Lecture Notes for

Machine Learning in Python

Professor Eric Larson

Basic Convolutional Neural Networks

Logistics and Agenda

- Logistics
  - Wide/Deep due soon!
  - Remember: Feel free to turn in late for partial credit.
- Agenda
  - Wide/Deep Finish Demo and Town Hall
  - Basic CNN architectures and Demo

Class Overview, by topic

Table Data Visualization

Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing

Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression

Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.

Numpy
Detailed mathematics for NN operations

Wide and Deep Networks | **Convolutional Networks** | **Recurrent Networks**

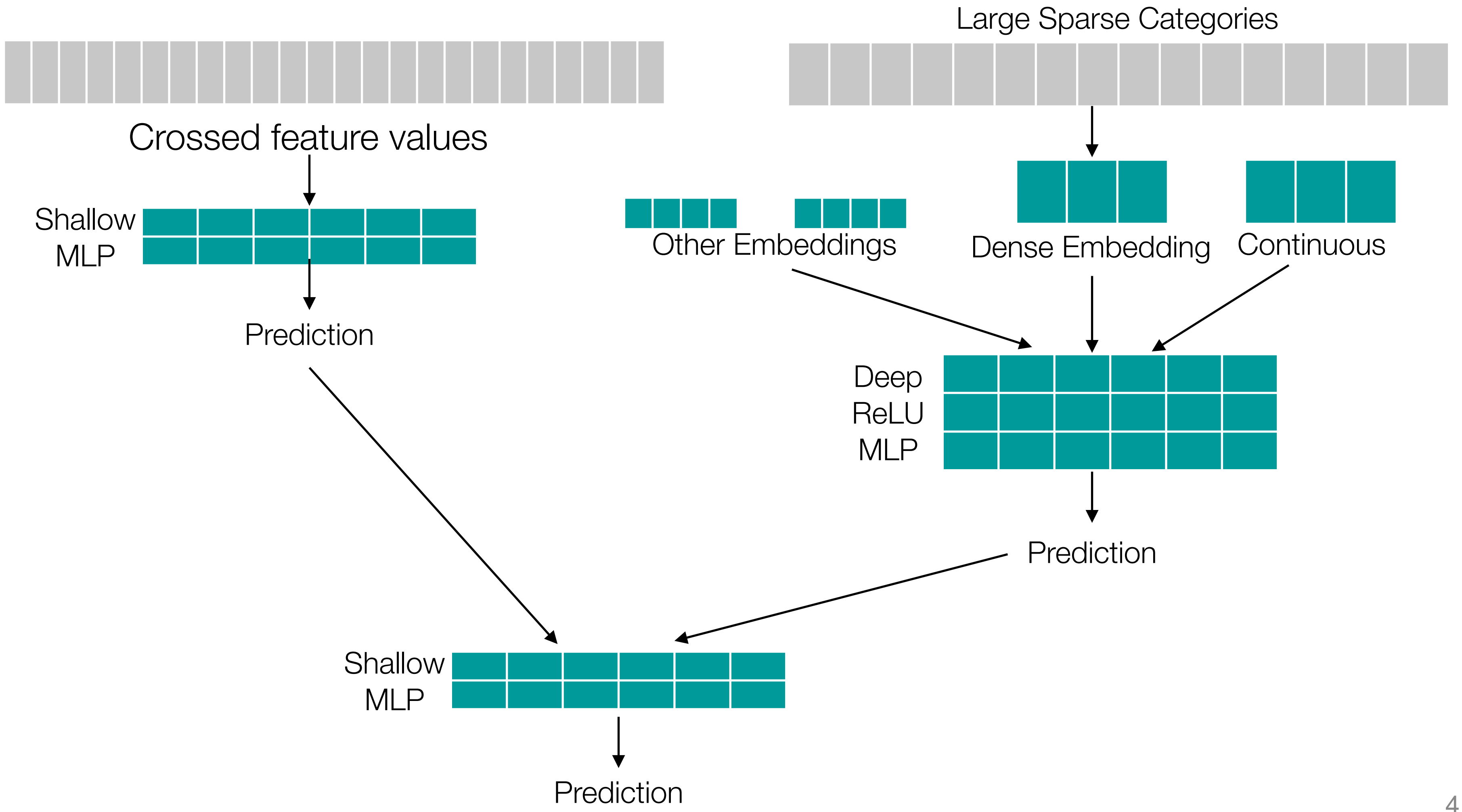
Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models

ConceptNet
Case studies

Last Time:

- Deep refers to increasingly smaller hidden layers
- Embed into sparse representations via ReLU



“Finish” **Demo**

10. Keras Wide and Deep.ipynb

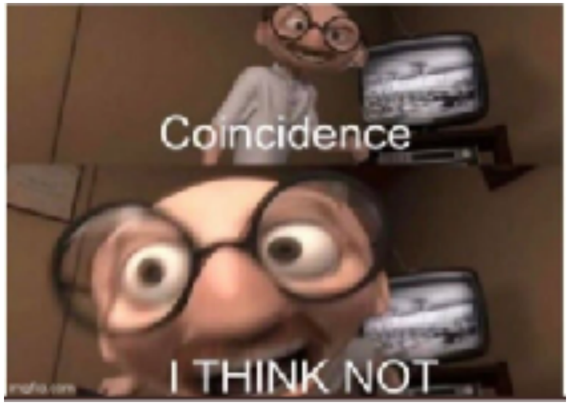
The awful dataset:
Toy Census Data Example

Other tutorials:

https://www.tensorflow.org/tutorials/wide_and_deep

Town Hall,
Wide and Deep Networks

When $p < 0.05$



Convolutional Neural Networks

IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

Reminder: Convolution

$$\sum\left(\mathbf{I}\left[i\pm\frac{r}{2}, j\pm\frac{c}{2}\right]\odot\mathbf{k}\right)=\mathbf{O}[i,j]$$

output image at pixel $\mathbf{i,j}$

input image at r x c range of pixels centered in i,j

kernel of size, r x c usually r=c

0	0	0	0	0	0	0	0	0
0	1	2	3	4	12	9	8	0
0	5	2	3	4	12	9	8	0
0	5	2	1	4	10	9	8	0
0	7	2	1	4	12	7	8	0
0	7	2	1	4	14	9	8	0
0	5	2	3	4	12	7	8	0
0	5	2	1	4	12	9	8	0
0	0	0	0	0	0	0	0	0

