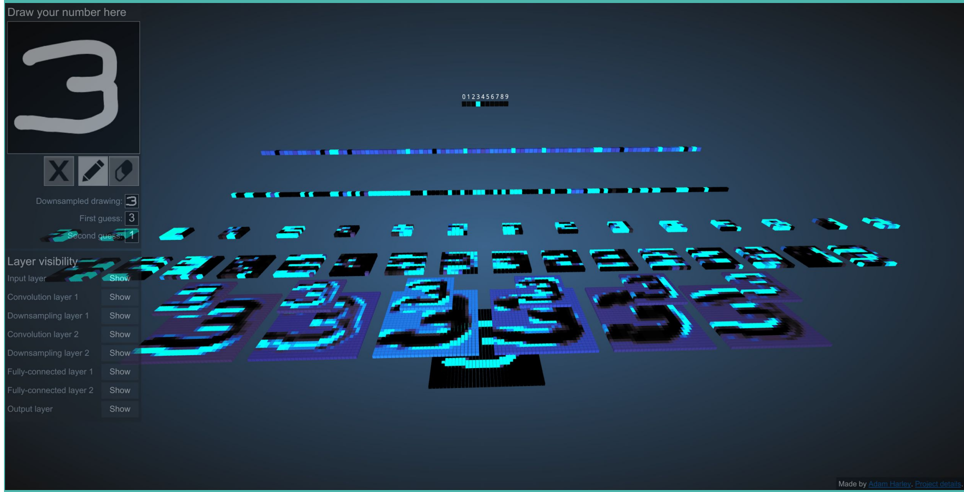
Convolutional Neural Networks

+ Tensorflow Introduction



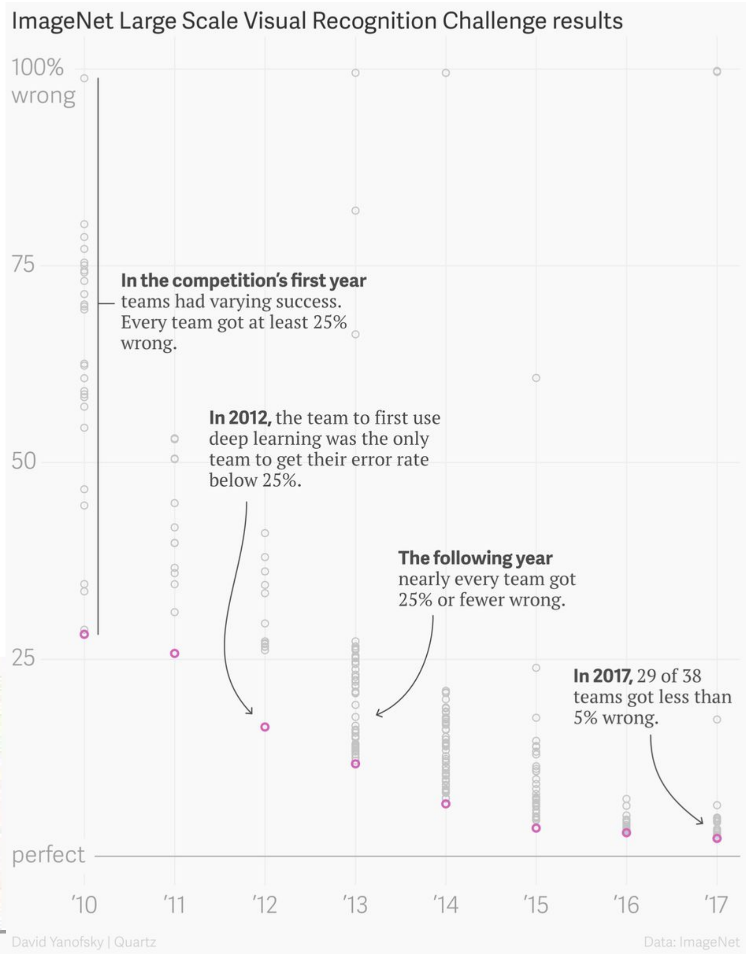
http://scs.ryerson.ca/~aharley/vis/

Imagenet (Fei-Fei Li, Stanford, 2006)

1990's [LeNet]: LeCune uses neural networks to read zip codes

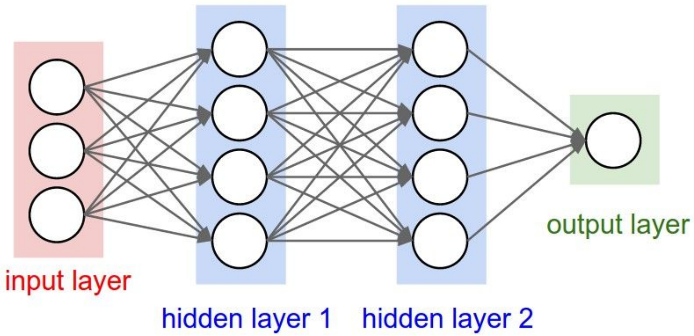
2012 [AlexNet]: Blows away the competition in visual classification using ConvNets

2013-2017 [ZF Net, GoogLeNet, VGGNet, ResNet]: ConvNets continue to evolve to human levels of accuracy

