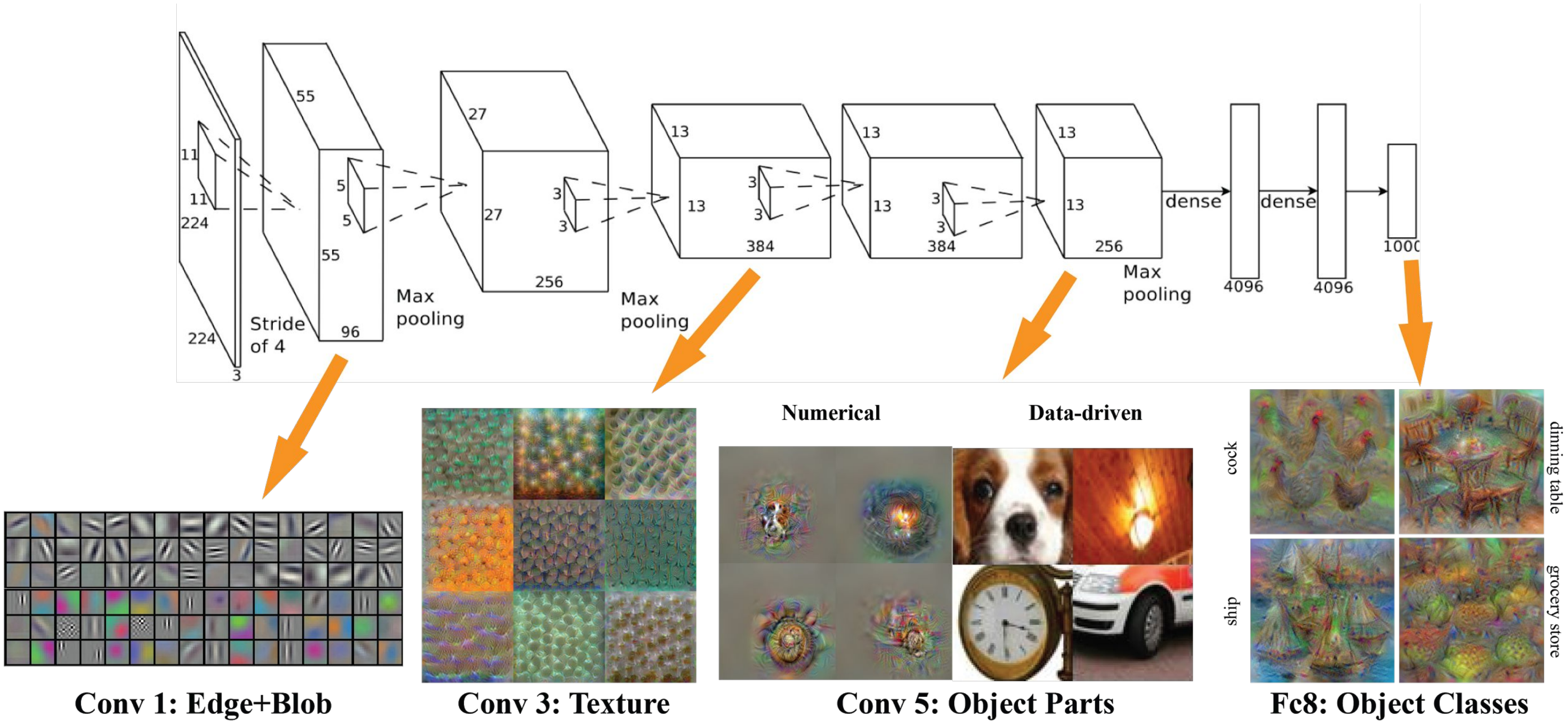
45 degree lines

2	-1	-1
-1	2	-1
-1	-1	2

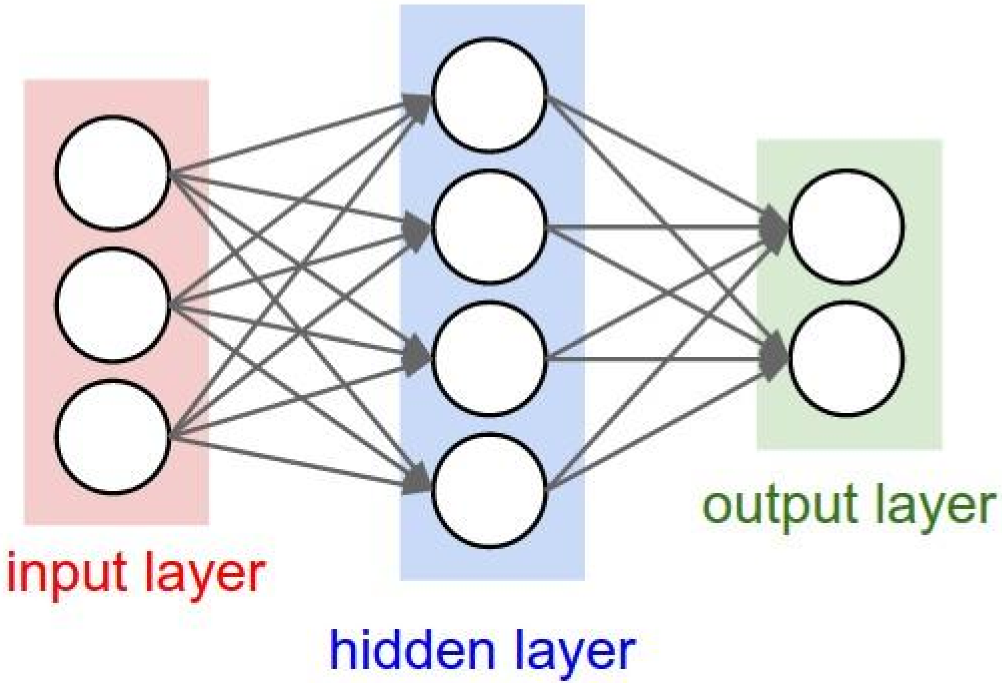
135 degree lines

Image from: http://www.cc.gatech.edu/~hays/compvision/proj6/deepNetVis.png

