Today: Outline

- Homework Submission
- **ConvNets**: Example
- Training Strategies for Neural Networks
- Feature Extraction and Transfer Learning

- Reminders: *Pre-lec Material 2,*
  due: Friday, Jun 4

  Problem Set 1,
  due: Friday, Jun 4

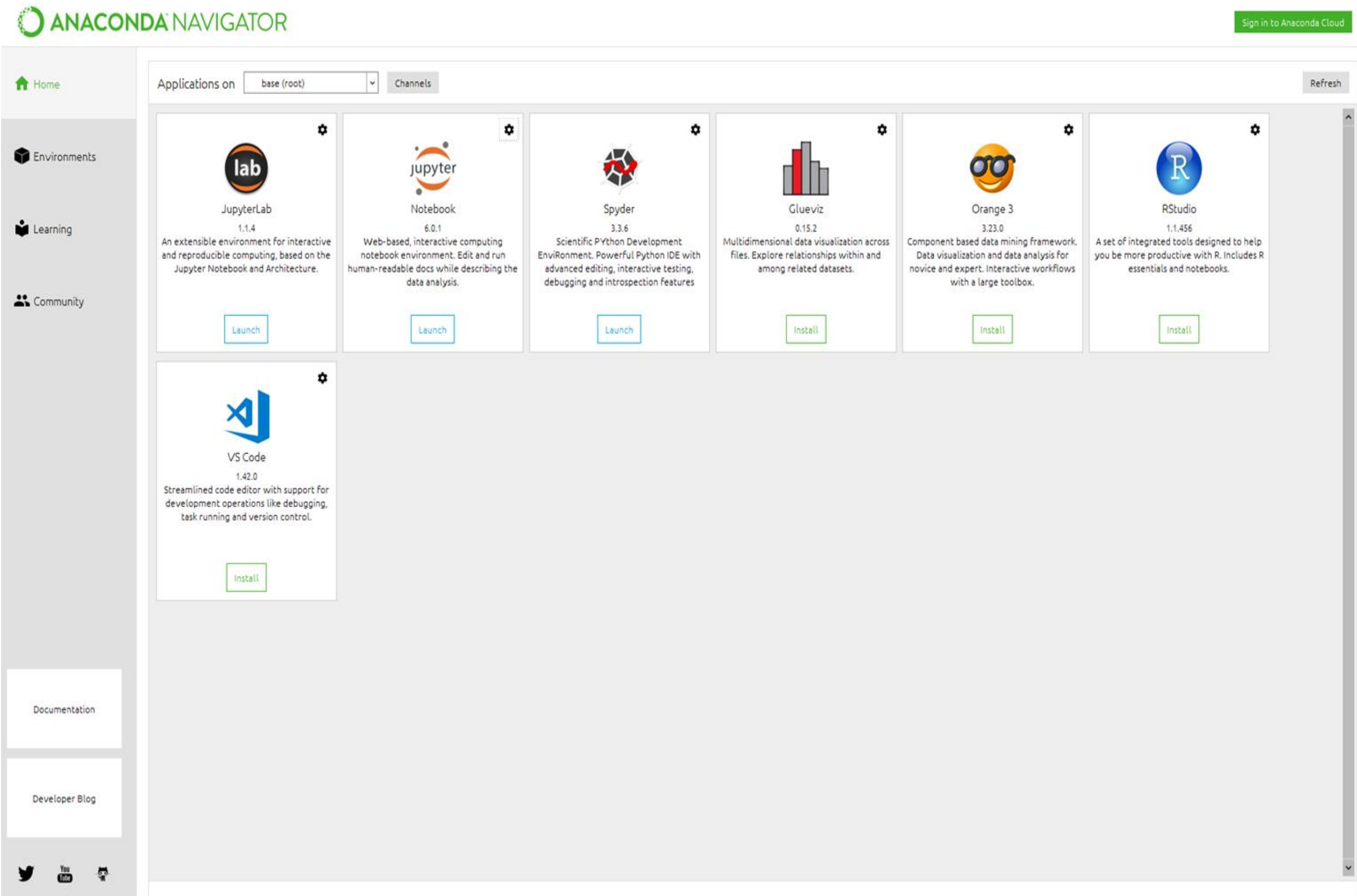
Anaconda Installation

- To run and solve assignments in this course, one must have a working IPython Notebook installation.

- The easiest way to set it up for both Windows and Linux is to:
  - install Anaconda: https://www.anaconda.com/distribution/ (Python Version 3)
  - save and run this file to your computer

- If you are new to Python or its scientific library, Numpy, there are some nice Tutorials: https://www.learnpython.org and http://scipy-lectures.org

- In Windows after installation, search “Anaconda Navigator”. In the GUI menu, you can launch Jupyer Notebook or Jupyter Lab. These IDEs are based on a web-browser, you can enter localhost:8888 in a web browser and enter the work directory.



ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
base (root)
Channels
Refresh
JupyterLab
1.1.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
6.0.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
Orange 3
3.23.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install
VS Code
1.42.0
Streamlined code editor with support for development operations like debugging, task running and version control.
Install

Jupyter Notebook

1. Numpy Tutorial

1.1 [5pt] Modify the cell below to return a 5x5 matrix of ones. Put some code there and press `Ctrl+Enter` to execute contents of the cell. You should see something like the output below. [1] (https://docs.scipy.org/doc/numpy-1.13.0/user/basics.creation.html#arrays-creation) [2] (https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.array-creation.html#routines-array-creation)

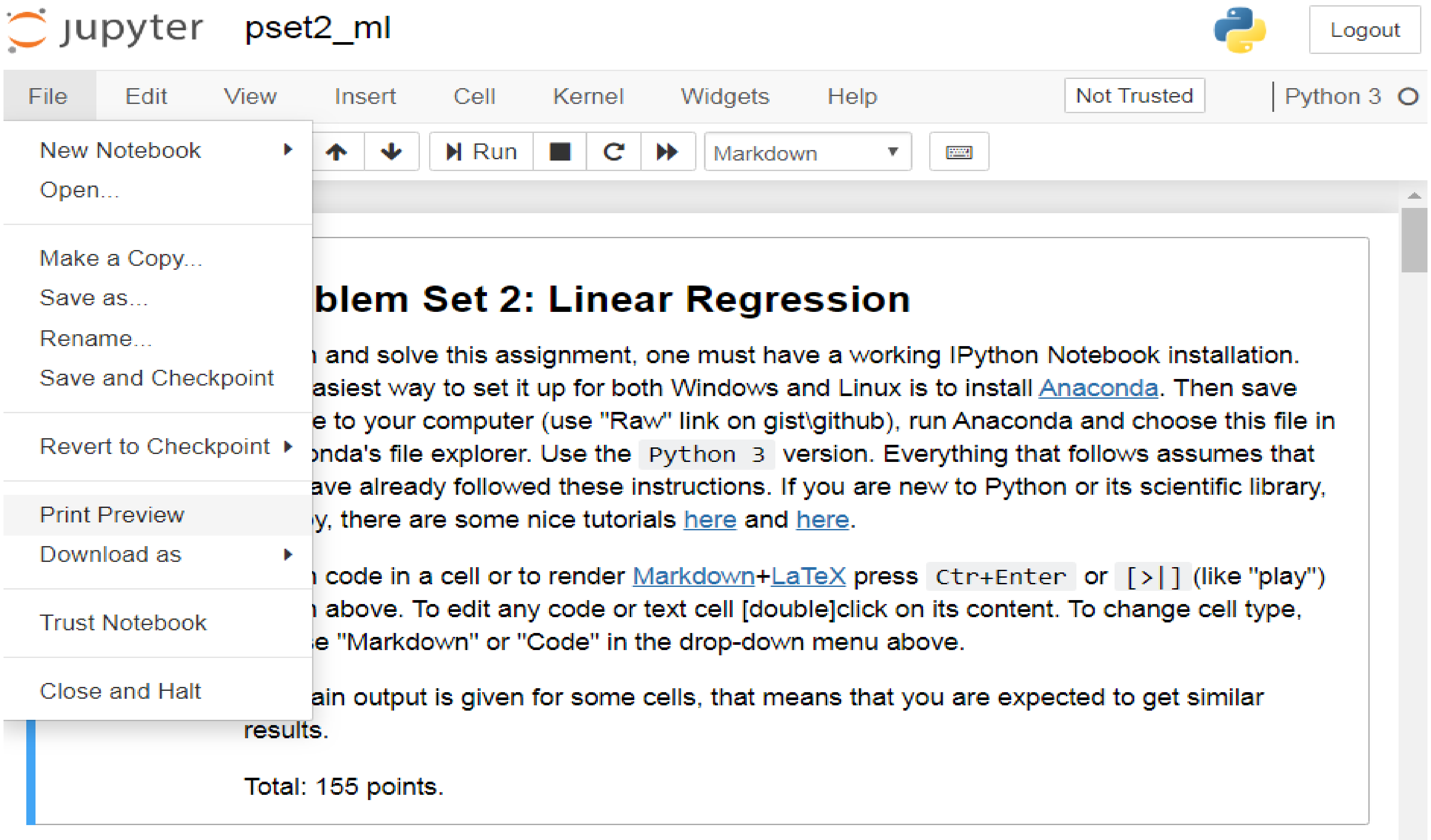
In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