input image, $\mathbf{I}$

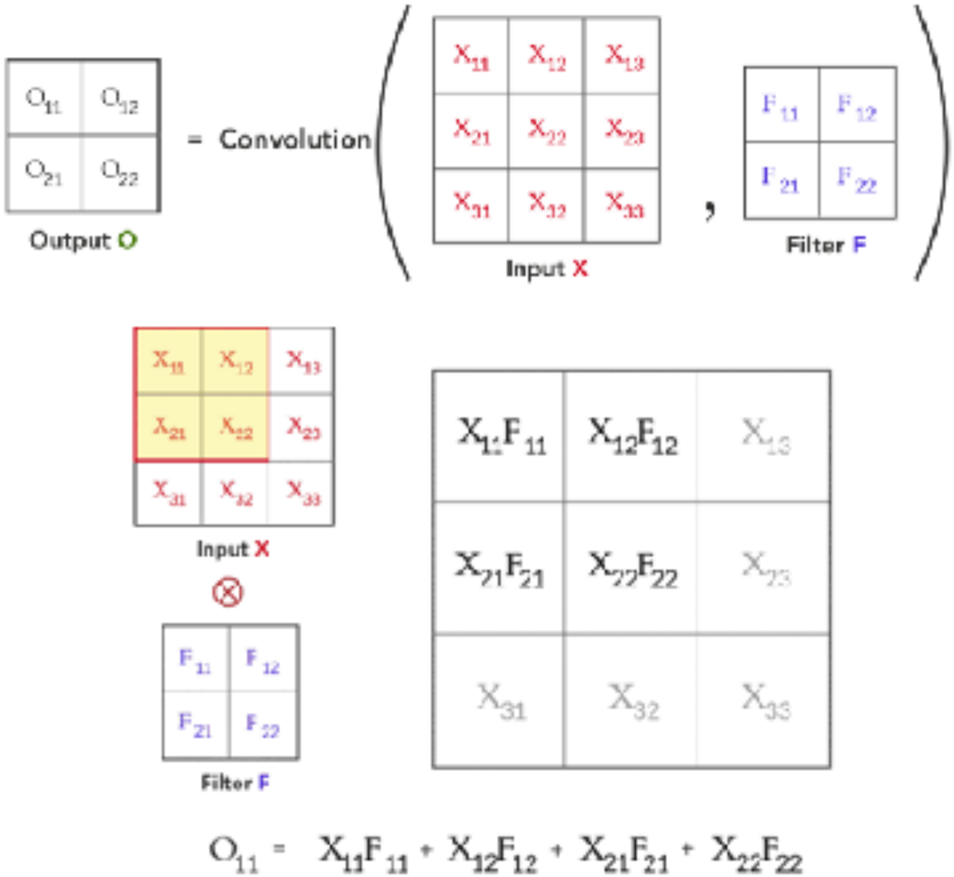
1	2	1
2	4	2
1	2	1

kernel filter, $\mathbf{k}$ 3x3

20	21	36	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…
…	…	…	…	…	…	…

output image, $\mathbf{O}$

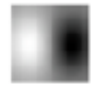
Reminder: Convolution



https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

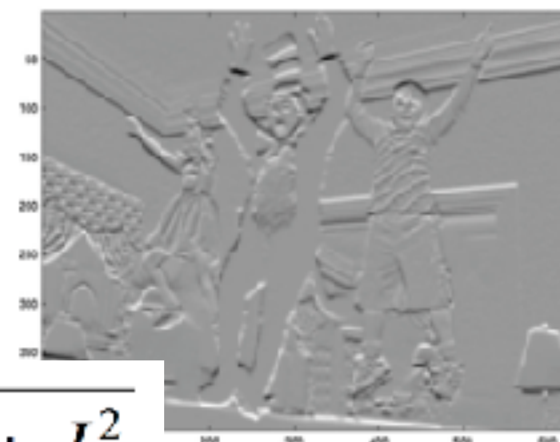
What we did before

- the gradient (2D derivative)



