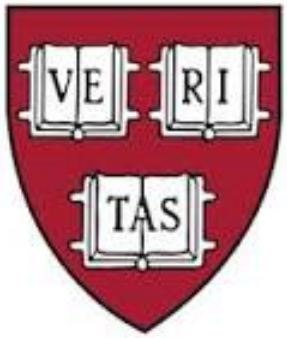


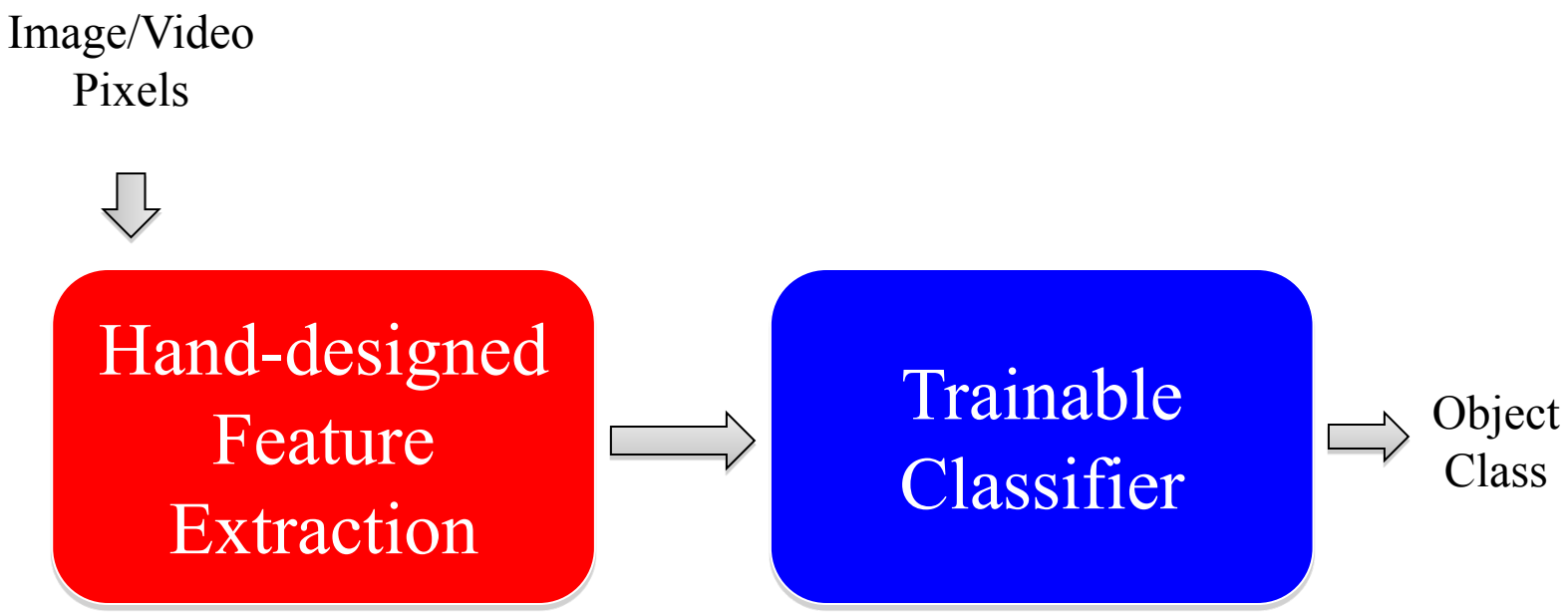
Week 12 – Rahul Joglekar

CSCI-E63, 11-19-2017

Section Objectives

1	CNN Concepts
2	CNN with Tensor Flow

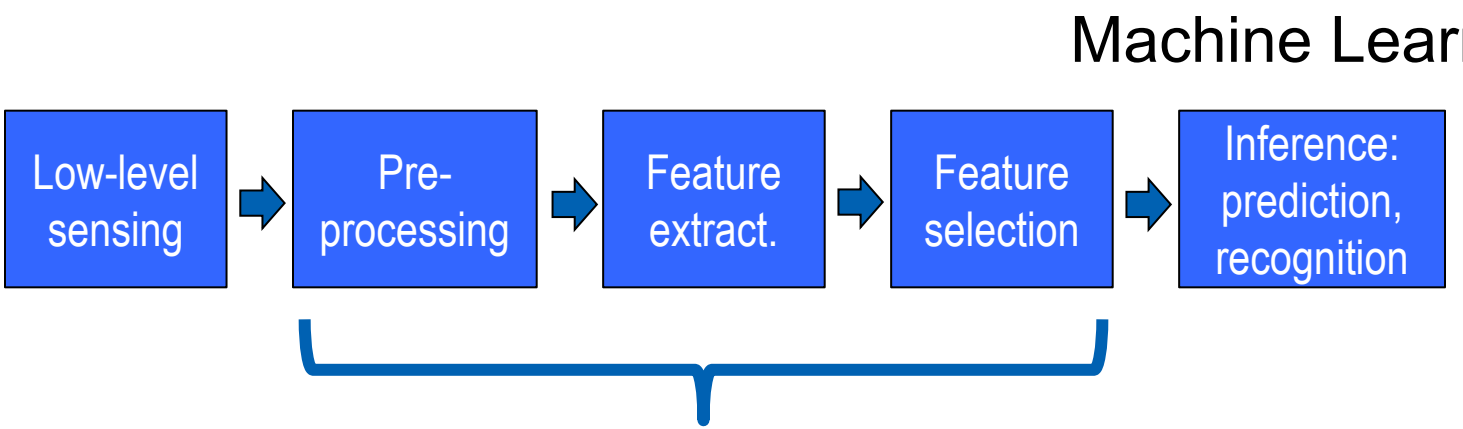
Old Approach to Image Classification



- Features are not learned
- Trainable classifier is often generic (e.g. SVM)

Slide Credit- Y. Lecun

Learning features from data



Conv Nets - Common Terms

Channel conventional term used to refer to a certain component of an image. An image from a standard digital camera will have three channels – red, green and blue – you can imagine those as three 2d-matrices stacked over each other (one for each color), each having pixel values in the range 0 to 255.

Convolution

The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

A convolution is a mathematical operation that is performed on two functions.

For continuous functions, it is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

For discrete functions, it is:

$$(f * g)(t) = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Mid-Level Representations or Filters