raise NotImplementedError("Replace this raise statement with the code "
                          "that prints 5x5 matrix of ones")
```

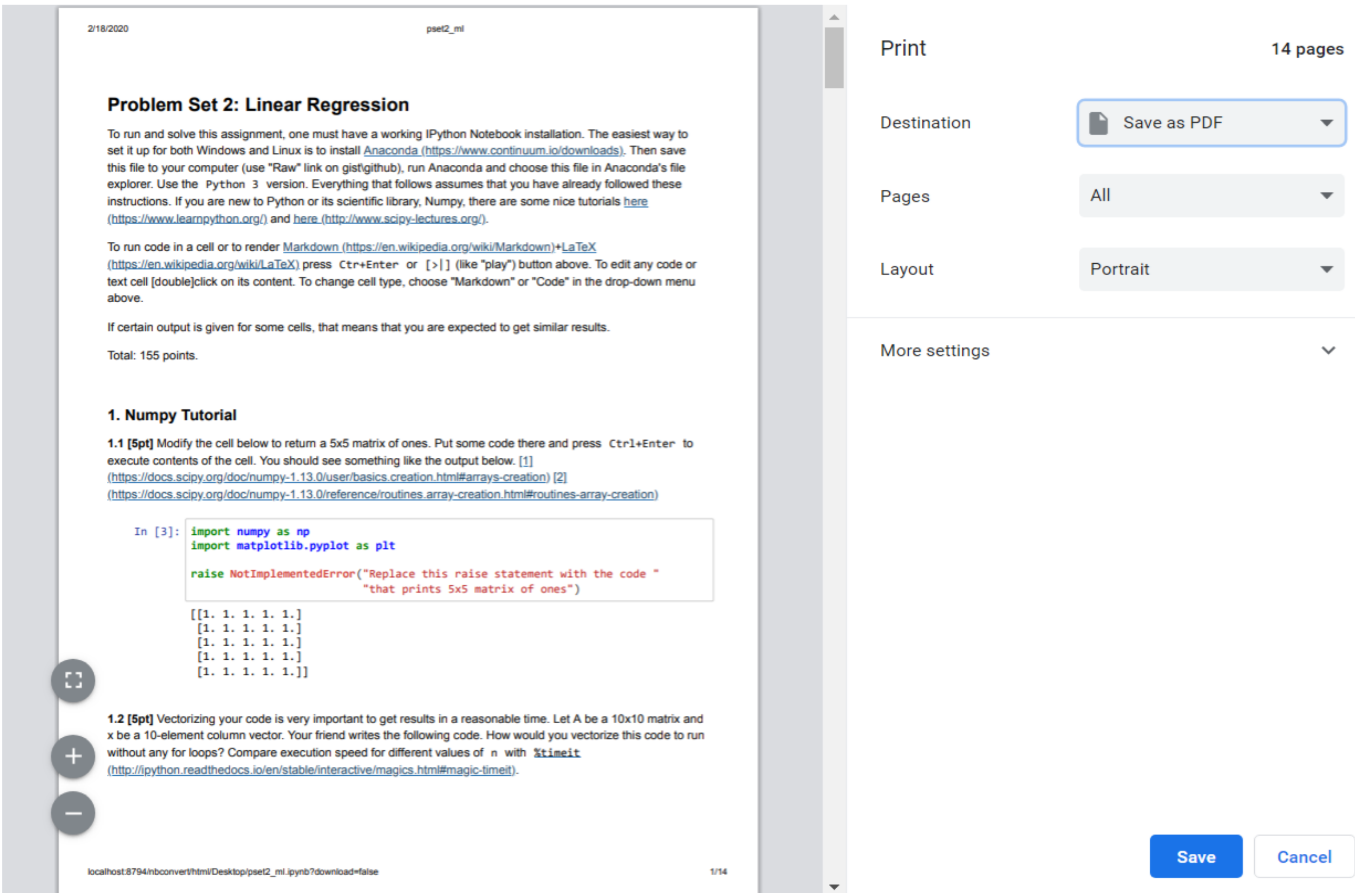
```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

1.2 [5pt] Vectorizing your code is very important to get results in a reasonable time. Let A be a 10x10 matrix and x be a 10-element column vector. Your friend writes the following code. How would you vectorize this code to run without any for loops? Compare execution speed for different values of `n` with `%timeit` (http://ipython.readthedocs.io/en/stable/interactive/magics.html#magic-timeit).

Jupyter notebook - Save as pdf (i)



Jupyter notebook - Save as pdf (ii)