J_x

J_y

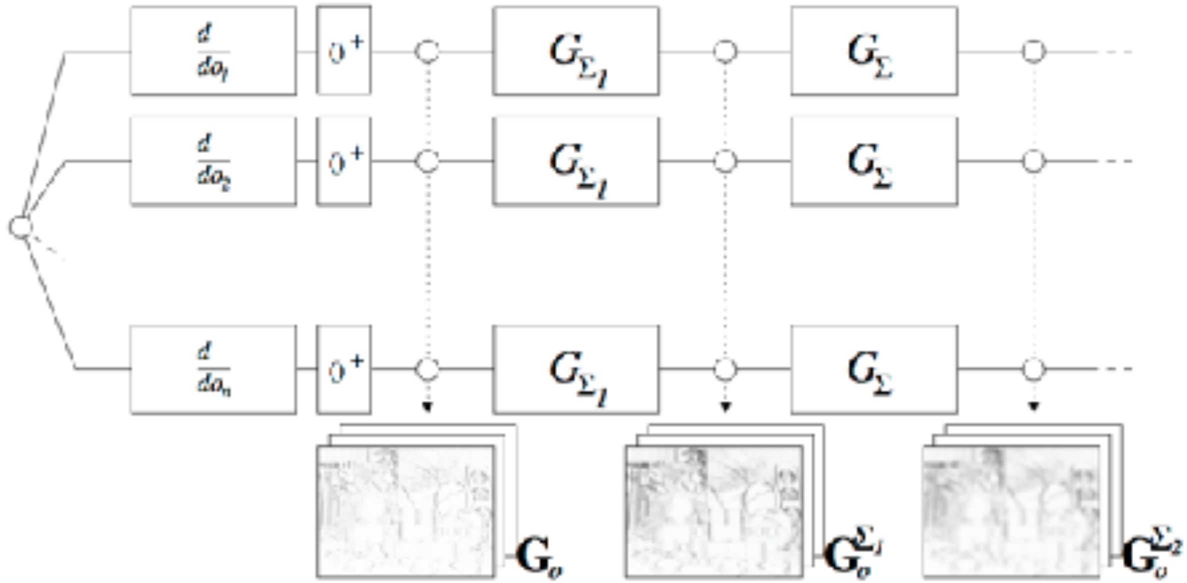


$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

$$O = \tan^{-1}(Jx / Jy)$$

images: Jianbo Shi, Upenn

What we did before



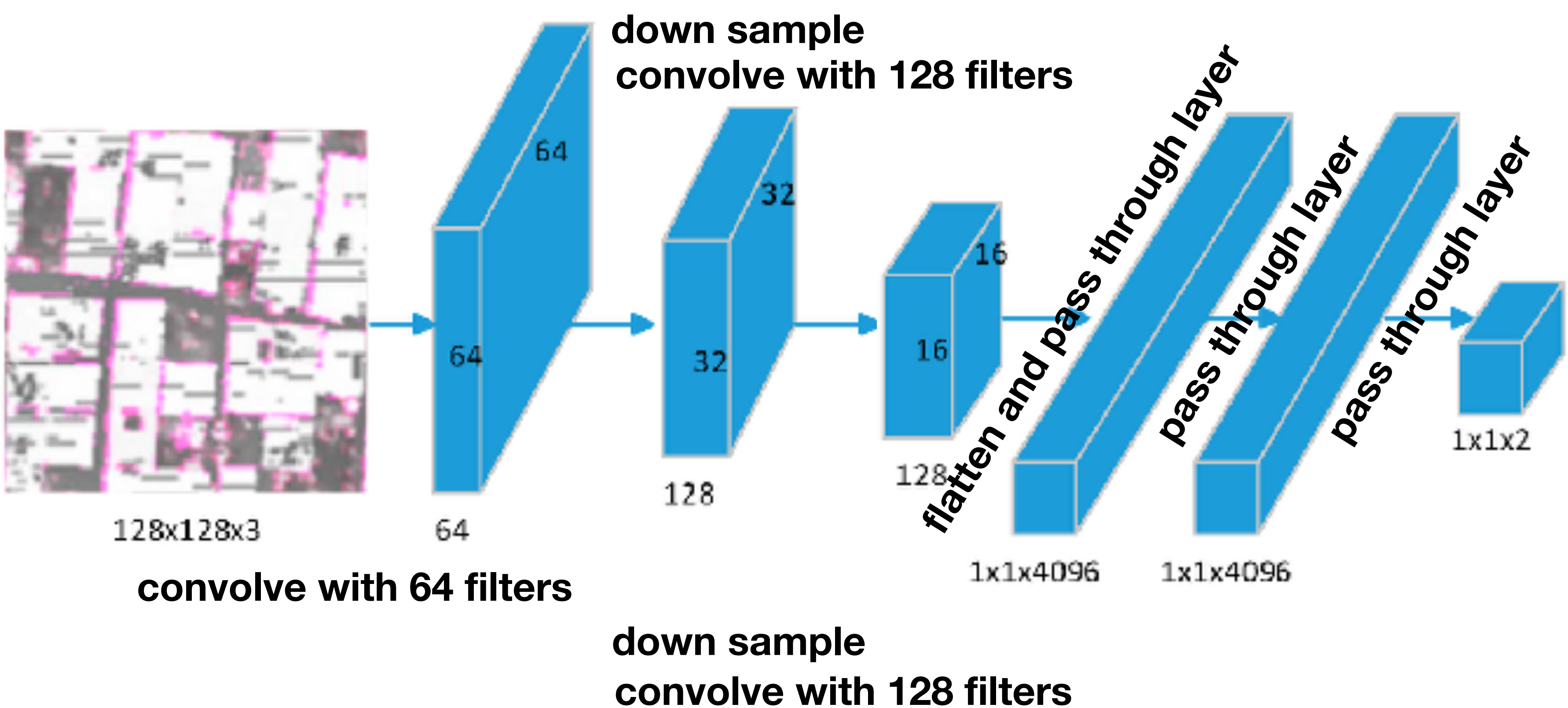
take normalized histogram at point u,v

$$\widetilde{\mathbf{h}}_{\Sigma}(u,v) = \left\| \left[\mathbf{G}_1^{\Sigma}(u,v), \ldots, \mathbf{G}_H^{\Sigma}(u,v)\right]^{\top} \right\|$$

$$\begin{aligned}
\mathcal{D}(u_0, v_0) = \\
\Big[\ & \widetilde{\mathbf{h}}_{\Sigma_1}^{\top}(u_0, v_0), \\
& \widetilde{\mathbf{h}}_{\Sigma_1}^{\top}(\mathbf{l}_1(u_0, v_0, R_1)), \cdots, \widetilde{\mathbf{h}}_{\Sigma_1}^{\top}(\mathbf{l}_T(u_0, v_0, R_1)), \\
& \widetilde{\mathbf{h}}_{\Sigma_2}^{\top}(\mathbf{l}_1(u_0, v_0, R_2)), \cdots, \widetilde{\mathbf{h}}_{\Sigma_2}^{\top}(\mathbf{l}_T(u_0, v_0, R_2)),
\end{aligned}$$

Tola et al. “Daisy: An efficient dense descriptor applied to wide-baseline stereo.” Pattern Analysis and Machine Intelligence, IEEE Transactions

Anatomy of a convolutional network

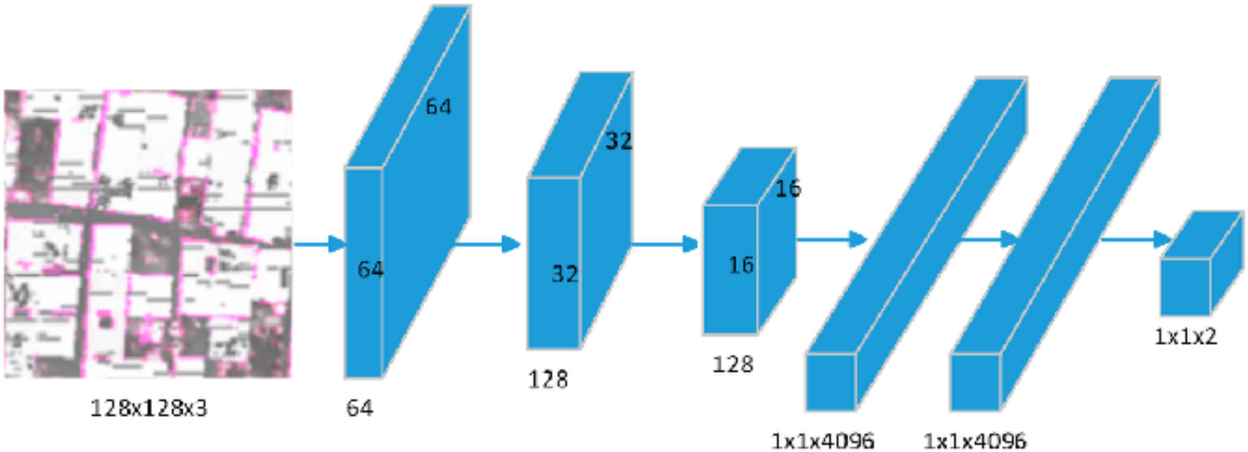


Blue Tensors: Outputs of Each Layer

Learned Params: Weights in Each Filter and Fully Connected Layer

CNN Overview

- First layer(s):
  - convolution
  - activation
  - pooling
  - Each pooling layer *can* make the input image "smaller"
    - allows for "Information Distillation"
    - less dependence on exact pixels



