Lecture Notes for

Machine Learning in Python

Professor Eric Larson

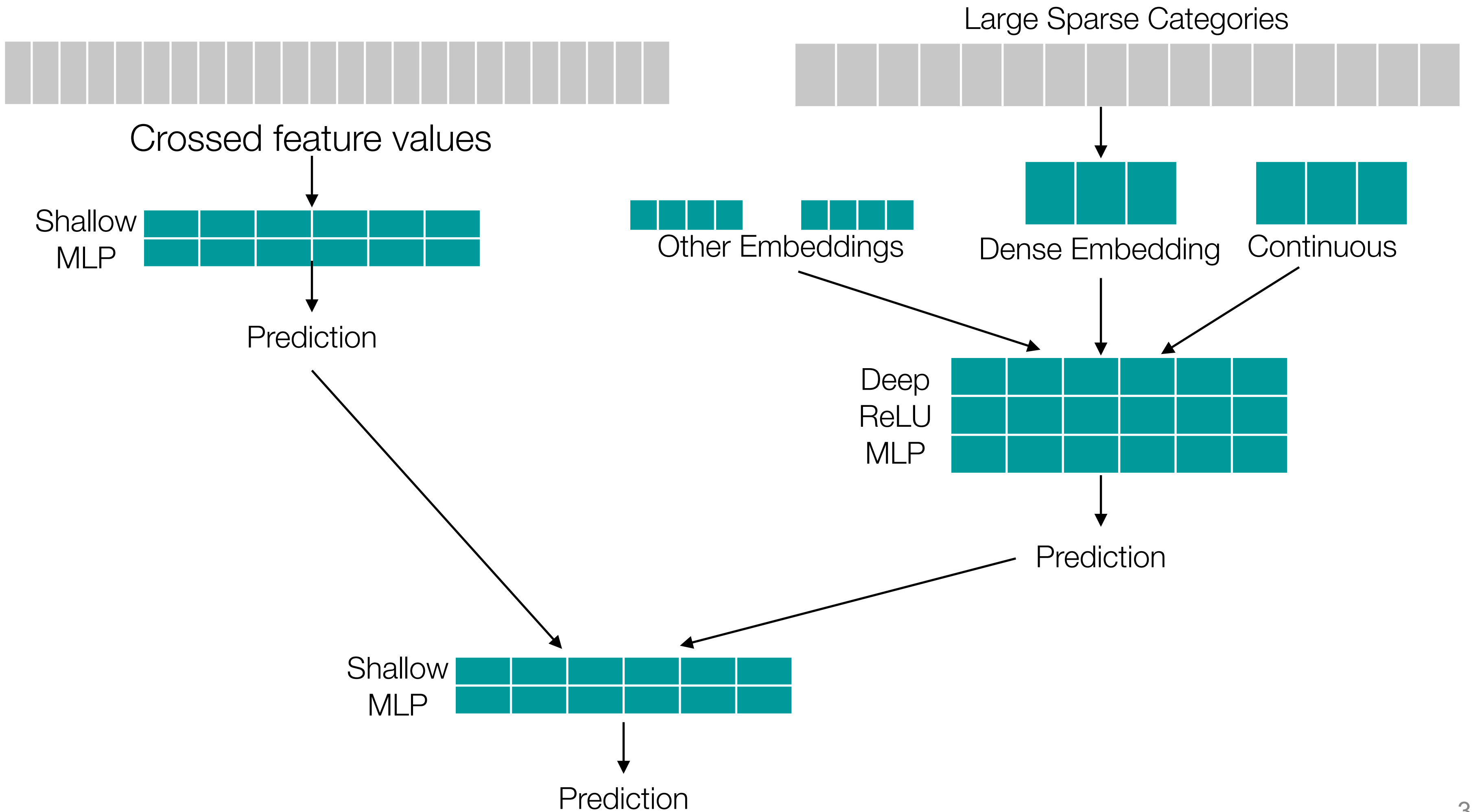
Basic Convolutional Neural Networks

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

Last Time:

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



Wide and Deep   “Finish” Demo

```
10. Keras Wide and Deep.ipynb
```

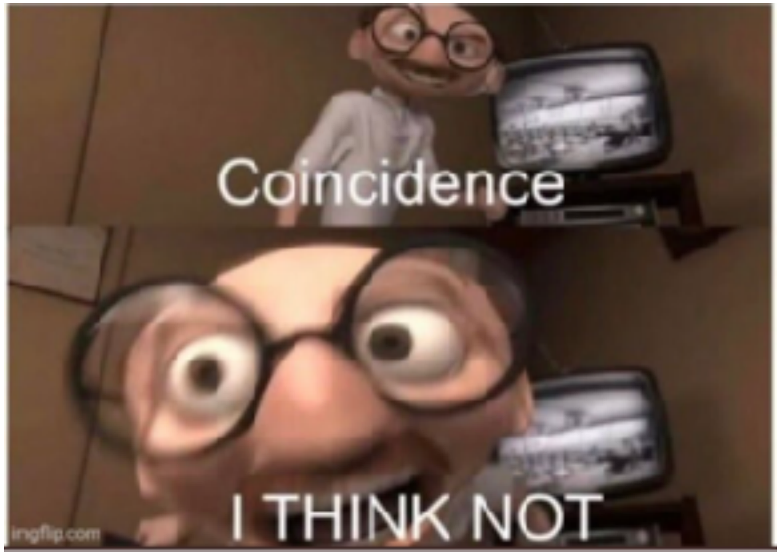
The awful dataset:
Toy Census Data Example