Submission

CS 523 | Summer 2021 — Entry Code: **BP8RKK**

DESCRIPTION

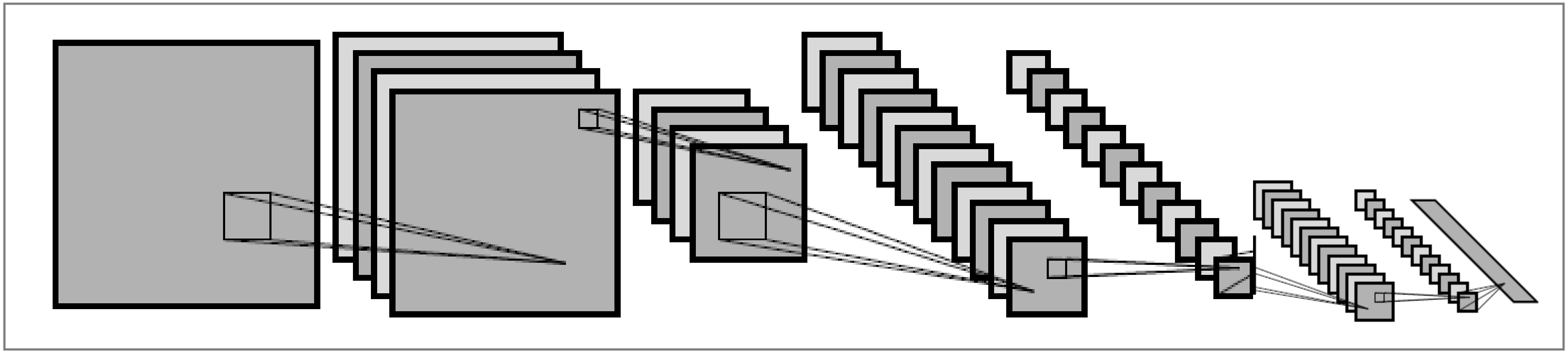
Deep Learning

ACTIVE ASSIGNMENTS	RELEASED	DUE (EDT)	SUBMISSIONS
Problem Set 1	MAY 28	JUN 04 AT 11:59PM	1

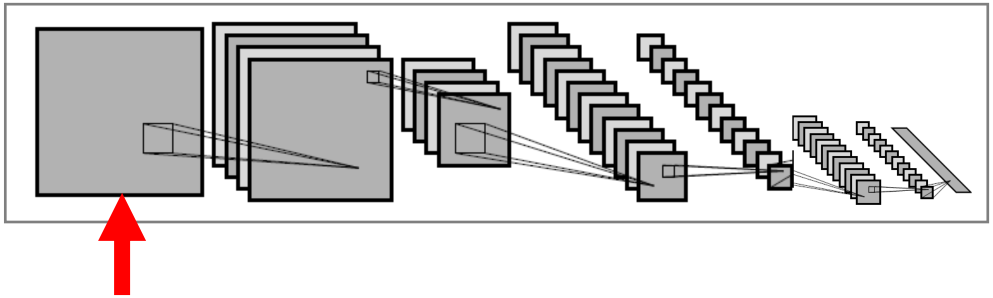
Convolutional Neural Nets

Example

CIFAR-10 Demo ConvJS Network

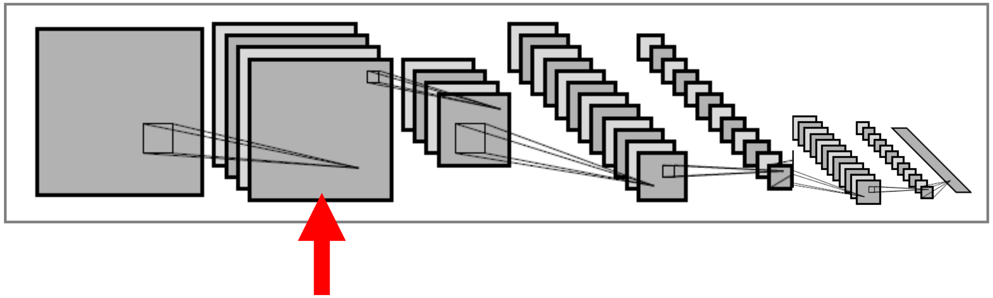


input image conv RELU pool conv RELU pool conv RELU pool fc



filter size 5x5x3, stride 1

input (32x32x3)

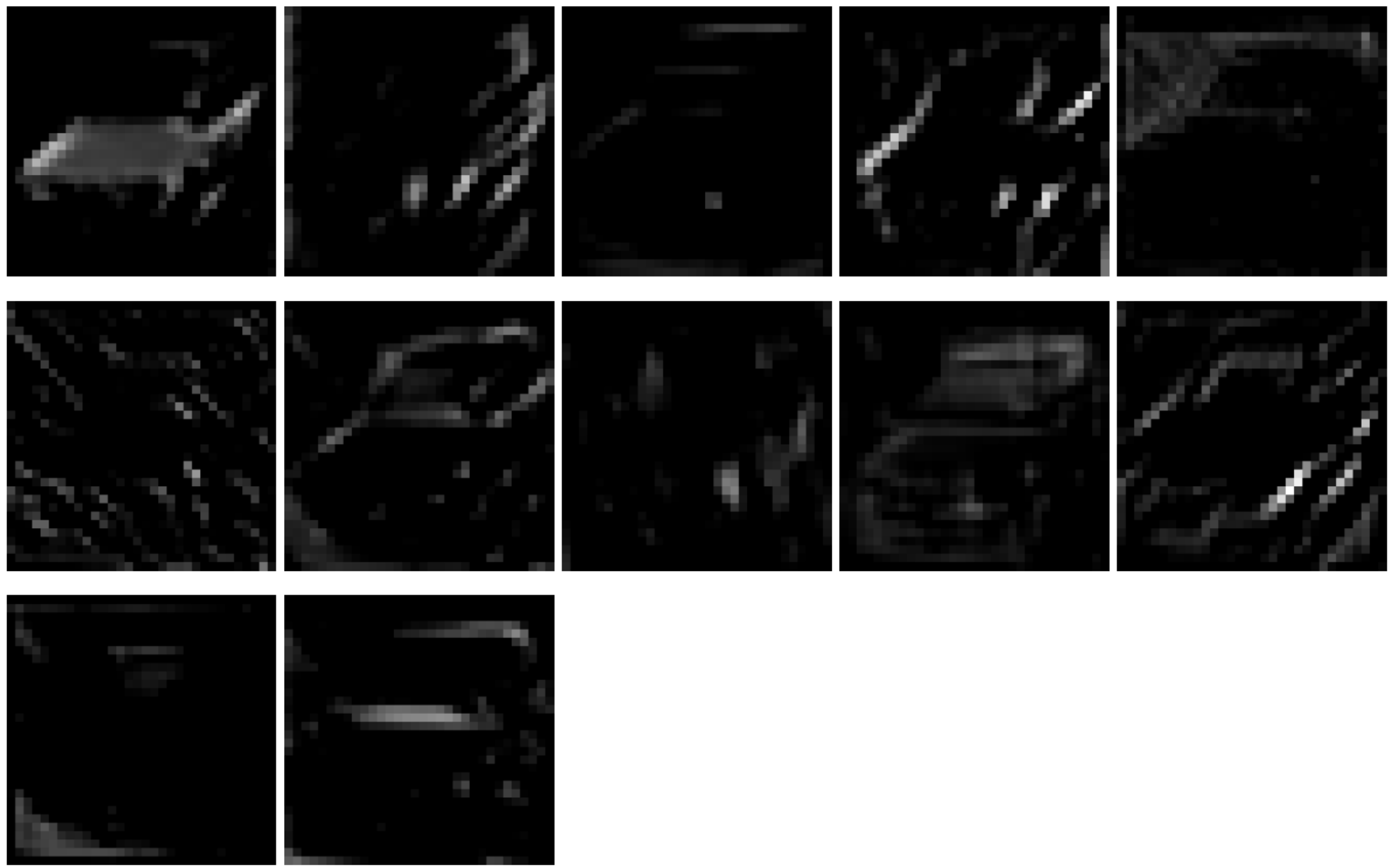


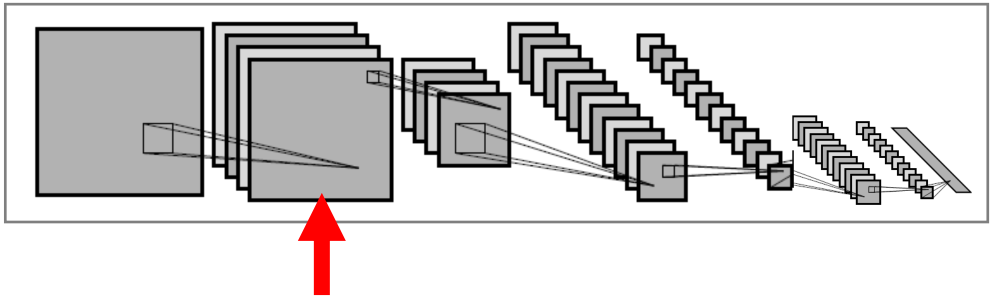
filter size 5x5x3, stride 1

input (32x32x3)