- Final layers are densely connected
  - typically multi-layer perceptrons

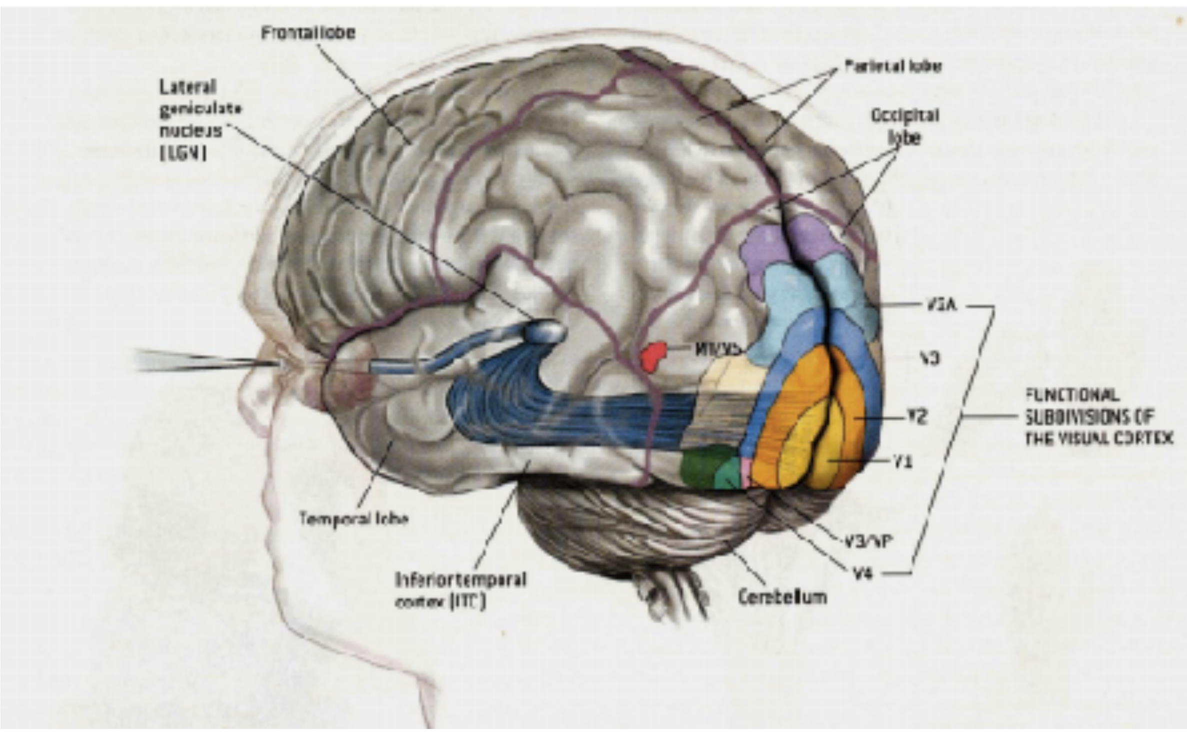
CNN Overview: Self Test

- First layer(s):
  - convolution
  - nonlinearity
  - pooling
  - Each pooling layer *can* make the input image “smaller”
    - allows for “Information Distilation”
    - less dependence on exact pixels
- Final layers are densely connected
  - typically multi-layer perceptrons

- Where are unstable gradients **most** problematic?
  - (A) During Convolution Layer(s) updates
  - (B) During Fully Connected Layer(s) updates
  - (C) Both A and B
  - (D) They are not a problem

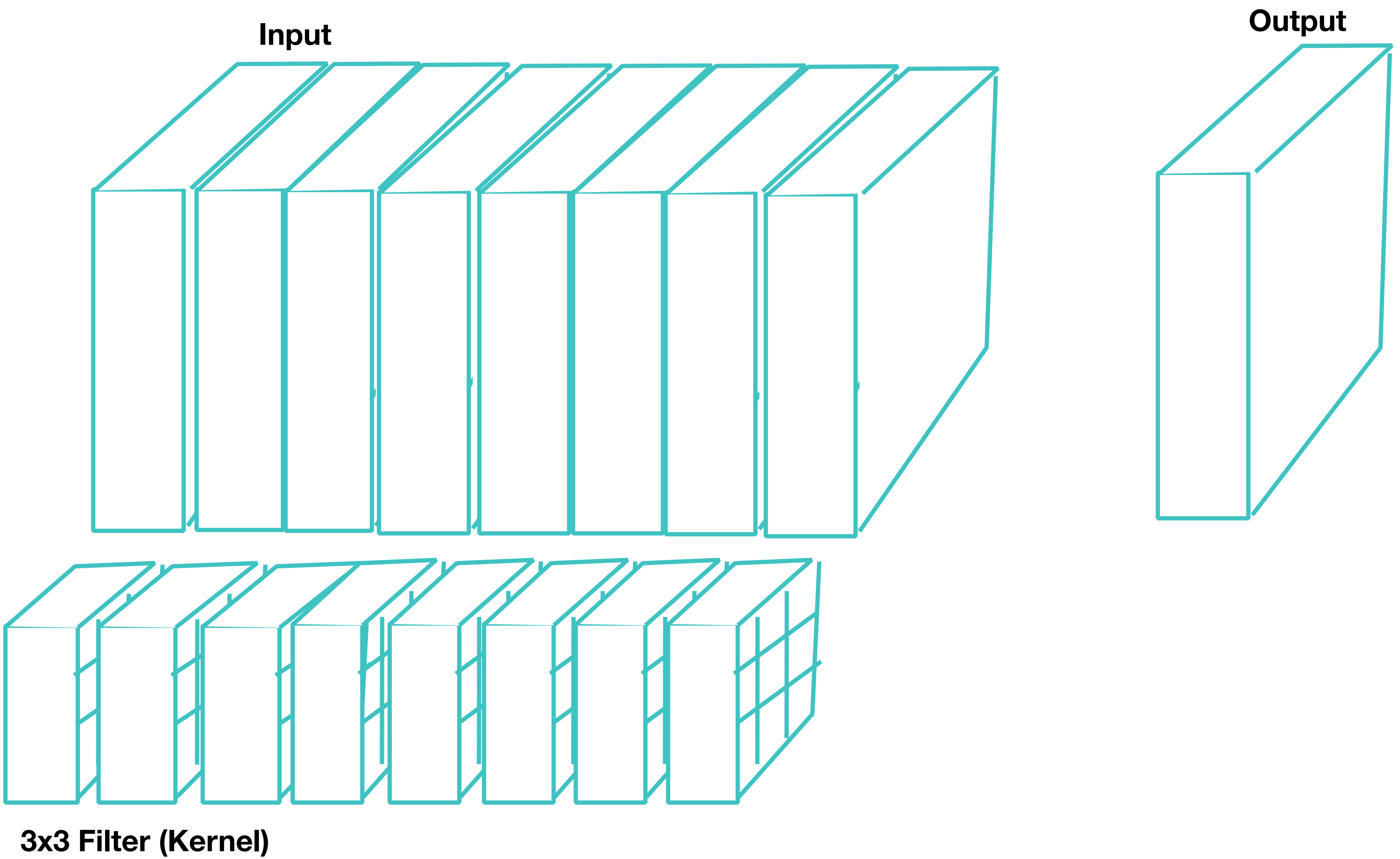
CNN Filtering

- Why perform lots of filtering?
  - “recall” gabor filtering?