Continuation Parallelism Junctions Corners

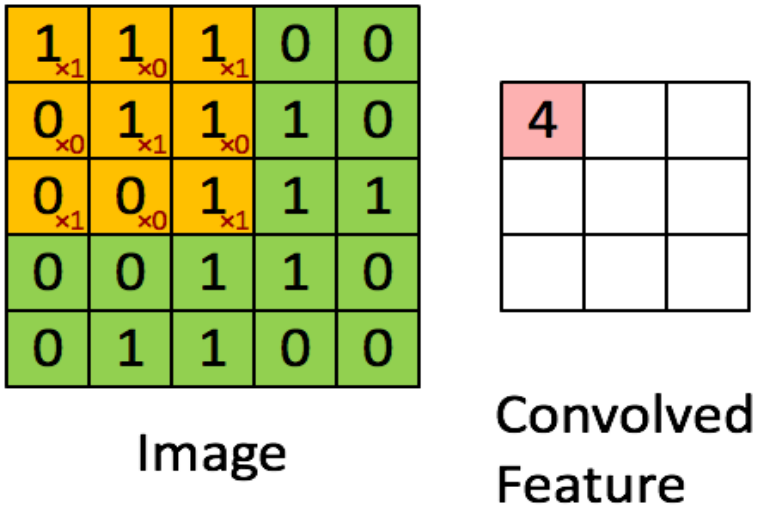
Object parts:

Difficult to hand-engineer → What about learning them?

CNN – Common Terms - Filters and Feature maps

Filter - The 3×3 matrix is called a '**filter**' or 'kernel' or 'feature detector' and the matrix formed by sliding the filter over the image and computing the dot product is called the 'Convolved Feature' or 'Activation Map' or the '**Feature Map**'. It is important to note that filters acts as feature detectors from the original input image.

CNN *learns* the values of these filters on its own during the training process - we still need to specify parameters such as number of filters, filter size, architecture of the network before the training process.



Conv Nets - Common Terms

Depth: Depth corresponds to the number of filters we use for the convolution operation.You can think of these three feature maps as stacked 2d matrices, so, the 'depth' of the feature map would be three.

Stride: Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.

Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix.