RELU

conv (32x32x16) params: 16x5x5x3+16 = 1216

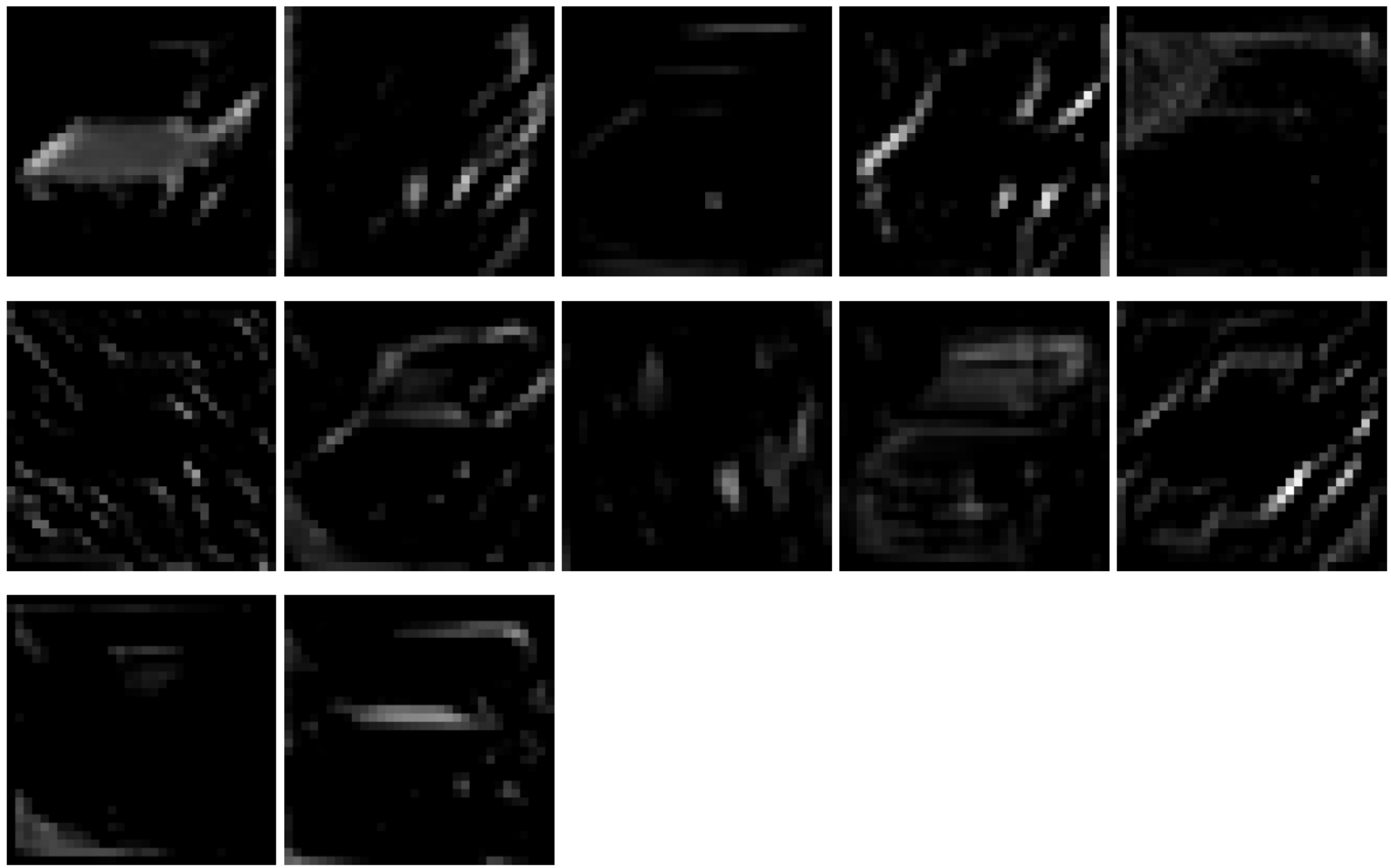


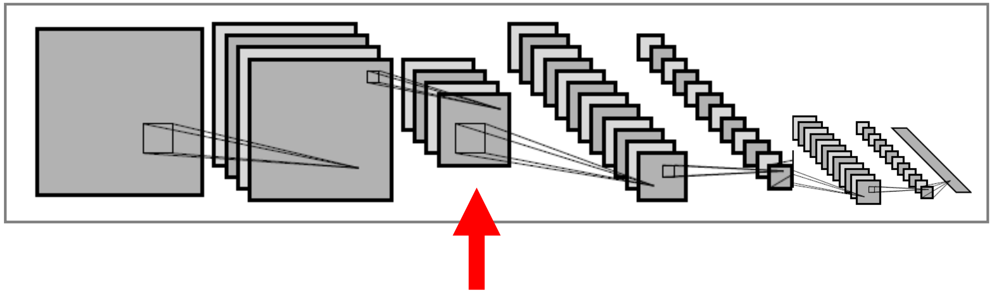


filter size 5x5x3, stride 1

input (32x32x3)

conv (32x32x16) params: 16x5x5x3+16 = 1216

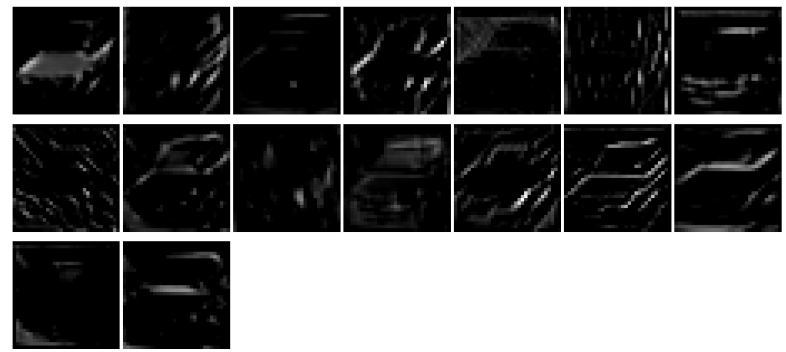




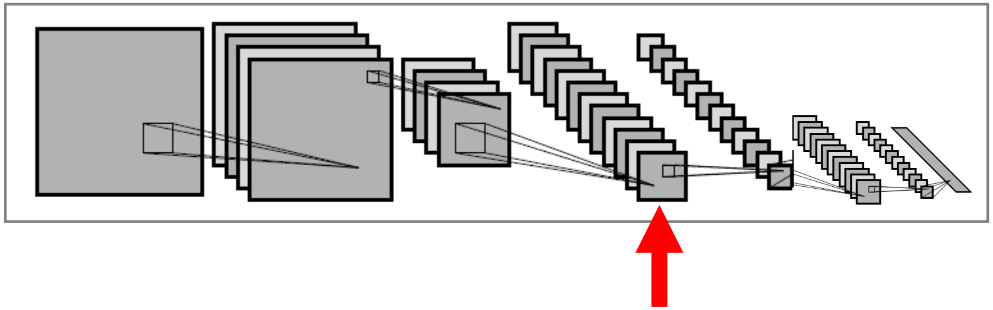
filter size 5x5x3, stride 1

input (32x32x3)

conv (32x32x16) params: 16x5x5x3+16 = 1216



pool (16x16x16)
pooling size 2x2, stride 2

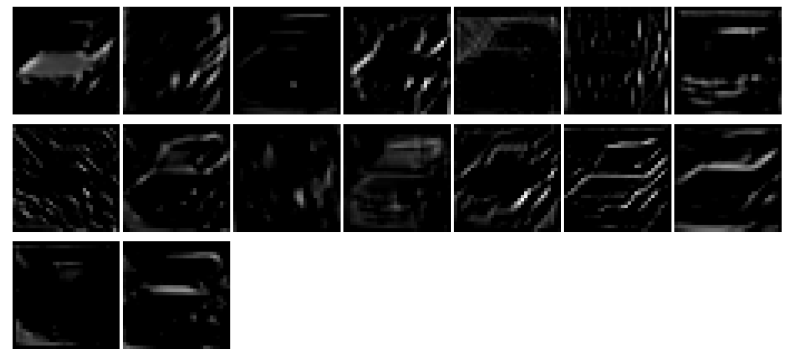


filter size 5x5x3, stride 1

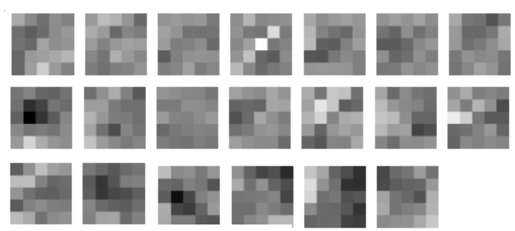
input (32x32x3)