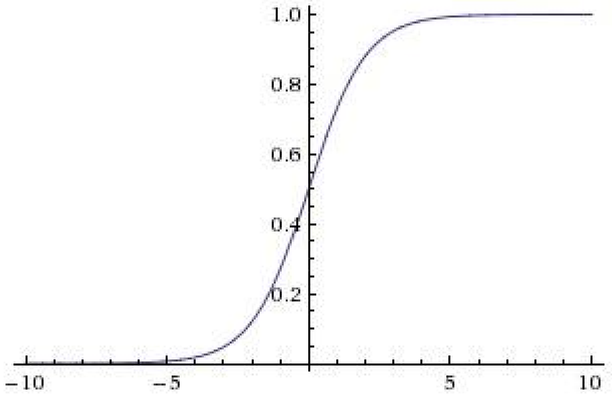
Representation learning

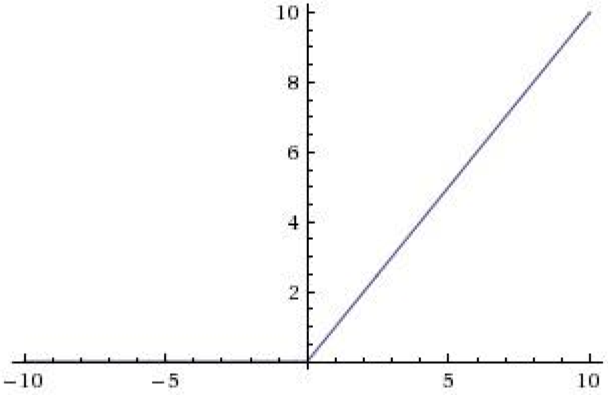


Neural Networks



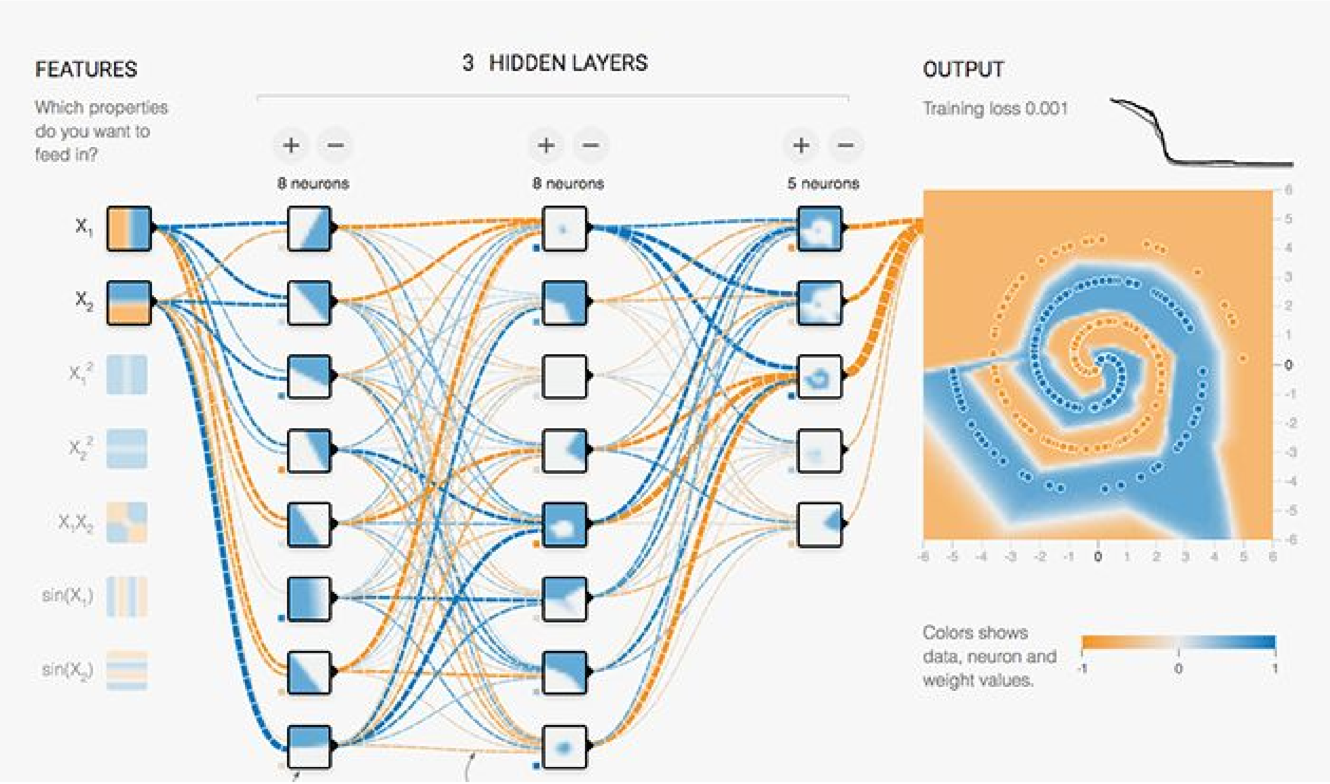
Activation functions





Neural Network Playground

https://playground.tensorflow.org



also, http://karpathy.github.io/neuralnets/

ReLU layer in TF

```
W_fc1 = tf.truncated_normal(shape, stddev=0.1)
b_fc1 = tf.constant(0.1, shape=shape)
h_fc1 = tf.nn.relu(tf.matmul(h_fc0, W_fc1) + b_fc1)
```