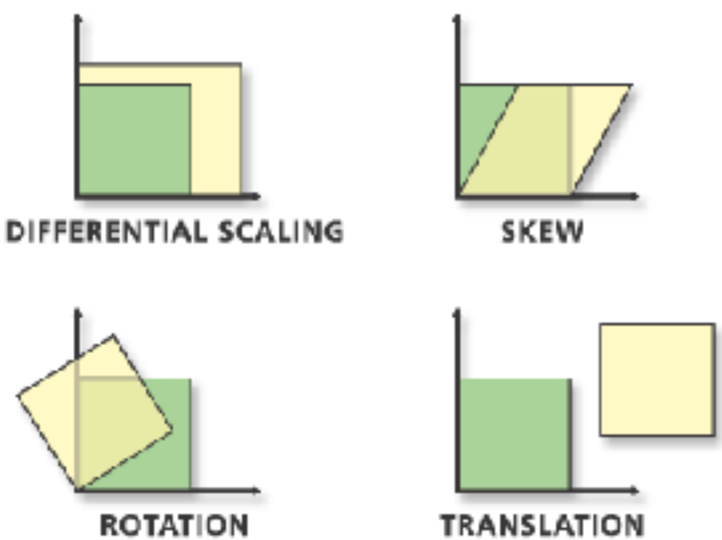
V1	Motion
V2	Stereo
V3	Color
V3a	Texture segregation
V3b	Segmentation, grouping
V4	Recognition
V7	Face recognition
MT	Attention
MST	Working memory/mental imagery

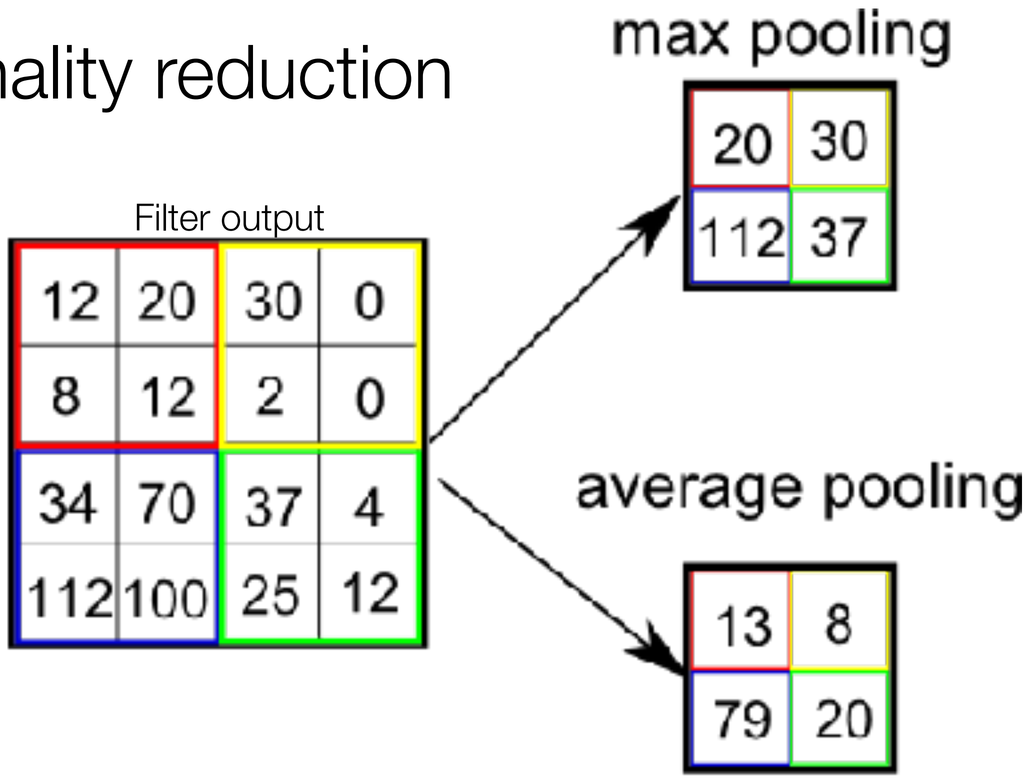
Convolution in a CNN



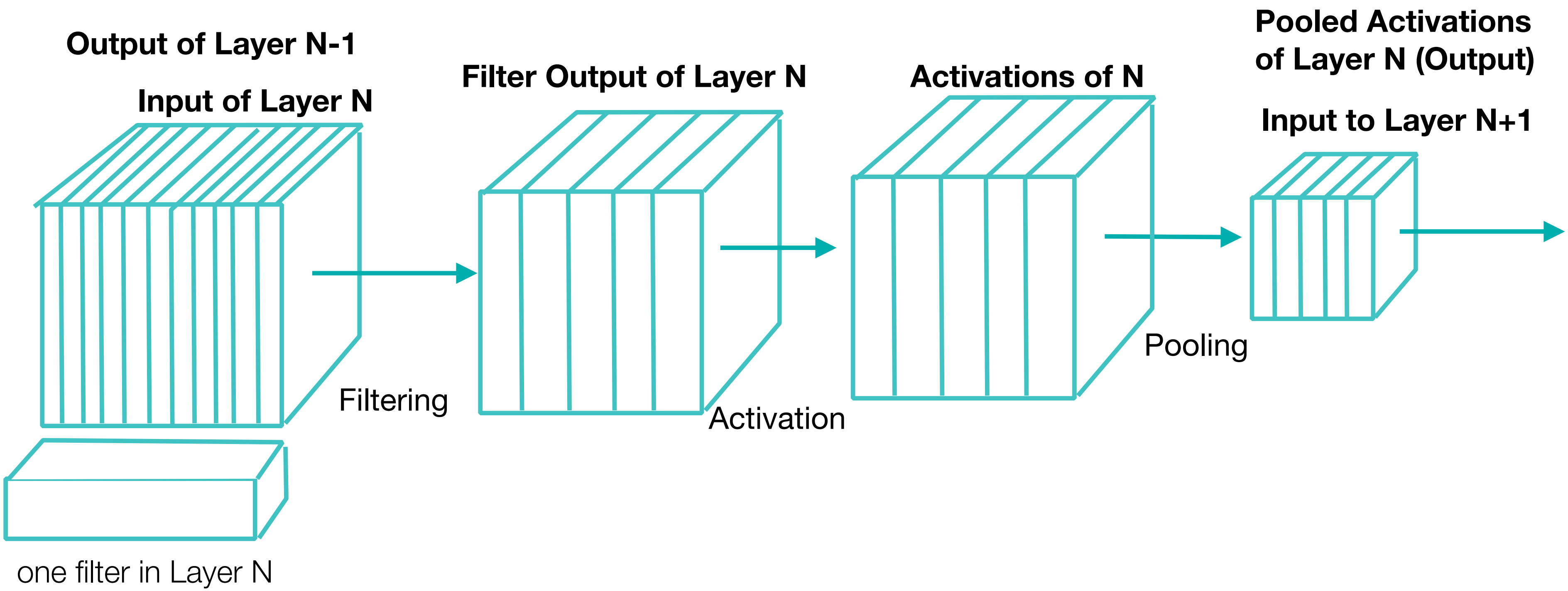
CNN Pooling

- Why perform pooling?
- Why max pooling?
  - reduce translation effects
  - **mostly**: dimensionality reduction