Non Linearity - Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU. ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero.

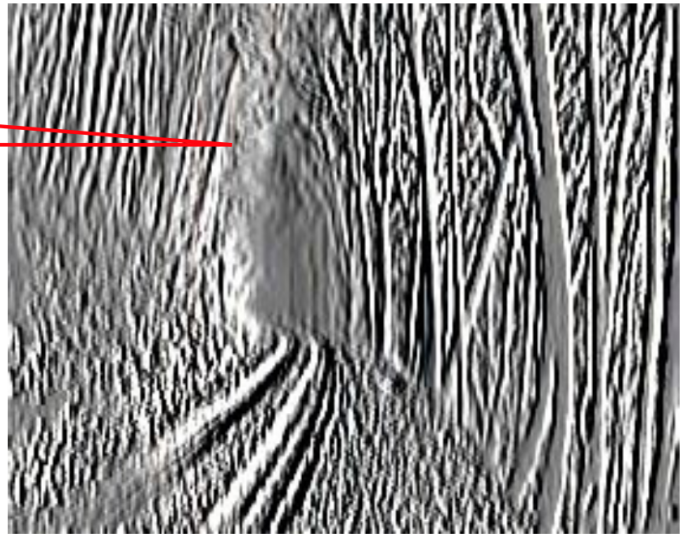
Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

Simple example of Applying a Filter

Filter

Mathematically Filter

$$* \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} =$$



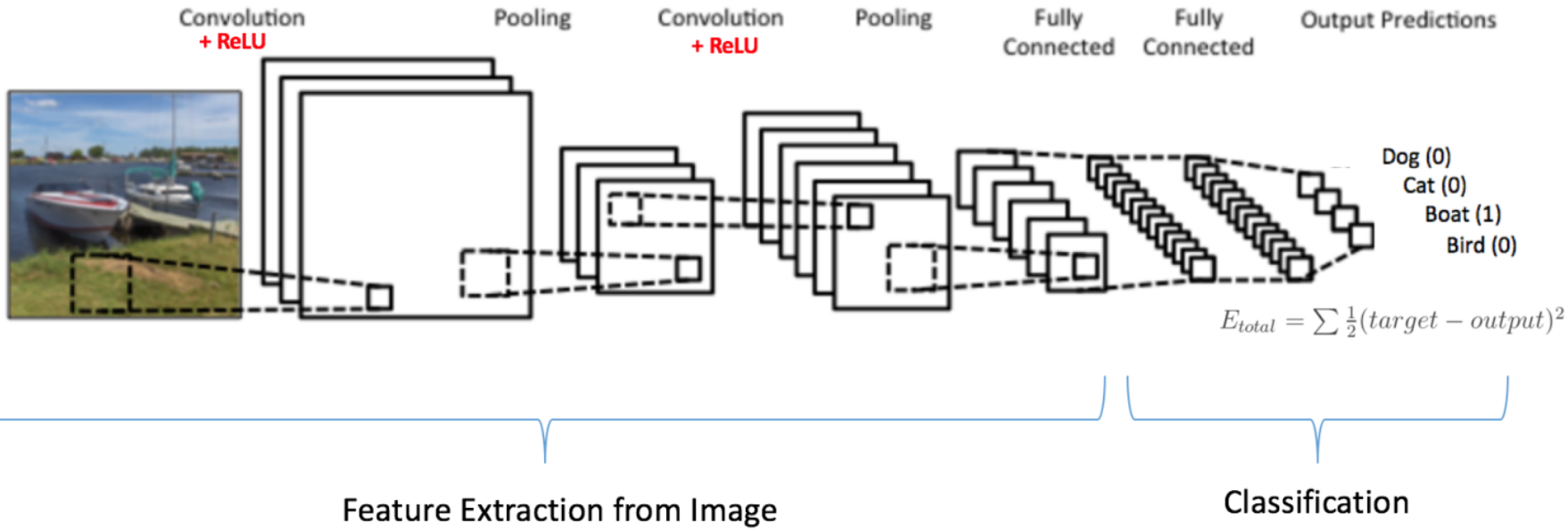
Input : Grey Scale Picture

Output : Feature map

Filtering in action

ConvNet Architecture

- Input Image = Boat
- Target Vector = [0, 0, 1, 0]