pool (16x16x16)
pooling size 2x2, stride 2

conv (32x32x16) params: 16x5x5x3+16 = 1216

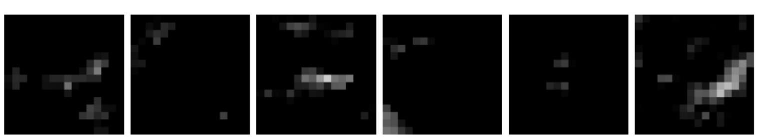


filter size 5x5x16, stride 1

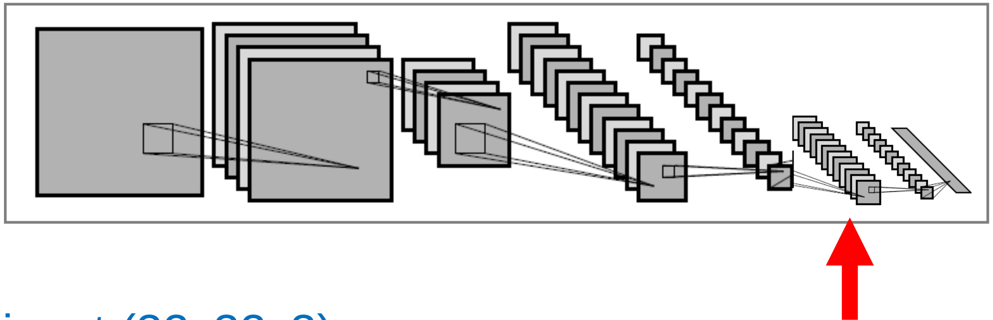


RELU

conv (16x16x20) params: 20x5x5x16+20 = 8020



pool (8x8x20)
pooling size 2x2, stride 2



One more conv+RELU+pool:

conv (8x8x20)
filter size 5x5x20, stride 1
relu (8x8x20)
pool (4x4x20)
pooling size 2x2, stride 2

input (32x32x3)

fc (1x1x10); parameters: 10x320+10 = 3210

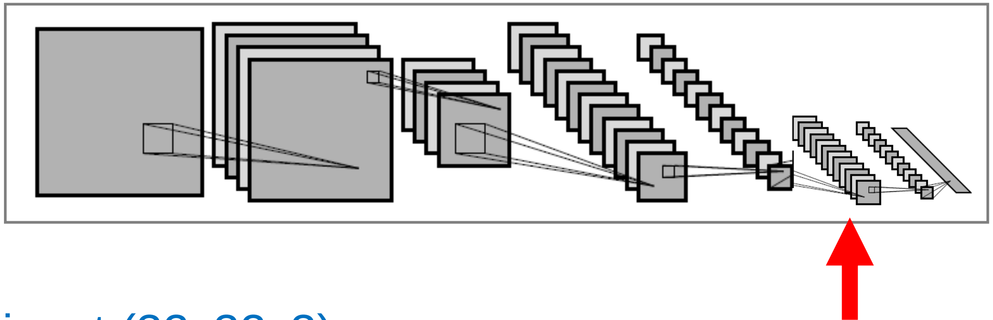
softmax (1x1x10)

Dog car Cat …

Softmax

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

$\vec{z}$	The input vector to the softmax function, made up of (z0, ... zK)
z_i	All the zi values are the elements of the input vector to the softmax function, and they can take any real value, positive, zero or negative. For example a neural network could have output a vector such as (-0.62, 8.12, 2.53), which is not a valid probability distribution, hence why the softmax would be necessary.
e^{z_i}	The standard exponential function is applied to each element of the input vector. This gives a positive value above 0, which will be very small if the input was negative, and very large if the input was large. However, it is still not fixed in the range (0, 1) which is what is required of a probability.
$\sum_{j=1}^{K} e^{z_j}$	The term on the bottom of the formula is the normalization term. It ensures that all the output values of the function will sum to 1 and each be in the range (0, 1), thus constituting a valid probability distribution.
K	The number of classes in the multi-class classifier.



One more conv+RELU+pool:

conv (8x8x20)
filter size 5x5x20, stride 1
relu (8x8x20)
pool (4x4x20)
pooling size 2x2, stride 2

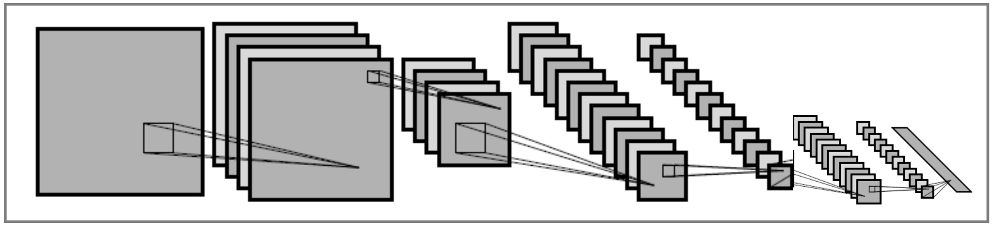
input (32x32x3)

fc (1x1x10); parameters: 10x320+10 = 3210

softmax (1x1x10)

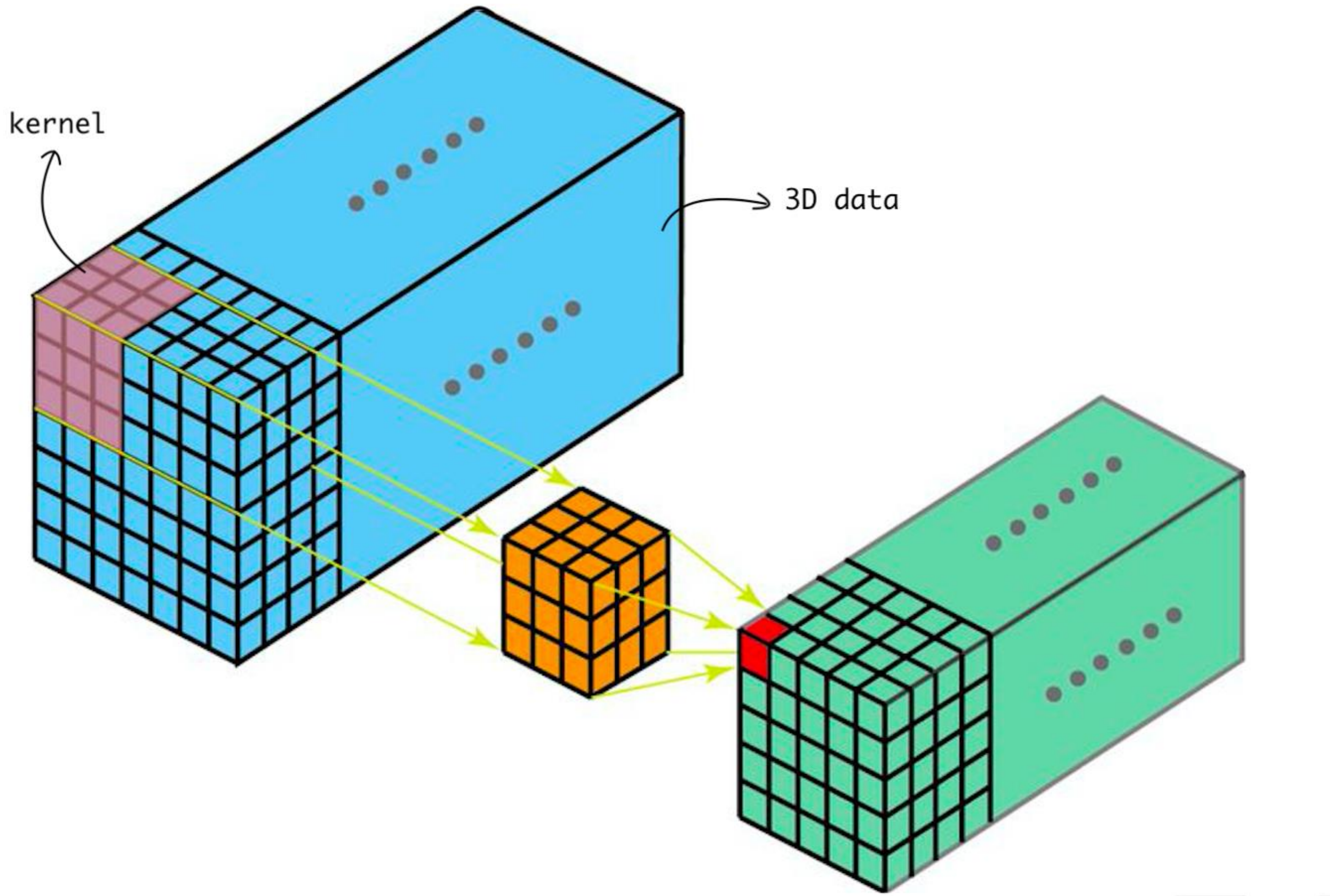
Testing the network

- Show top three most likely classes



http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

3D Convolutional Neural Networks



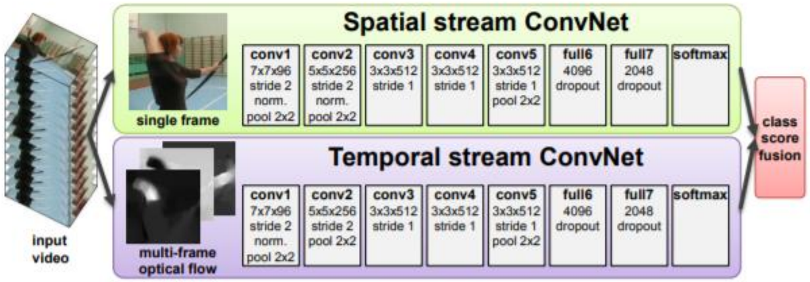
https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610