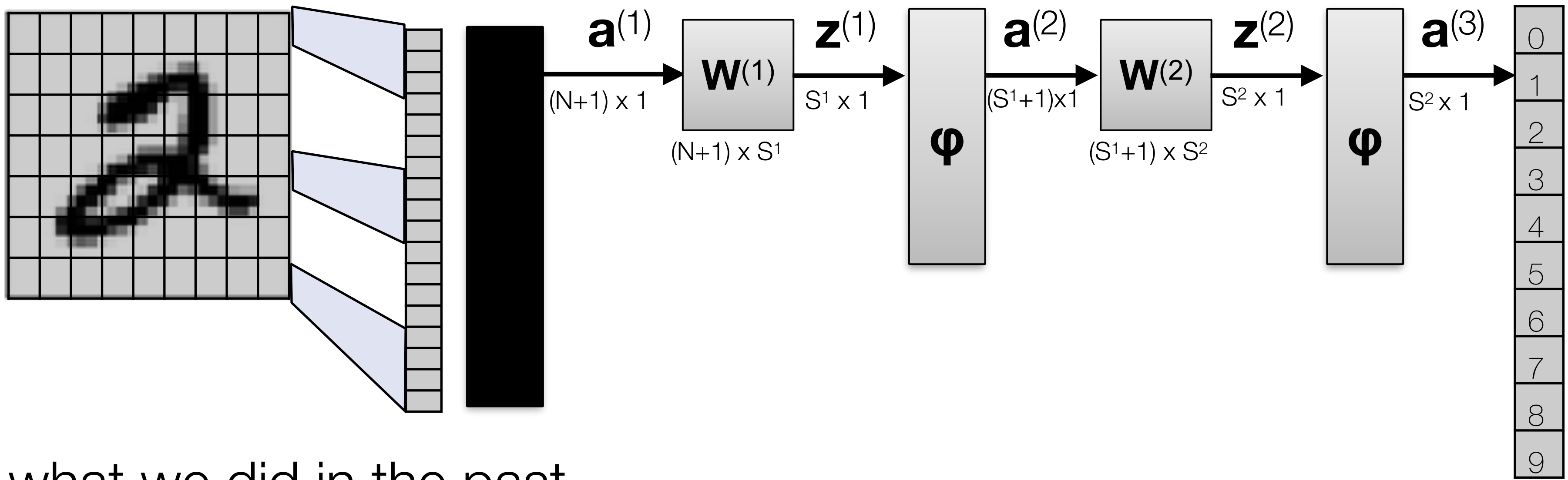


CNNs: Putting it together



Structure of Each Tensor: Channels x Rows x Columns

Simple Example: From Fully Connected to CNN



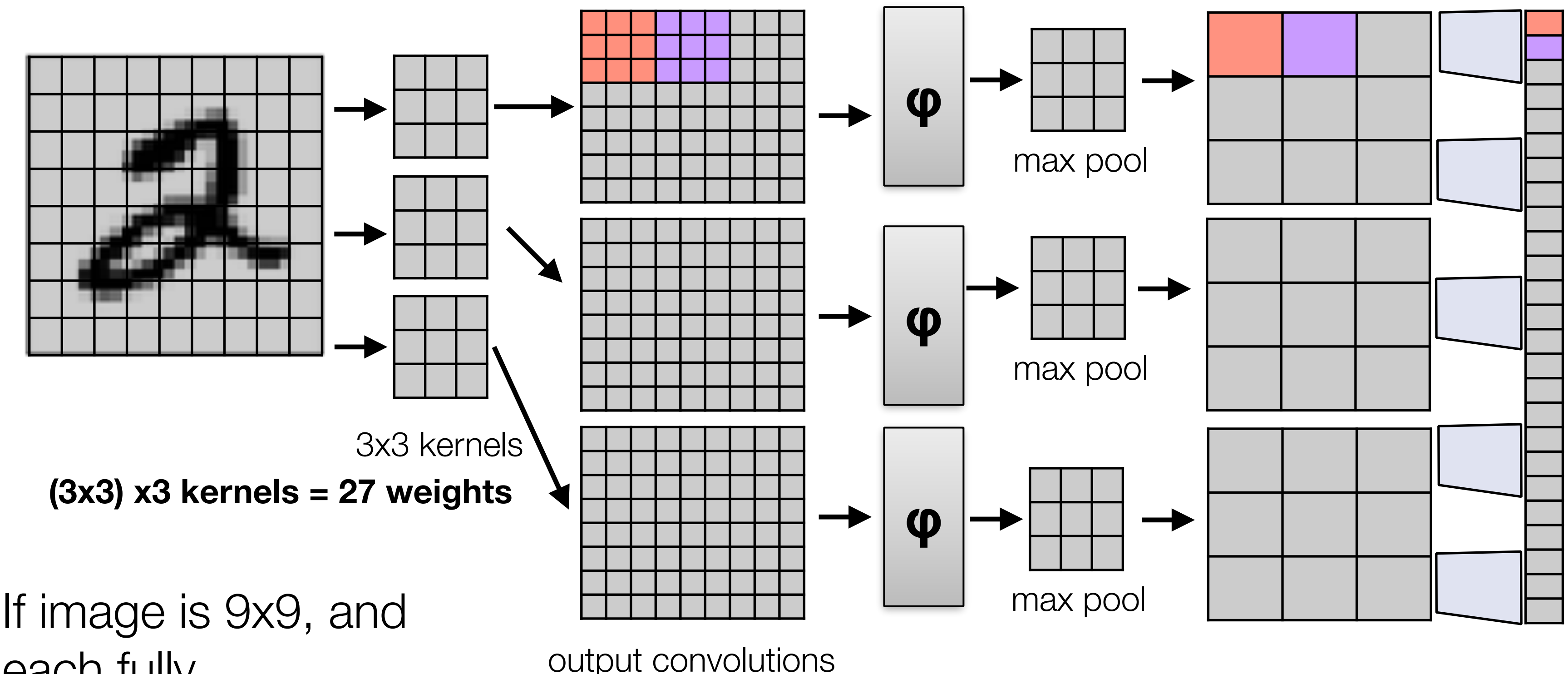
what we did in the past

If image is 9x9, and each fully connected layer is 20 hidden neurons wide, how many parameters are in this NN (ignore bias)?