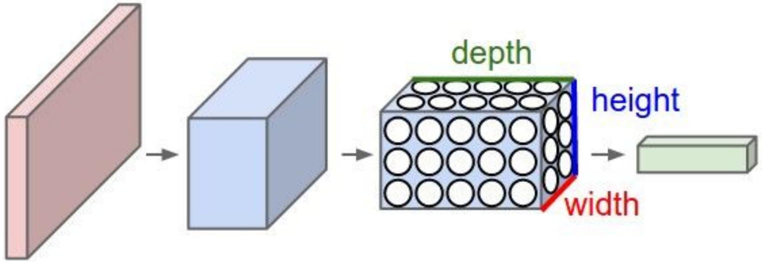


https://qz.com/1077484/brexit-is-over-britain-needs-to-rebuild-this-school-may-hold-some-answers/

Fully connected vs. convolutional architecture



A typical 3 layer fully connected network



A convolutional network, the individual layers are a 3D tensor of shape:
`[kernel_height, kernel_width, features]`

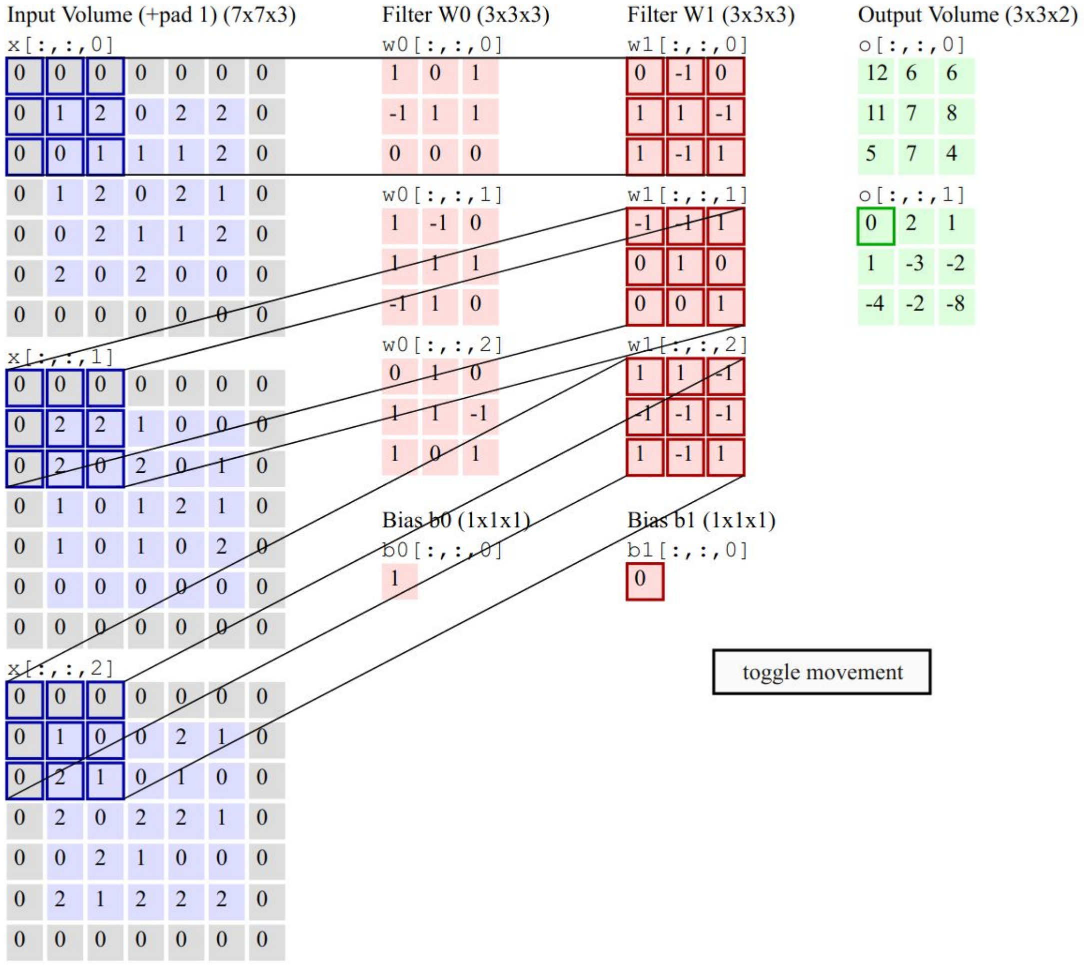
Note: weights are not depicted

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

Convolutions

Stanford's online CS231n:

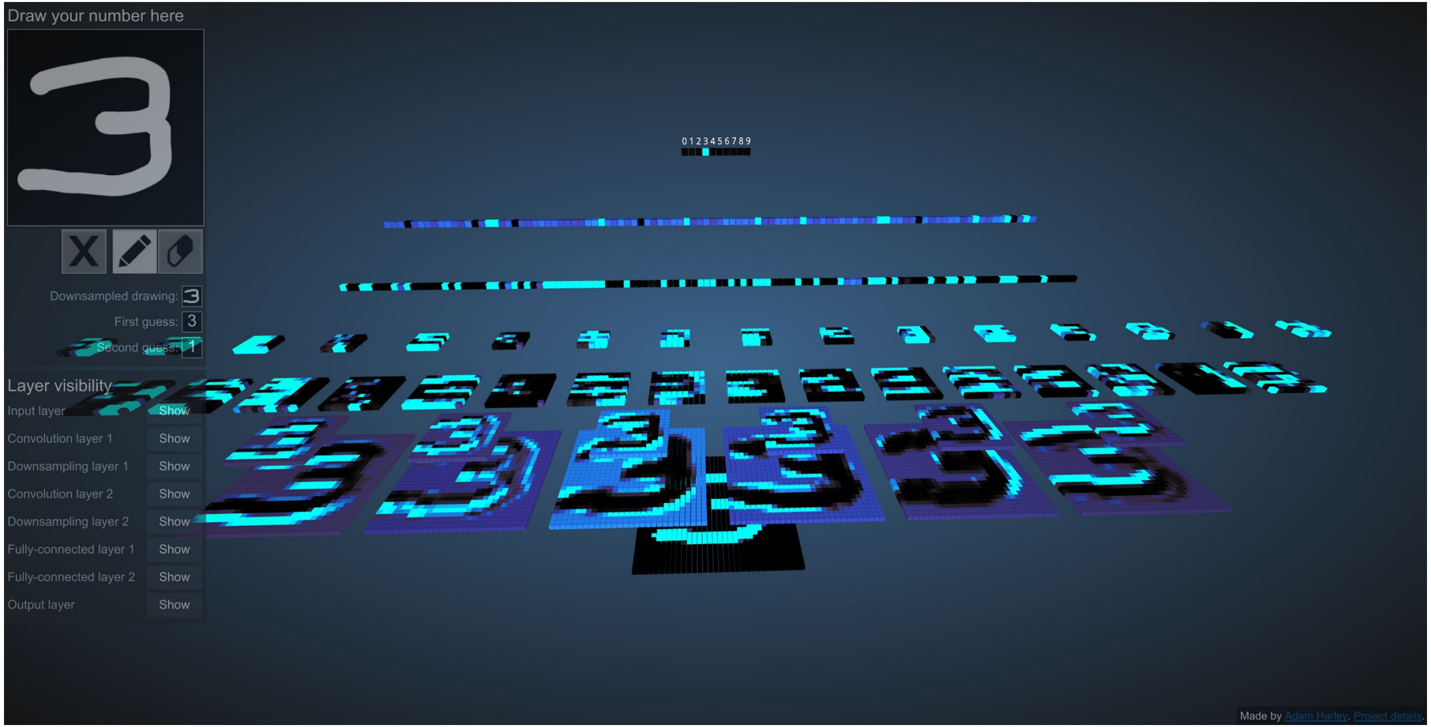
http://CS231n.github.io/convolutional-networks/



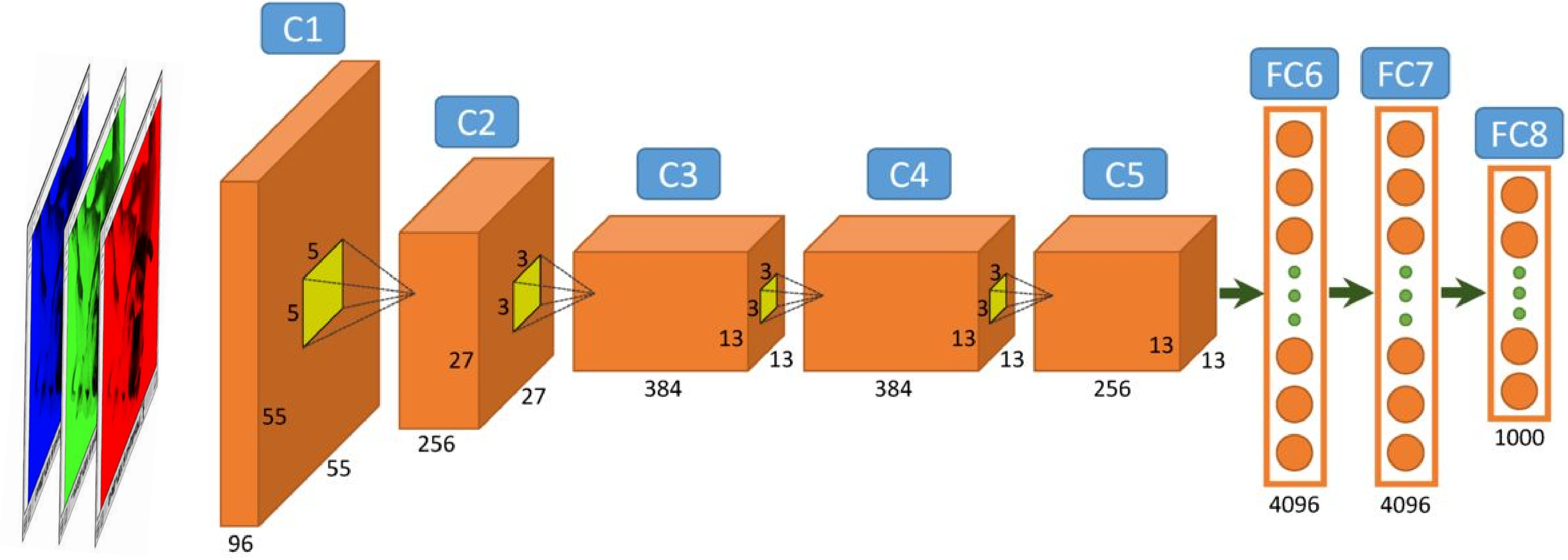
http://cs231n.github.io/convolutional-networks/