Or use **tf.contrib.layers.fully_connected**

Dropout

```
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

Convolution

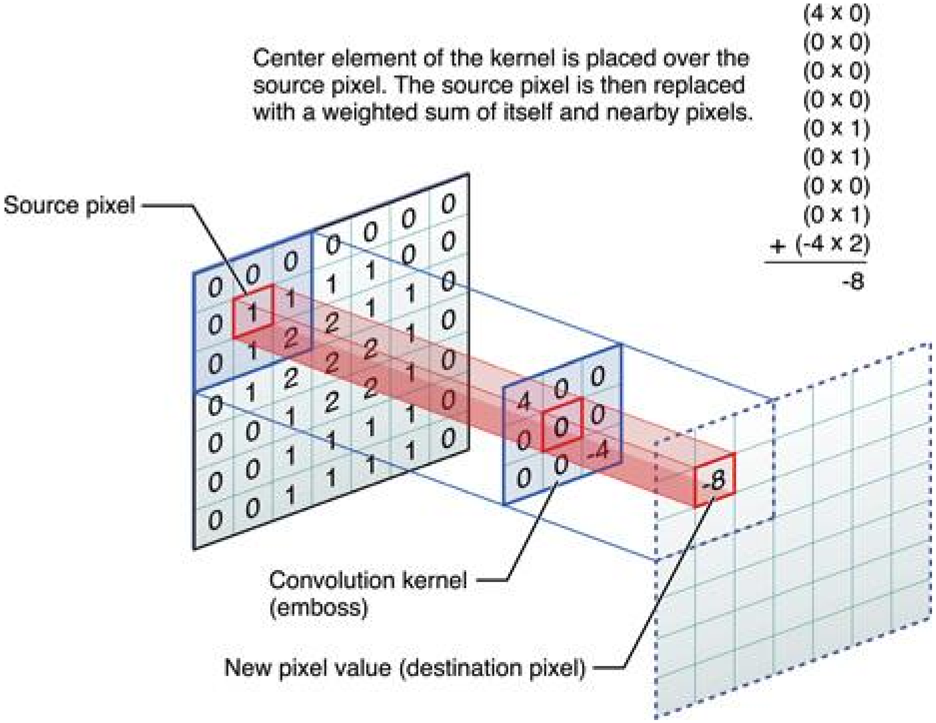
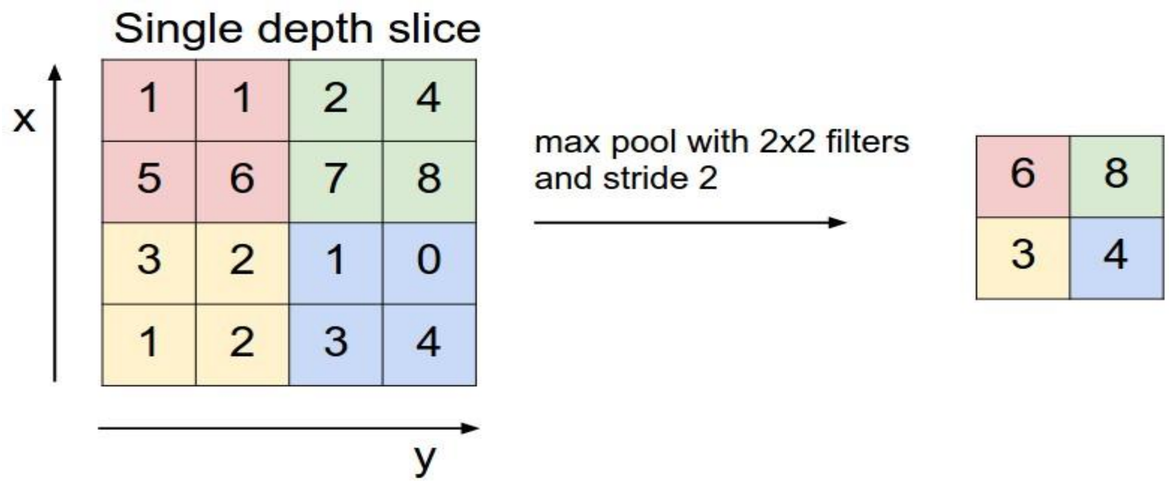


Image from: https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

Image from: http://cs231n.github.io/convolutional-networks/

Pooling



Convolution and pooling in TensorFlow

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


def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                        strides=[1, 2, 2, 1],
                        padding='SAME')
```

Convolution and pooling in TensorFlow

```
x_image = tf.reshape(x, [-1,28,28,1])

h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)

h_pool1 = max_pool_2x2(h_conv1)
```

Simple CNN accuracy

INPUT -> [CONV -> RELU -> POOL]*2 -> FC -> RELU -> FC

- MNIST: 0.9915
- CIFAR10: 0.7353

Further reading

- Demo network for CIFAR10: http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html
- Deep MNIST for experts: https://www.tensorflow.org/get_started/mnist/pros
- CNN with tf.contrib.learn API: https://www.tensorflow.org/tutorials/layers
- A more complex tutorial for CIFAR10 (~86% accuracy): https://www.tensorflow.org/tutorials/deep_cnn

What next?

- Andrew Ng’s course on Machine Learning: https://www.coursera.org/learn/machine-learning
- Ng’s Stanford course CS229 http://cs229.stanford.edu
- Geoffrey Hinton’s course on Neural Networks: https://www.coursera.org/learn/neural-networks
- Vincent Vanhoucke’s course on Deep Learning with TensorFlow: https://www.udacity.com/course/deep-learning--ud730

What next?

- CS231n: Convolutional Neural Networks for Visual Recognition: http://cs231n.stanford.edu
- CS224d: Deep Learning for Natural Language Processing: http://cs224d.stanford.edu
- TensorFlow tutorials: https://www.tensorflow.org/tutorials
- http://learningtensorflow.com
- Kaggle

Thanks!