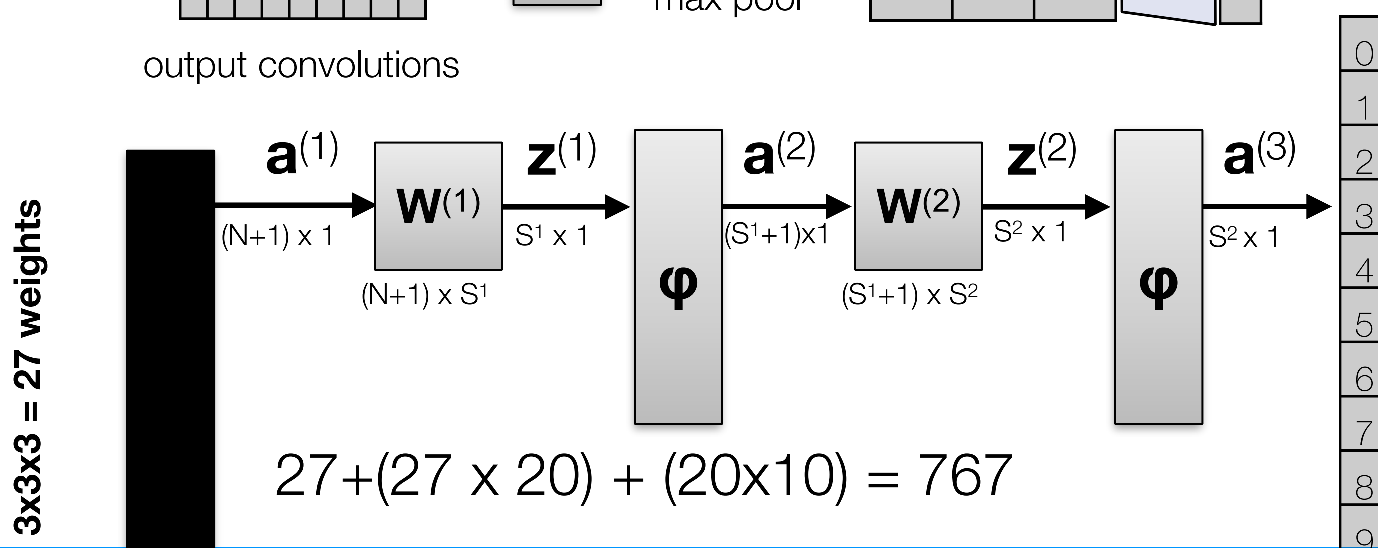
$(K^2 \times 20) + (20\text{x}10) = 200 + 20\ K^2$

for $9\text{x}9 = 200 + 20\text{x}9^2 =\ 1{,}820$ parameters

Simple Example: From Fully Connected to CNN

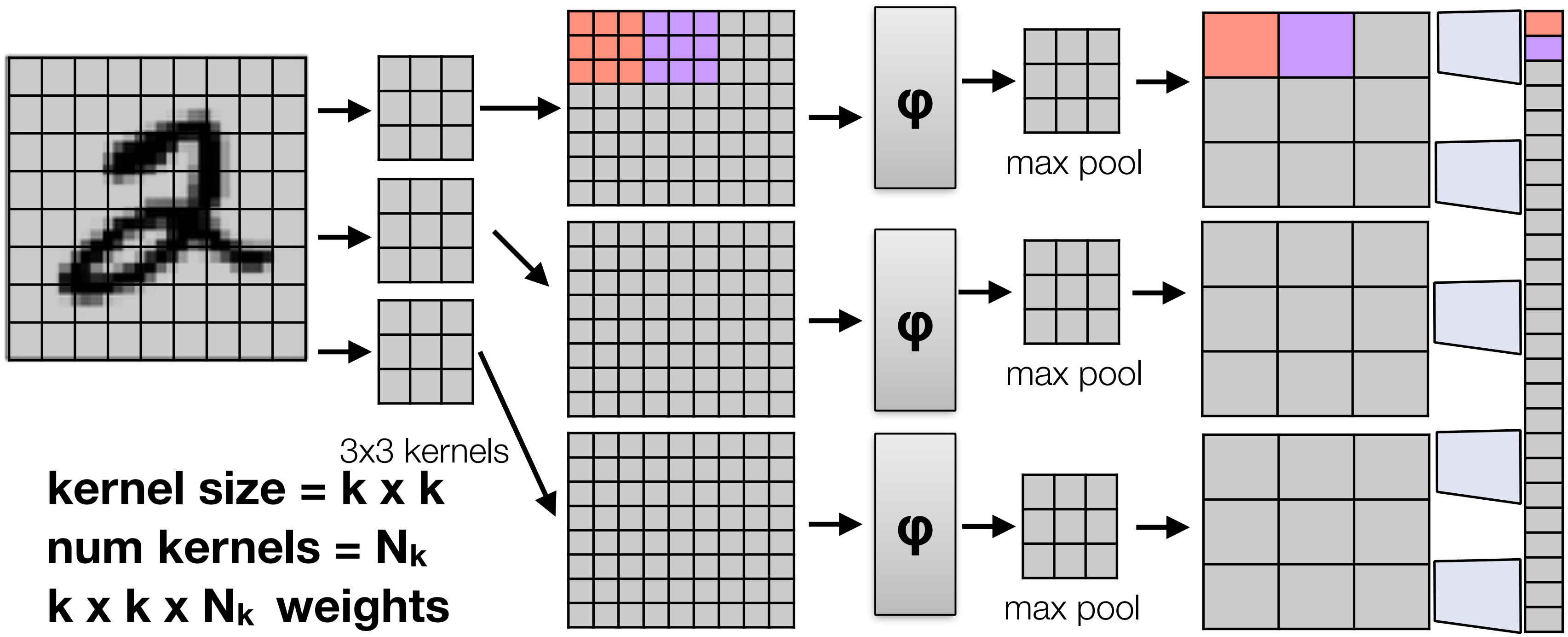


If image is 9x9, and each fully connected layer is 20 hidden neurons wide, how many parameters are in this NN (ignore bias)?