Application:
AI Generated Match Highlights

- IBM's produce the official match highlights of Wimbledon and US Open tennis tournaments.

- https://www.usopen.org/en_US/video/2017-08-31/1504233424.html

- Multi-modal System



- Bias Considerations

Neural Networks

Training Strategies

Universality

- Why study neural networks in general?
  - Neural network can approximate any continuous function!
  - http://neuralnetworksanddeeplearning.com/chap4.html

Why Study Deep Networks?

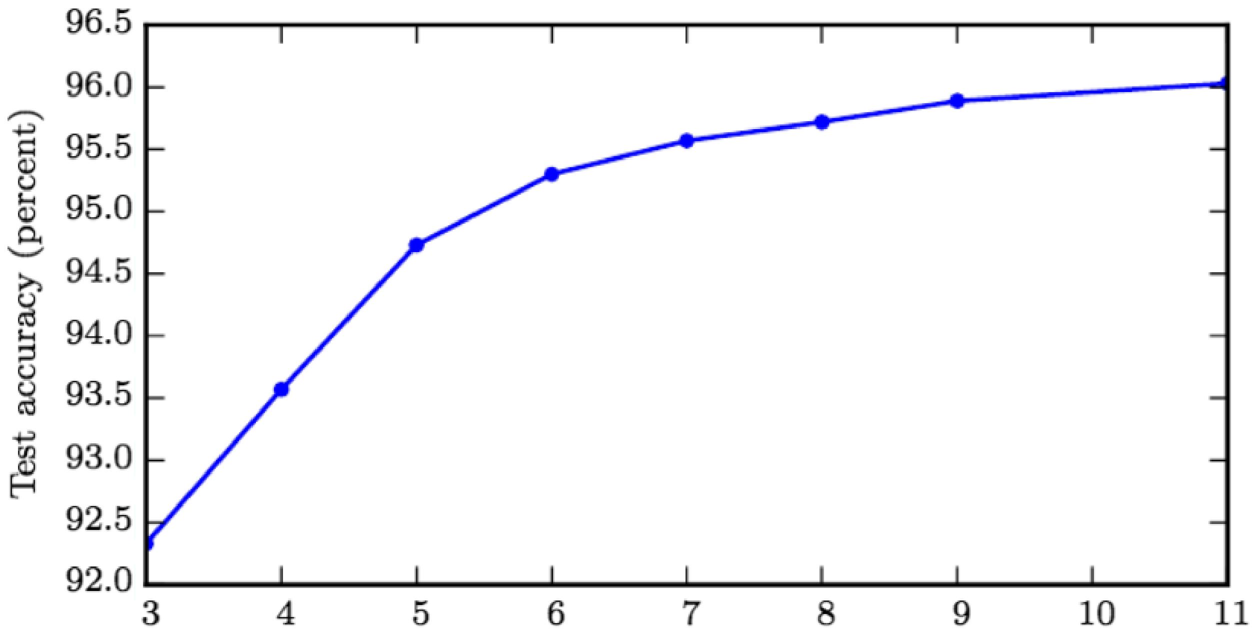
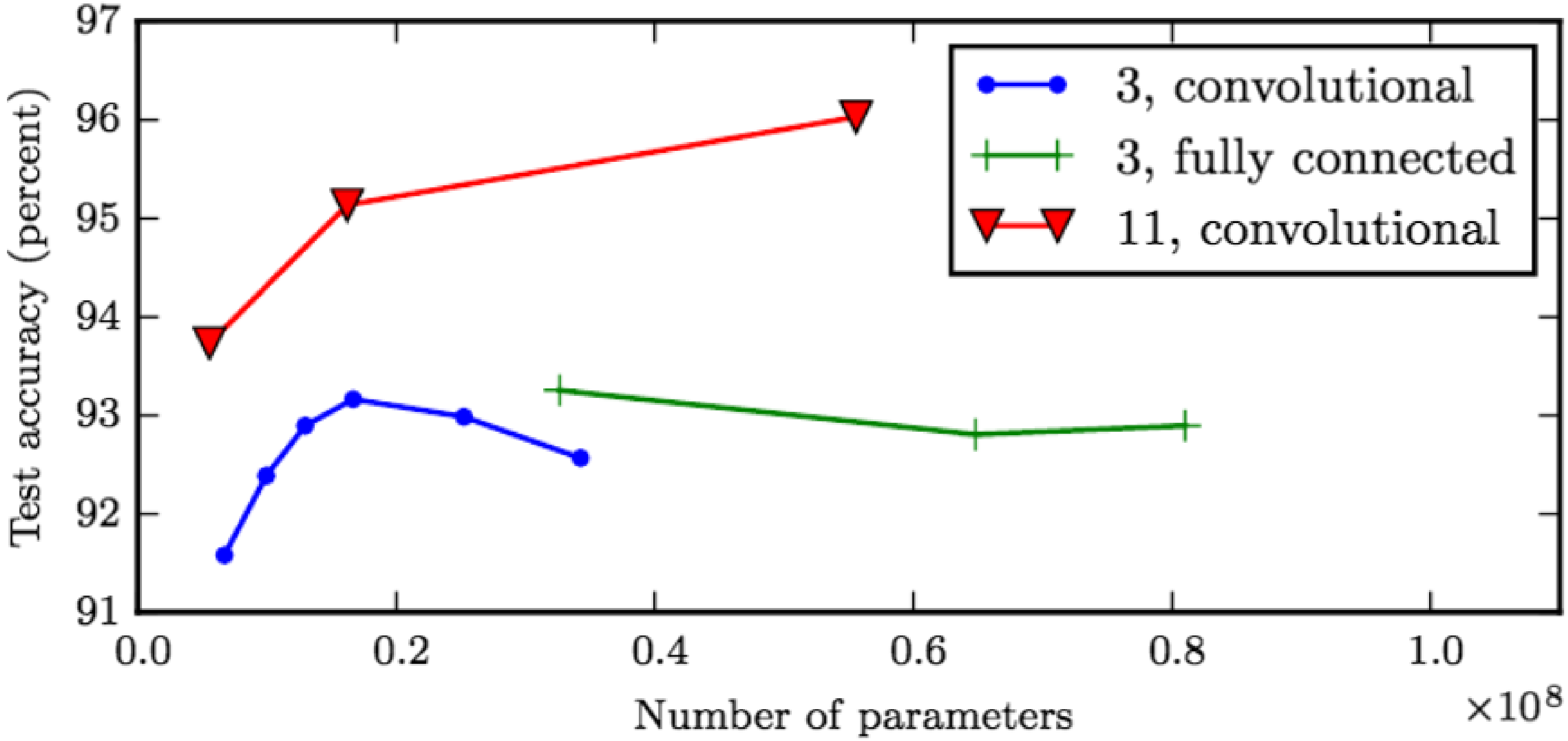


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow *et al.* (2014d). The test set accuracy consistently increases with increasing depth. See figure 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

Efficiency of convnets



Activation Functions

- ReLU: $g(x) = \max(0, x)$
- Leaky ReLU: $g(x) = \max(0, x) + \alpha \min(0, x) \quad (\alpha \approx .01)$
- Tanh: $g(x) = 2\sigma(2x) - 1$
- Radial Basis Functions: $g(x) = \exp(-(w - x)^2 / \sigma^2)$
- Softplus: $g(x) = \log(1 + e^x)$
- Hard Tanh: $g(x) = \max(-1, \min(1, x))$
- Maxout: $g(x) = \max_{j \in \mathbb{G}} x_j$
- ....