Assignment 12 Discussion

Problem 1

Info provided to us

- 2 files are provided to us cnn_mnist.py and mnist2-1.py (renamed as mnist2.py)
- Accuracy of mnist2.py is around 70%
- Accuracy of cnn_mnist.py is around 95%

```
Generation # 465. Train Loss: 0.04. Train Acc (Test Acc): 99.00 (95.00)
Generation # 470. Train Loss: 0.07. Train Acc (Test Acc): 97.00 (95.80)
Generation # 475. Train Loss: 0.08. Train Acc (Test Acc): 96.00 (96.40)
Generation # 480. Train Loss: 0.04. Train Acc (Test Acc): 99.00 (96.20)
Generation # 485. Train Loss: 0.13. Train Acc (Test Acc): 96.00 (96.80)
Generation # 490. Train Loss: 0.17. Train Acc (Test Acc): 94.00 (95.40)
Generation # 495. Train Loss: 0.10. Train Acc (Test Acc): 97.00 (96.60)
Generation # 500. Train Loss: 0.08. Train Acc (Test Acc): 99.00 (94.40)
rjoglekar74 at RAHULs-MBP in ~/Harvard/CSCIE63-Fall-2017/CNN Lab Week 12
$
```

What are we asked to do ?

- fix the first file (mnist.py) based on what you saw in file cnn_mnist.py
- Capture the Accuracy and Cross Entropy (summary) graphs from the corrected version of mnist2.py called mnist2_corrected.py

Assignment 12 Discussion

Problem 1

1. We need to know the progress of training and we cannot see it , so need to fix that
2. After you altered the code #1 above , run it again , Oh This is very poor

```
Generation # 580. Train Acc: 32.00
Generation # 585. Train Acc: 31.00
Generation # 590. Train Acc: 26.00
Generation # 595. Train Acc: 27.00
Generation # 600. Train Acc: 29.00
Generation # 605. Train Acc: 30.00
Generation # 610. Train Acc: 34.00
Generation # 615. Train Acc: 28.00
```

3. To narrow the problem , could we start by capping the epochs to 100 only – this is purely optional

```
rjoglekar74 at RAHULs-MBP in ~/Harvard/CSCIE63-Fall-2017/C
$ python mnist2-itr2.py
Extracting log3-1/data/train-images-idx3-ubyte.gz
Extracting log3-1/data/train-labels-idx1-ubyte.gz
Extracting log3-1/data/t10k-images-idx3-ubyte.gz
Extracting log3-1/data/t10k-labels-idx1-ubyte.gz
Starting run for lr_1E-04conv2fc2
2017-11-19 09:31:29.056699: I tensorflow/core/platform/cpu
 Your CPU supports instructions that this TensorFlow binar
use: SSE4.1 SSE4.2 AVX AVX2 FMA
Generation # 99. Itr1 Train Acc: 9.00
```

Assignment 12 Discussion

Problem 1

4. 9% is very bad ☺ What can we do better ? Perhaps change the Rate of learning?
5. Perhaps change the Optimizer – Adam to Momentum ?
6. What else could we do here ?
    a) mnist2.py uses mnist.train.next_batch() to create batches of training examples, where as cnn_mnist.py constructs batches manually using np.random.choice
    b) mnist2.py uses one-hot encoding to encode the labels, whereas cnn_mnist.py just uses the actual digit values
    c) mnist2.pygeneratessummariesandembeddingswhereascnn_mnist.py does not
    d) The two scripts have slightly different number of features/nodes in each layer, so let us examine the layers now . Do we see /understand what is going on in your FCC and Conv Layers. Could it be the Activation functions ?

7. Problem Solved – Whola !!! We get a 94% accuracy

```
rjoglekar74 at RAHULs-MBP in ~/Harvard/CSCIE63-Fall-2017/CNN Lab Wee
$ python mnist2-itr-final.py
Extracting log3/data/train-images-idx3-ubyte.gz
Extracting log3/data/train-labels-idx1-ubyte.gz
Extracting log3/data/t10k-images-idx3-ubyte.gz
Extracting log3/data/t10k-labels-idx1-ubyte.gz
Starting run for lr_1E-03conv2fc2
2017-11-19 09:50:58.989417: I tensorflow/core/platform/cpu_feature_g
 Your CPU supports instructions that this TensorFlow binary was not
use: SSE4.1 SSE4.2 AVX AVX2 FMA
Generation # 99. Train Acc: 94.00
```