3D visualization of convolutional network

http://scs.ryerson.ca/~aharley/vis/



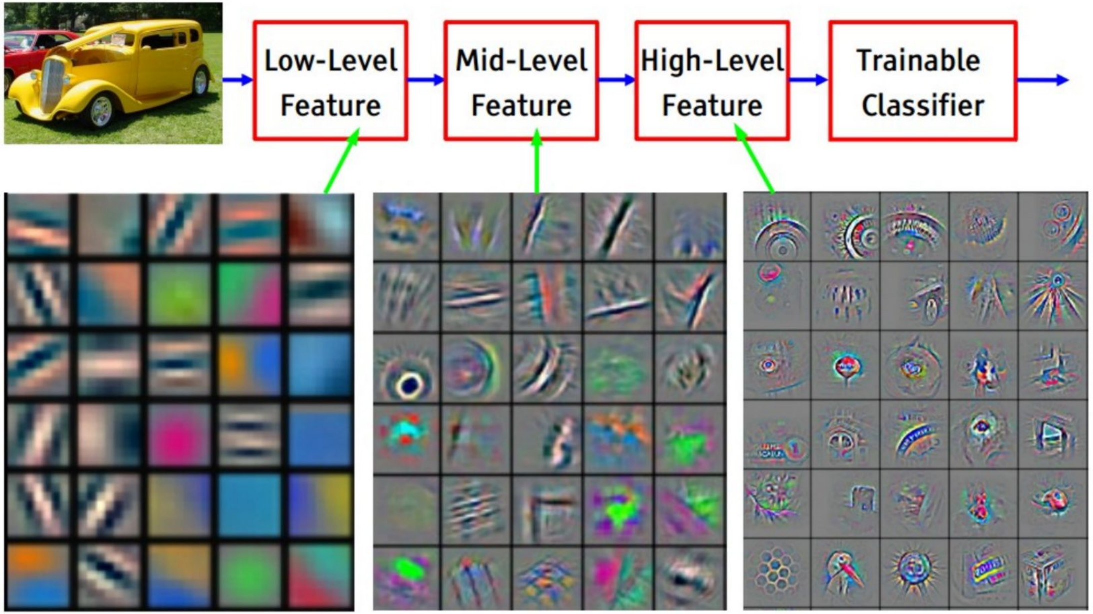
Reading CNN diagrams



http://www.mdpi.com/2072-4292/7/11/14680/htm https://sites.google.com/site/learnphotographybasics/Home/color-bit-depth-channels-info-pallet

Visualizing what was learned by the network

[From recent Yann LeCun slides]



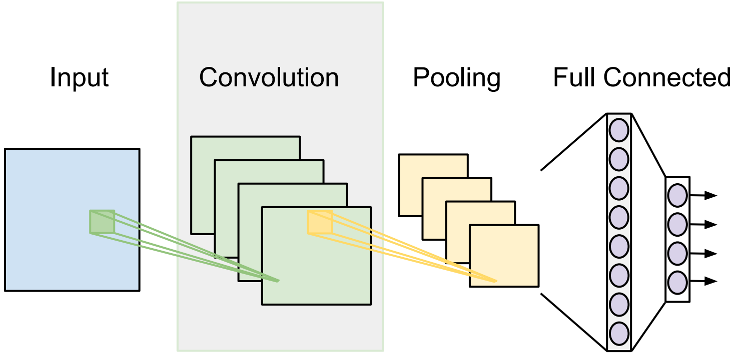
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

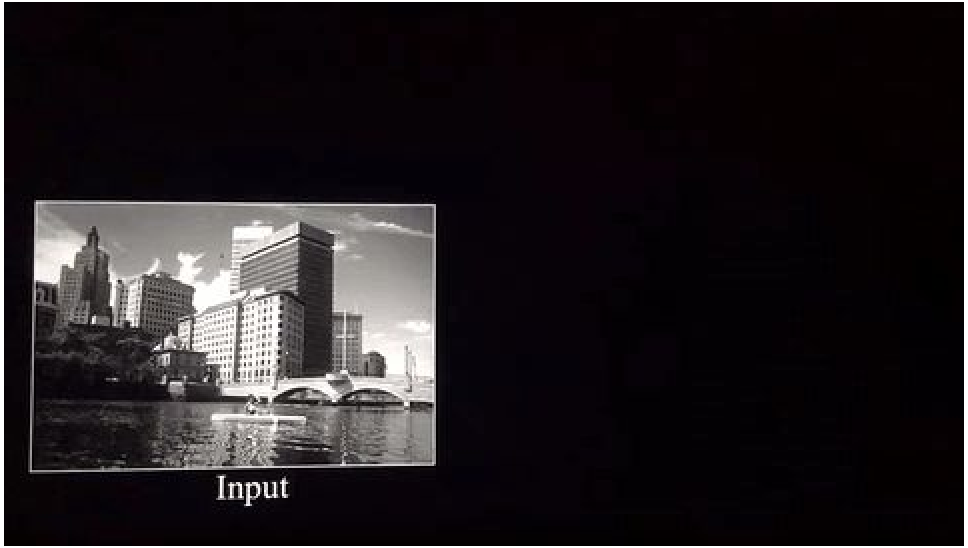
Convolutional filters

The Convolution layer uses kernels to extract patterns from the image.