Architectures

- Some commonly referred to architectures:
  - AlexNet
  - VGG16/19
  - GoogleNet
  - ResNet
  - WideResNet
  - Inception
  - …

Architecture Design and Training Issues

- How many layers? How many hidden units per layer? How to connect layers together? How to optimize?
  - Cost functions
  - L2/L1 regularization
  - Data Set Augmentation
  - Early Stopping
  - Dropout
  - Minibatch Training
  - Momentum
  - Initialization
  - Batch Normalization

Cost Functions

- For regression problems, quadratic error is typical
- For classification, one typically uses softmax outputs with cross-entropy error function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

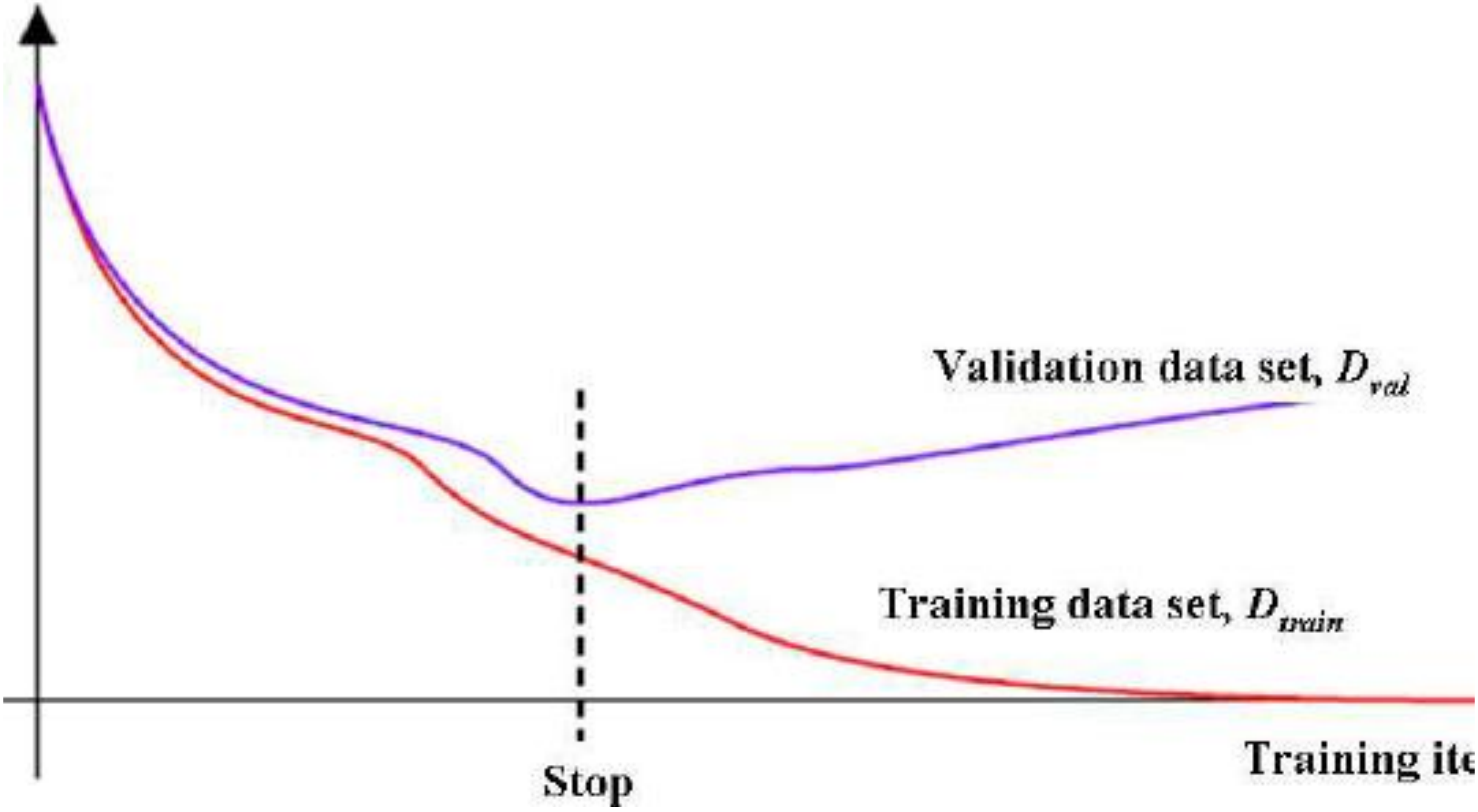
Regularization

- In machine learning, we care about *generalization performance*, not just training error
- With many parameters, models are prone to *overfitting*
- How to regularize?
  - Restrictions on parameter values
  - Adding terms to the objective function
  - Examples: L2 or L1 regularization
- Other ways to improve generalization?

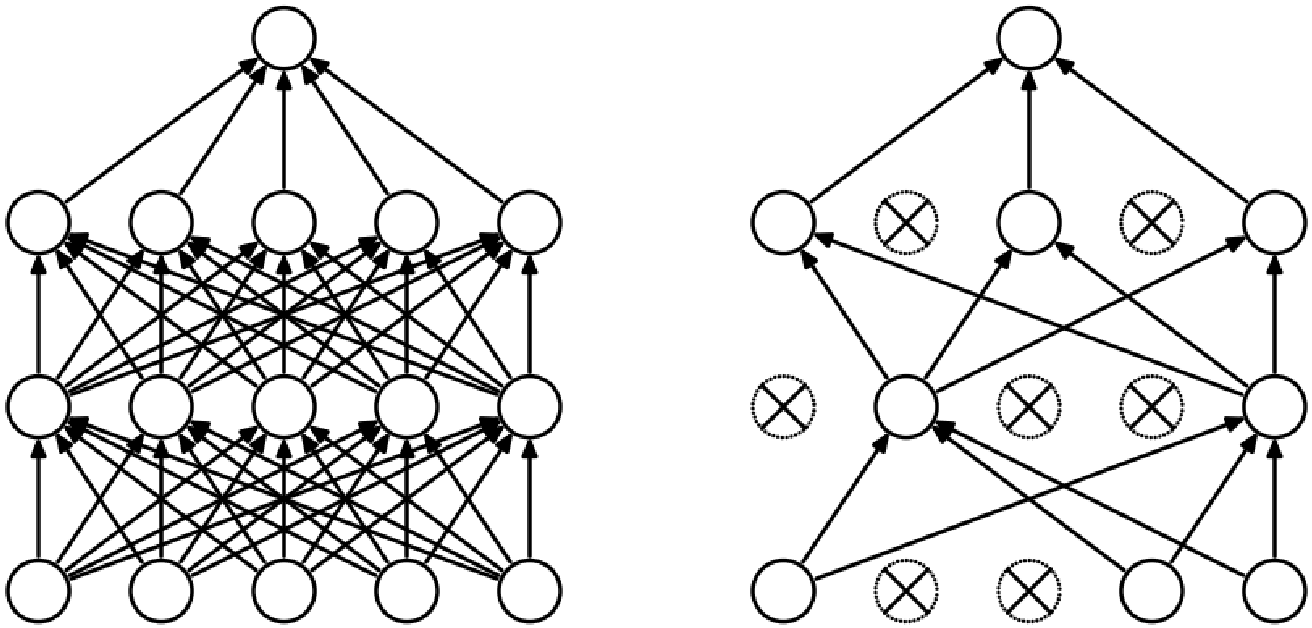
Data Set Augmentation

- The more data, the better for generalization (usually)
- Sometimes we can augment our existing data set
  - Example: for image classification, mirror-image all images to double the size of the training set
  - Injecting noise to training data is also a form of data augmentation