Assignment 12 Discussion

Problem 2

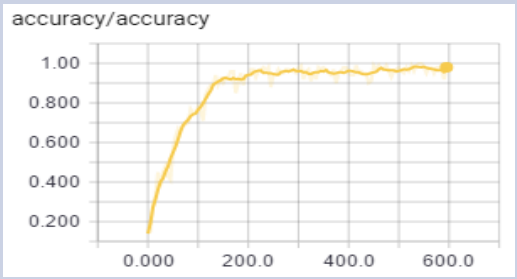
What are we asked to do ?

1. Run corrected version of mnist2.py for 4 different architectures (2 conv, 1 conv, 2 fully connected, 1 fully connected layer) and 3 values of the learning rate
2. One learning rate choose the one you selected in Problem 1 and then add one smaller and one larger learning rate around that one.
3. Collect execution times, final (smoothed) accuracies and final cross entropies for different models and provide tabulated presentation of the final results of different models

Assignment 12 Discussion

Problem 2

Present your results as

learning rate	CONV	FC	time	Graphs
0.01	2	2	Starting run for: Learning rate 0.010000 2 CONV 2 FC num generations = 600 0.000000 Done 50.083472 Done Elapsed time = 185.996742	

Learning rate	*# Conv layers*	*# FC layers*	*Accuracy (smoothed)*	*Cross entropy*	*Execution time*
10^{-3}	1	2	0.9	0.3759	54s
10^{-3}	2	1			

Assignment 12 Discussion

Problem 3

- Modify file cnn_mnist.py so that it publishes its summaries to the TensorBoard
- Describe the differences if any between the graph of this program and the graph generated by mnist2.py script running with 2 convolutional and 2 fully connected layers.

Solution Approach

- Initialize the File writer

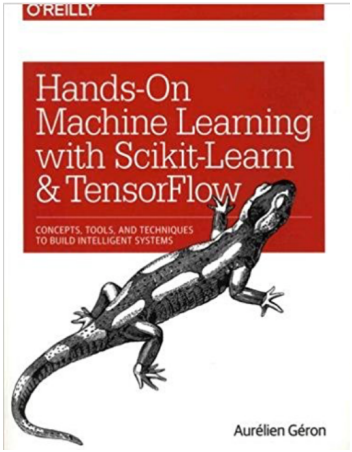
  writer = tf.summary.FileWriter(data_dir)
  writer.add_graph(sess.graph)
- We add a Scalar Summary for both Loss and Accuracy

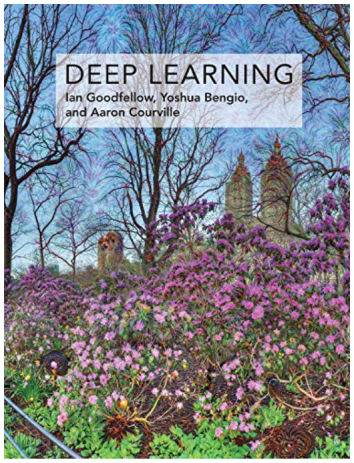
  tf.summary.scalar("accuracy", accuracy)
- Next merge summaries together

- Finally do a Writer.addsummary

  writer.add_summary(summary, i)

References





– Yoshua Bengio, Learning Deep Architectures for AI, Foundations and Trends in Machine Learning, 2(1), pp.1-127, 2009.
– LeCun, Chopra, Hadsell, Ranzato, Huang: A Tutorial on Energy-Based Learning, in Bakir, G. and Hofman, T. and Scholkopf, B. and Smola, A. and Taskar, B. (Eds), Predicting Structured Data, MIT Press, 2006
– LeCun, Bottou, Bengio and Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998
– Jarrett, Kavukcuoglu, Ranzato, LeCun: What is the Best Multi-Stage Architecture for Object Recognition?, Proc. International Conference on Computer Vision (ICCV'09), IEEE, 2009
- Kavukcuoglu, Sermanet, Boureau, Gregor, Mathieu, LeCun: Learning Convolutional Feature Hierachies for Visual Recognition, Advances in Neural InformationProcessing Systems (NIPS 2010), 23, 2010