$$27+(27 \times 20) + (20 \times 10) = 767$$

Simple Example: From Fully Connected to CNN

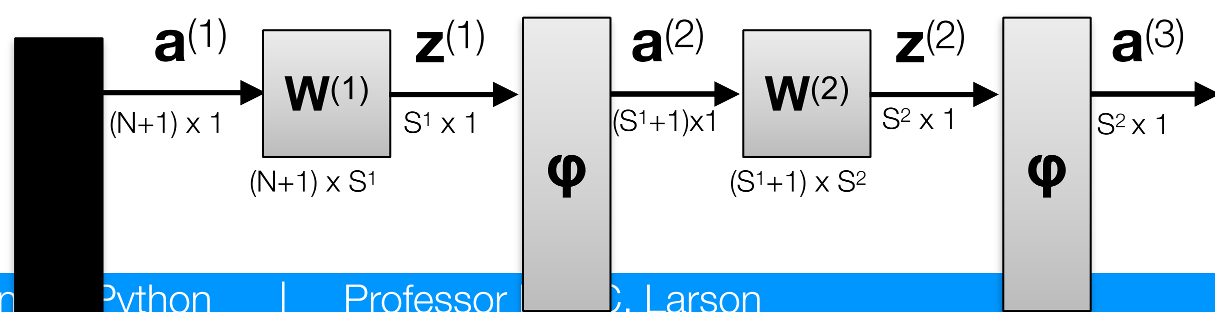


kernel size = k x k
num kernels = N_k
$k \times k \times N_k$ weights

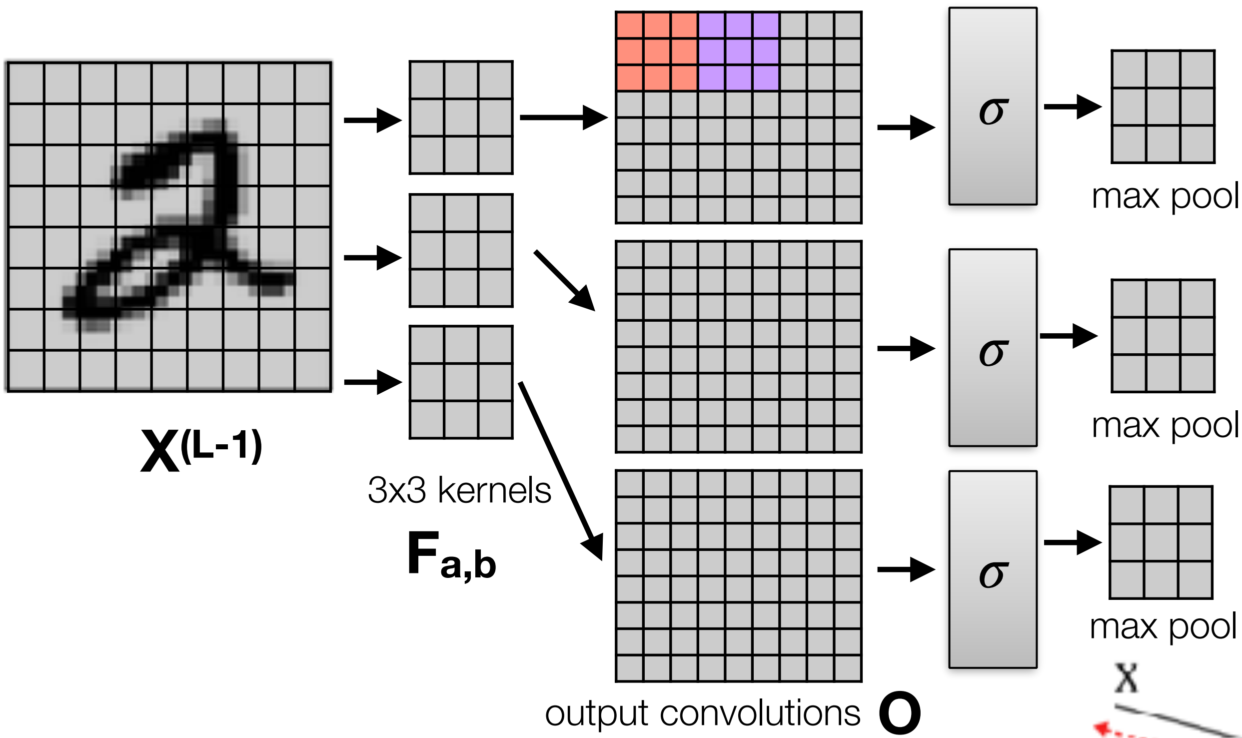
pooling outputs
if stride = k
$(K/k) \times (K/k) \times N_k$

convolutional params
$N_k \times k^2$ ← filter dimension
num filters

Input to MLP
$N_k \times (K^2/k^2)$
image dimension



CNN gradient



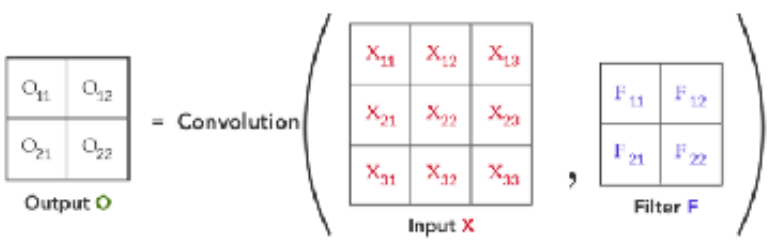
Derivative of max pool is easy:

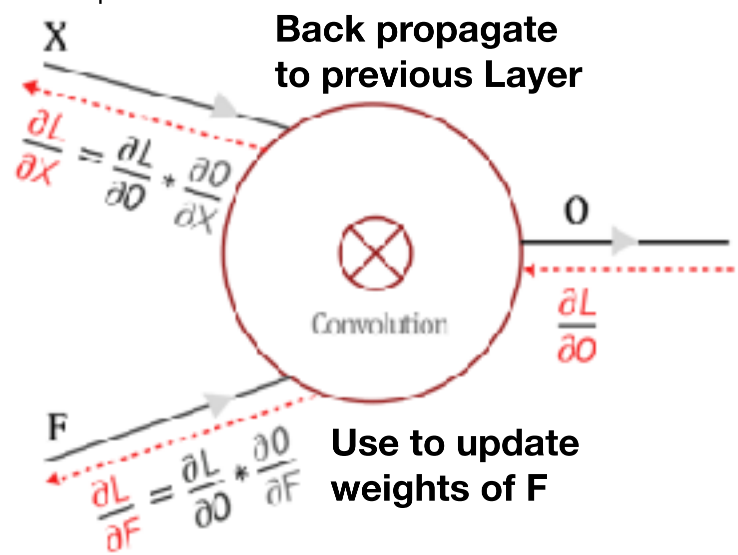
for each input X_i

$$\text{pool}'(X_i) = \begin{cases} 1 & \text{if } X_i \text{ is max} \\ 0 & \text{else} \end{cases}$$

$$\mathbf{X}^{(L)} = \text{pool}(\sigma(\mathbf{O}))$$

Derivative of convolution is more involved:





https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

Next Lecture

- More CNN architectures and CNN history