The kernels are learned during training

Pooling is a fancy method of downsampling





https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

DeepViz

Open source visualization tool based on a number of papers circa 2014

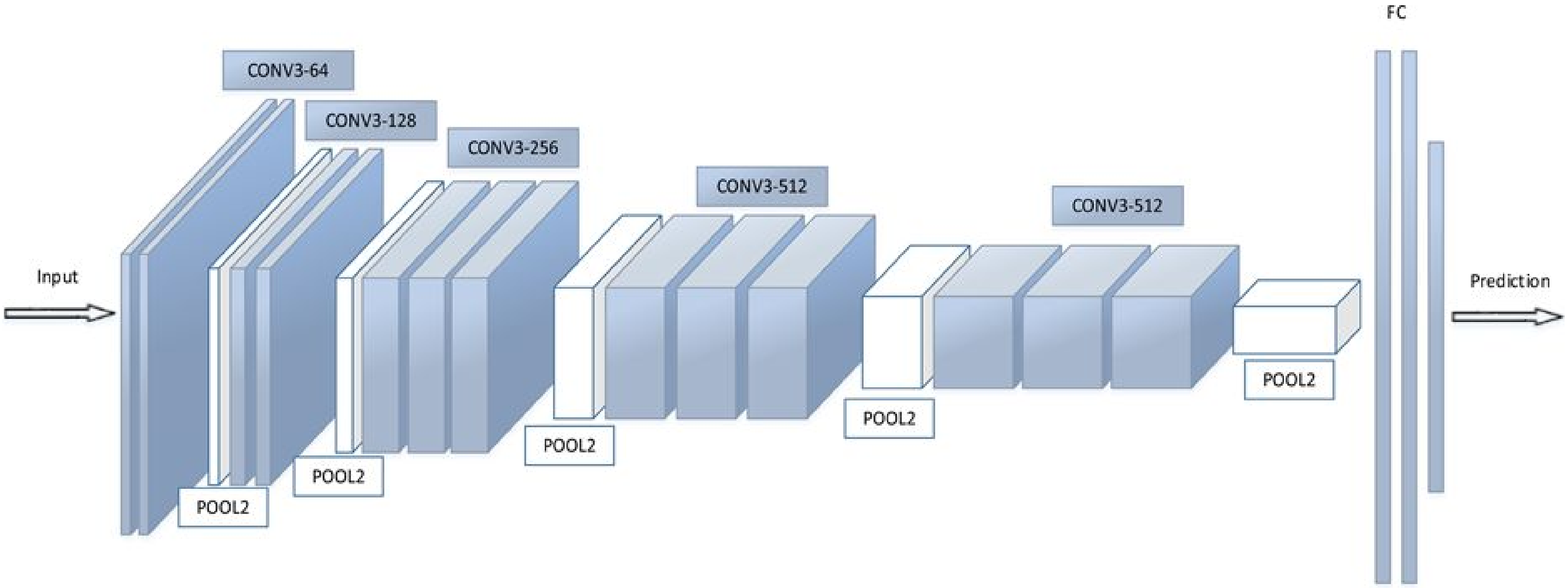
“Deep Visualization Toolbox” (4min)

“Visualizing and Understanding Deep Neural Networks by Matt Zeiler” (1hr)

0:49 (light -> dark filter)
1:00 (layer 1 filters)
2:33 (conv5 wrinkles filter)
3:05 (conv5 text filter)

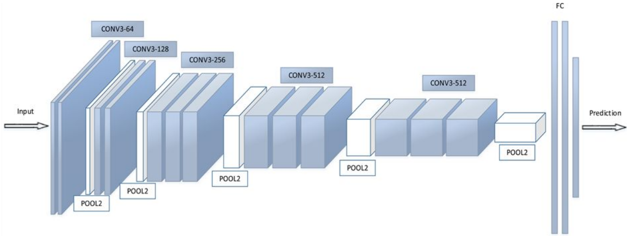
http://yosinski.com/deepvis - (4 min version) https://www.youtube.com/watch?v=AgkfIQ4IGaM - (1hr version) https://www.youtube.com/watch?v=ghEmQSxT6tw&t=1295s

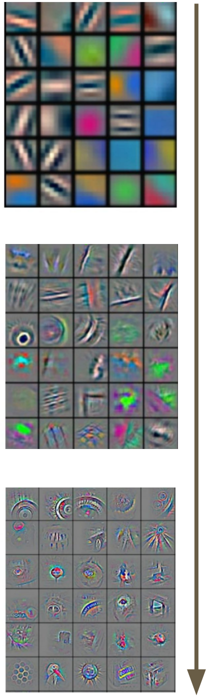
VGGNet case study (ILSVRC 2014 runner up)



http://file.scirp.org/Html/4-7800353_65406.htm
http://cs231n.github.io/convolutional-networks/

VGGNet case study (ILSVRC 2014 runner up)





```
INPUT:     [224x224x3]   memory:  224*224*3=150K    weights: 0
CONV3-64:  [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*3)*64 = 1,728
CONV3-64:  [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*64)*64 = 36,864
POOL2:     [112x112x64]  memory:  112*112*64=800K   weights: 0
CONV3-128: [112x112x128] memory:  112*112*128=1.6M  weights: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory:  112*112*128=1.6M  weights: (3*3*128)*128 = 147,456
POOL2:     [56x56x128]   memory:  56*56*128=400K    weights: 0
CONV3-256: [56x56x256]   memory:  56*56*256=800K    weights: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]   memory:  56*56*256=800K    weights: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]   memory:  56*56*256=800K    weights: (3*3*256)*256 = 589,824
POOL2:     [28x28x256]   memory:  28*28*256=200K    weights: 0
CONV3-512: [28x28x512]   memory:  28*28*512=400K    weights: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]   memory:  28*28*512=400K    weights: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]   memory:  28*28*512=400K    weights: (3*3*512)*512 = 2,359,296
POOL2:     [14x14x512]   memory:  14*14*512=100K    weights: 0
CONV3-512: [14x14x512]   memory:  14*14*512=100K    weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]   memory:  14*14*512=100K    weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]   memory:  14*14*512=100K    weights: (3*3*512)*512 = 2,359,296
POOL2:     [7x7x512]     memory:  7*7*512=25K       weights: 0
FC:        [1x1x4096]    memory:  4096              weights: 7*7*512*4096 = 102,760,448
FC:        [1x1x4096]    memory:  4096              weights: 4096*4096 = 16,777,216
FC:        [1x1x1000]    memory:  1000              weights: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters
```