Early Stopping



Dropout



(a) Standard Neural Net

(b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Dropout

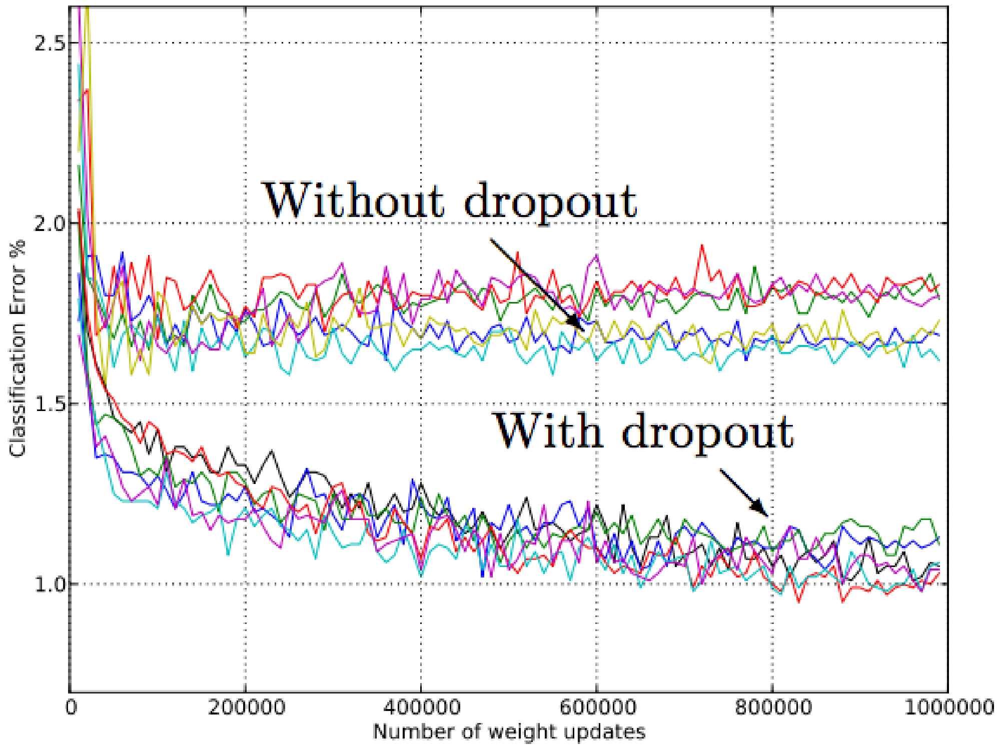


Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Architecture Design and Training Issues

- How many layers? How many hidden units per layer? How to connect layers together? How to optimize?
  - Cost functions
  - *L2/L1 regularization*
  - *Data Set Augmentation*
  - *Early Stopping*
  - *Dropout*
  - Minibatch Training
  - Momentum
  - Initialization
  - Batch Normalization

Avoid Overfitting

Minibatch Training

- Gradient descent uses all training points, fully online (stochastic) methods update using a single point
- Many deep learning methods fall in between
- Example: computing the mean of a set of samples
    - Standard error based on a sample of n points is $\sigma / \sqrt{n}$
    - Consider using 100 versus 10,000 samples
    - Latter requires 100x more computation but reduces error by factor of 10

Minibatch Training

- Larger batches provide a more accurate estimate of the gradient. If all examples in the batch are processed in parallel, amount of memory scales with batch size.

- Small batches can offer a regularizing effect due to the noise added during the learning process.

- When using GPUs it is common for power of 2 batch sizes to offer better runtime; typical sizes range from 32 to 256, with 16 being common for large models.

- Minibatches should be selected randomly!

GPUs

- NVIDIA TITAN V GPU

Mini-batches

- Gradients could be updated using:
  - One data point (too inaccurate)
  - All data points (too expensive)
  - Mini-batch (a good trade-off)

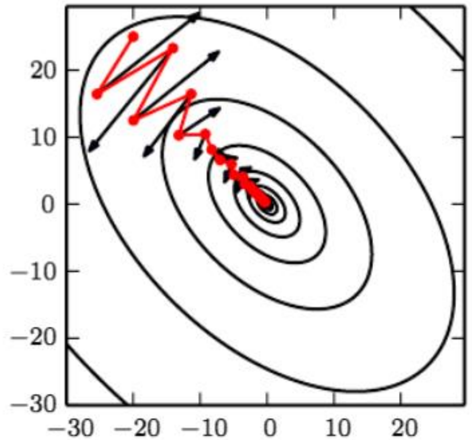
- The size of the mini-batch depends on:
  - How good of an approximation you need
  - How much GPU memory you have per GPU
  - How many GPUs you have

- GPUs can compute gradients of mini-batches in parallel, *i.e. Training on multiple GPUs*: Divide and Conquer.

Momentum

- Accumulates an exponentially decaying moving average of past gradients and continues to move in their direction

- exponentially weighed averages can provide us a better estimate which is closer to the actual derivate than our noisy batch calculations



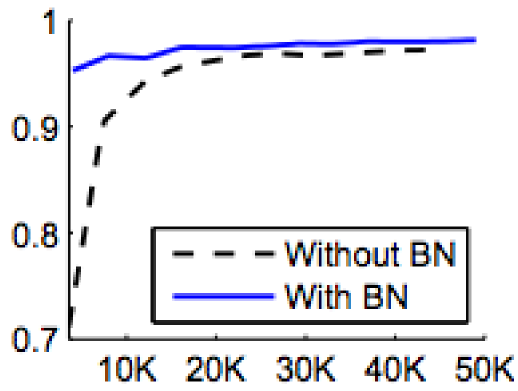
Initialization

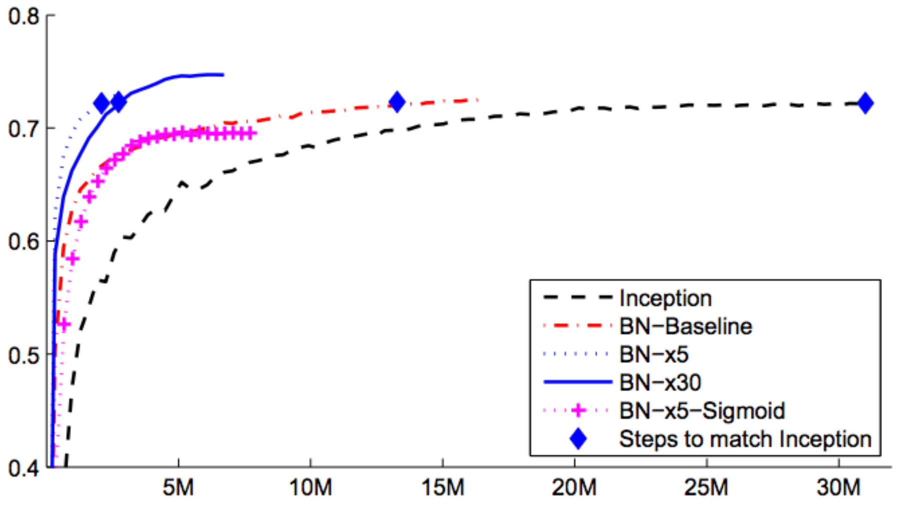
- Important: need to "break symmetry"
  - E.g. Choose weights randomly
- Combined with early stopping, can think of initialization as a prior on the weights
- Usually use uniform or Gaussian weights with a zero-mean
- Examples with m inputs and n outputs:

$$W_{ij} \sim U\left(-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}\right) \qquad W_{ij} \sim U\left(-\frac{6}{\sqrt{m+n}}, \frac{6}{\sqrt{m+n}}\right)$$

Batch Normalization

- High-level idea: whitening the data at each layer makes training faster
- Left: MNIST, Right: ImageNet





Data Independence

- NN models converging to the correct solution depends on the iid assumption
- i.e. that our data are independent and identically distributed
- Important to randomly shuffle examples! Otherwise net can fail to converge