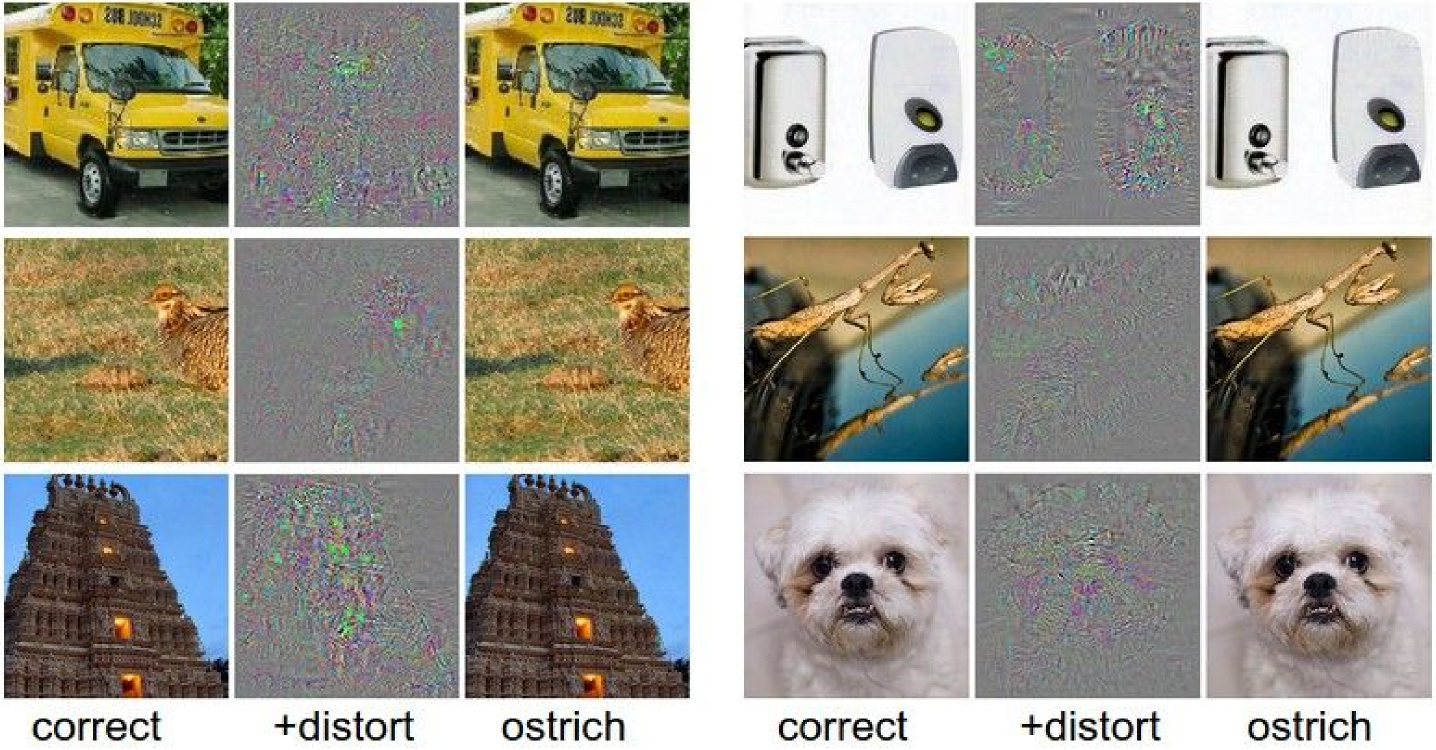
http://cs231n.github.io/convolutional-networks/

Question

Convolutional networks changed the world
(thank the Imagenet competition)

Why don’t fully connected networks perform as well as ConvNets?

Fooling the network (optional)



https://arxiv.org/pdf/1412.6572.pdf - https://arxiv.org/pdf/1312.6199.pdf - http://karpathy.github.io/2015/03/30/breaking-convnets/

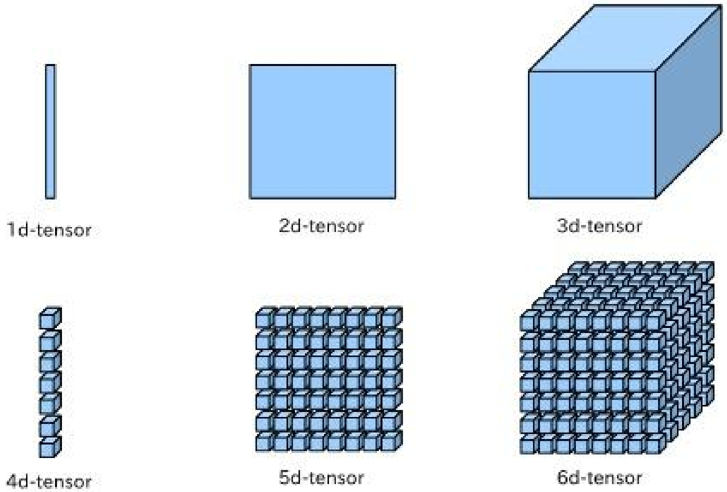
Further Resources

Convolutional Neural Networks:

- neuralnetworksanddeeplearning.com
- Stanford's CS231n online deep learning course
- Deep Learning Book, MIT press (online & in print)
- Neural Network Playground: playground.tensorflow.org

Tensorflow

What is a “Tensor”



The most common error in tensorflow:

```
ValueError: Shape must be rank 2 but is rank 3 for 'MatMul'
(op: 'MatMul') with input shapes: [10,512,1], [512,2]
```

Rank of a tensor	Math Entity	Example
0	Scalar	`x = 42`
1	Vector	`z = [10, 15, 20]`

2	Matrix	`a = [[1 0 2 3],`
		`[2 1 0 4],`
		`[0 2 1 1]]`

3	3-Tensor (a cube of numbers)	A single image of shape `[height, width, color_channels]` example: `[1080, 1920, 3]`
4	4-Tensor (a set of cubes)	A batch of n images with shape `[batch_size, height, width, channels]` example: `[10, 1080, 1920, 3]`
n	n-dimensional Tensor	You get the idea...

https://www.slideshare.net/yokotatsuya/principal-component-analysis-for-tensor-analysis-and-eeg-classification

Tensorflow is a math library

A general purpose math library created for gradient based operations

- Gradients are computed automatically
  - You can trivially ask for the gradient of any variable w.r.t. another
- Automatically handles optimized computation on GPU or CPU seamlessly
- Tensorflow is a symbolic computing library, not an imperative language
  - `if` and `while` loops don't function the same
- Provides a comprehensive set of operations used in constructing neural networks
- TF is NOT limited to neural networks!
- Tools for Visualization, Debugging, Profiling

Tensorflow constructs:

- Placeholders (Inputs)
- Tensors (Mutable variable)
- OPS (Operations)
- Constants

- Computation Graph data structure
- Session

Datatypes:

- tf.float32 is standard
- Supports many data types similar to numpy

Basic workflow for Tensorflow

1 Build a computation Graph

```
import tensorflow as tf

a = tf.constant(2.0, tf.float32, name='a')
b = tf.constant(3.0, tf.float32, name='b')

c = tf.multiply(a, b)
```

```
<tf.Tensor 'a:0' shape=() dtype=float32>
<tf.Tensor 'b:0' shape=() dtype=float32>

<tf.Tensor 'Mul_2:0' shape=() dtype=float32>
```

2 Run computations on the graph

```
sess = tf.Session()
result = sess.run([b, c])
```

```
print(result[0])
3.0

print(result[1])
6.0
```

Passing parameters to tensorflow

```python
import tensorflow as tf

# Build your graph
a = tf.placeholder(tf.float32, shape=(),  name='a')   # A scalar
b = tf.placeholder(tf.float32, shape=(2), name='b')   # A vector
c = tf.multiply(a, b, name='c')

# Launch a session & evaluate tensor c
with tf.Session() as sess:
  params = {a: 10, b: [1, 2]}    # a & b are tensor objects
  result = sess.run([c], feed_dict=params)
```

```
>>> print result
[array([ 10.,  20.], dtype=float32)]
```

Best practices note:

Structure your code with a `build_graph(...)` function which separates tensorflow graph operations from iterating through a dataset with calls to `sess.run(...)`

Demos

https://github.com/aymericdamien/TensorFlow-Examples

Tutorial index

0 - Prerequisite

- Introduction to Machine Learning.
- Introduction to MNIST Dataset.

1 - Introduction

- **Hello World** (notebook) (code). Very simple example to learn how to print "hello world" using TensorFlow.
- **Basic Operations** (notebook) (code). A simple example that cover TensorFlow basic operations.

2 - Basic Models

- **Linear Regression** (notebook) (code). Implement a Linear Regression with TensorFlow.
- **Logistic Regression** (notebook) (code). Implement a Logistic Regression with TensorFlow.
- **Nearest Neighbor** (notebook) (code). Implement Nearest Neighbor algorithm with TensorFlow.
- **K-Means** (notebook) (code). Build a K-Means classifier with TensorFlow.
- **Random Forest** (notebook) (code). Build a Random Forest classifier with TensorFlow.

3 - Neural Networks

Supervised

- **Simple Neural Network** (notebook) (code). Build a simple neural network (a.k.a Multi-layer Perceptron) to classify MNIST digits dataset. Raw TensorFlow implementation.
- **Simple Neural Network (tf.layers/estimator api)** (notebook) (code). Use TensorFlow 'layers' and 'estimator' API to build a simple neural network (a.k.a Multi-layer Perceptron) to classify MNIST digits dataset.
- **Convolutional Neural Network** (notebook) (code). Build a convolutional neural network to classify MNIST digits dataset. Raw TensorFlow implementation.
- **Convolutional Neural Network (tf.layers/estimator api)** (notebook) (code). Use TensorFlow 'layers' and 'estimator' API to build a convolutional neural network to classify MNIST digits dataset.
- **Recurrent Neural Network (LSTM)** (notebook) (code). Build a recurrent neural network (LSTM) to classify MNIST digits dataset.
- **Bi-directional Recurrent Neural Network (LSTM)** (notebook) (code). Build a bi-directional recurrent neural network (LSTM) to classify MNIST digits dataset.
- **Dynamic Recurrent Neural Network (LSTM)** (notebook) (code). Build a recurrent neural network (LSTM) that performs dynamic calculation to classify sequences of different length.

Unsupervised

- **Auto-Encoder** (notebook) (code). Build an auto-encoder to encode an image to a lower dimension and re-construct it.
- **Variational Auto-Encoder** (notebook) (code). Build a variational auto-encoder (VAE), to encode and generate images from noise.
- **GAN (Generative Adversarial Networks)** (notebook) (code). Build a Generative Adversarial Network (GAN) to generate images from noise.
- **DCGAN (Deep Convolutional Generative Adversarial Networks)** (notebook) (code). Build a Deep Convolutional Generative Adversarial Network (DCGAN) to generate images from noise.

4 - Utilities

- **Save and Restore a model** (notebook) (code). Save and Restore a model with TensorFlow.
- **Tensorboard - Graph and loss visualization** (notebook) (code). Use Tensorboard to visualize the computation Graph and plot the loss.
- **Tensorboard - Advanced visualization** (notebook) (code). Going deeper into Tensorboard; visualize the variables, gradients, and more...

5 - Data Management

- **Build an image dataset** (notebook) (code). Build your own images dataset with TensorFlow data queues, from image folders or a dataset file.
- **TensorFlow Dataset API** (notebook) (code). Introducing TensorFlow Dataset API for optimizing the input data pipeline.

6 - Multi GPU

- **Basic Operations on multi-GPU** (notebook) (code). A simple example to introduce multi-GPU in TensorFlow.
- **Train a Neural Network on multi-GPU** (notebook) (code). A clear and simple TensorFlow implementation to train a convolutional neural network on multiple GPUs.

Installing Tensorflow

Start with **Anaconda**, both python 2 and 3 are supported

All major python libraries are included: `Numpy, Matplotlib, Jupyter Notebook, etc.`

Then:

```
pip install tensorflow
```

or

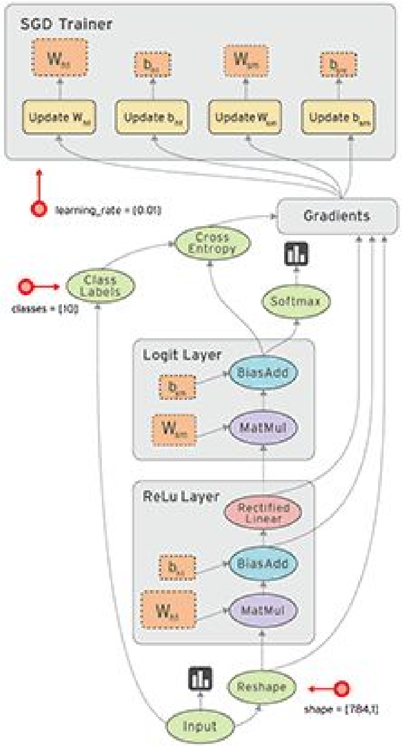
```
pip install tensorflow-gpu
```

University GPU resources:

- **`citrisdance.soe.ucsc.edu`**
  - 2 older K20 GPUs and 32 cores storage is an issue
- **`https://patternlab.calit2.optiputer.net/`**
  - 2 fast M40 GPUs, 40 cores, shared with UCSD, 60+TB of storage
- More coming thanks to a recent NSF grant!

Automatic Differentiation

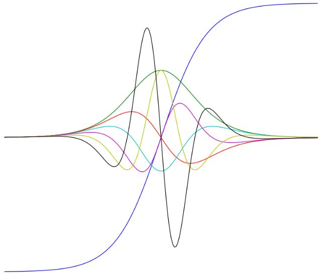
a.k.a.: How tensorflow and other tools do their magic




```
>>> import autograd.numpy as np  # Thinly-wrapped numpy
>>> from autograd import grad    # The only autograd function you may ever need
>>>
>>> def tanh(x):                 # Define a function
...     y = np.exp(-2.0 * x)
...     return (1.0 - y) / (1.0 + y)
...
>>> grad_tanh = grad(tanh)       # Obtain its gradient function
>>> grad_tanh(1.0)               # Evaluate the gradient at x = 1.0
0.41997434161402603
>>> (tanh(1.0001) - tanh(0.9999)) / 0.0002  # Compare to finite differences
0.41997434264973155
```

We can continue to differentiate as many times as we like, and use numpy's broadcasting of scalar-valued functions across many different input values:

```
>>> import matplotlib.pyplot as plt
>>> x = np.linspace(-7, 7, 200)          # grad broadcasts across inputs
>>> plt.plot(x, tanh(x),
...          x, grad(tanh)(x),                                # first  derivative
...          x, grad(grad(tanh))(x),                          # second derivative
...          x, grad(grad(grad(tanh)))(x),                    # third  derivative
...          x, grad(grad(grad(grad(tanh))))(x),              # fourth derivative
...          x, grad(grad(grad(grad(grad(tanh)))))(x),        # fifth  derivative
...          x, grad(grad(grad(grad(grad(grad(tanh))))))(x))  # sixth  derivative
>>> plt.show()
```



https://github.com/HIPS/autograd

https://github.com/HIPS/autograd
https://www.tensorflow.org/images/tensors_flowing.gif