Other tutorials:

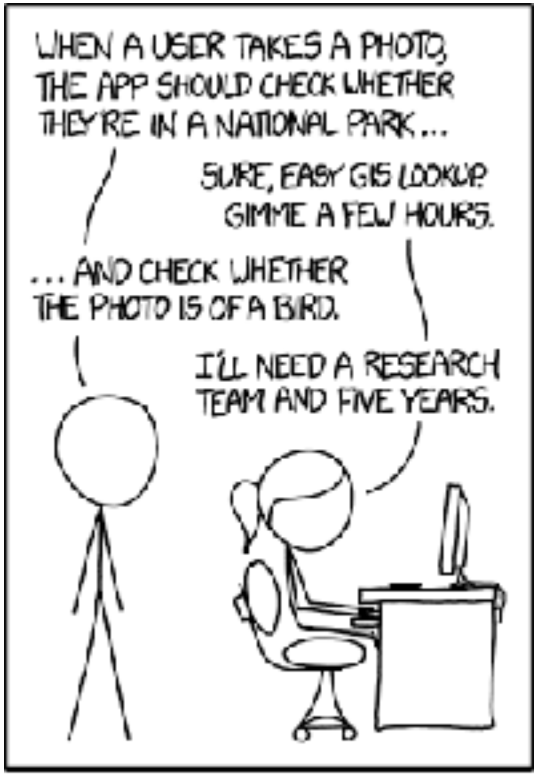
https://www.tensorflow.org/tutorials/wide_and_deep

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, $\mathbf{I}$

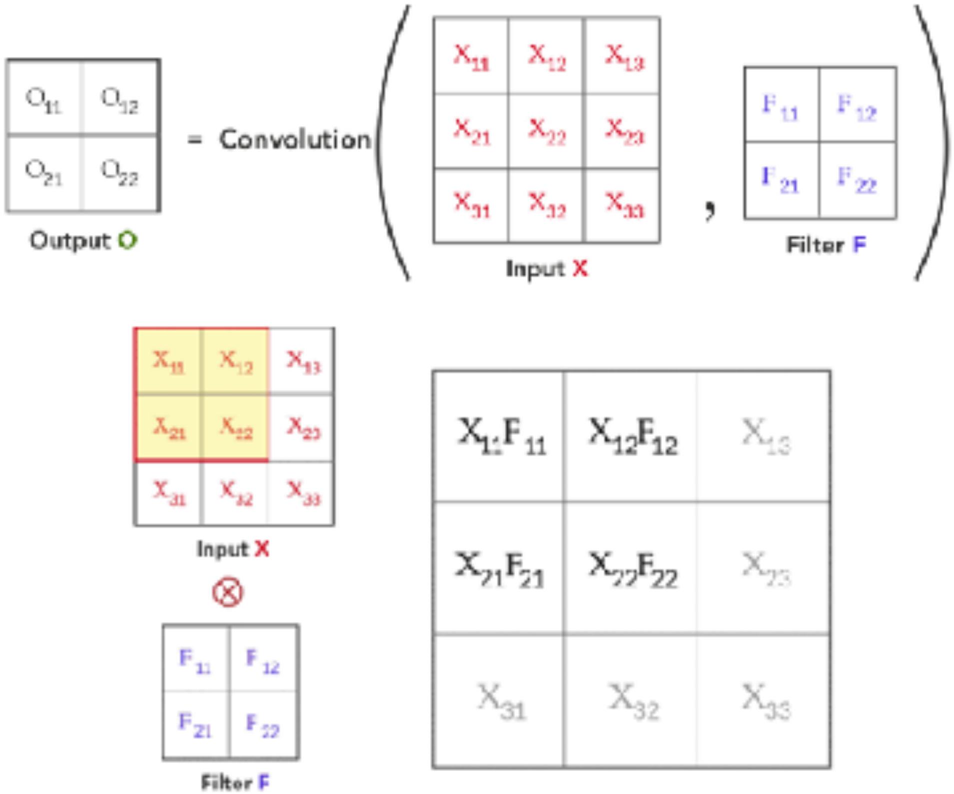
1	2	1
2	4	2
1	2	1

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

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

output image, $\mathbf{O}$

Reminder: Convolution



$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

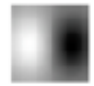
https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

What we did before

- the gradient (2D derivative)

I

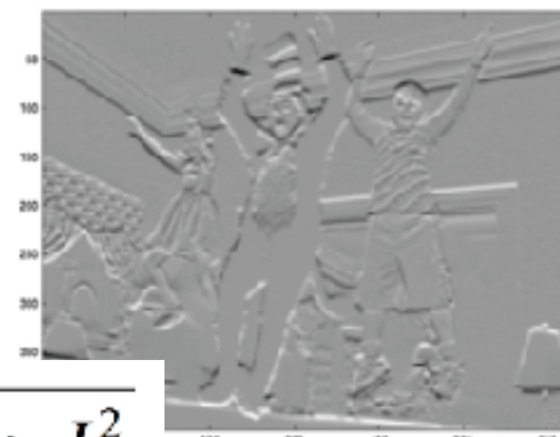
filter



filter

J_x

J_y

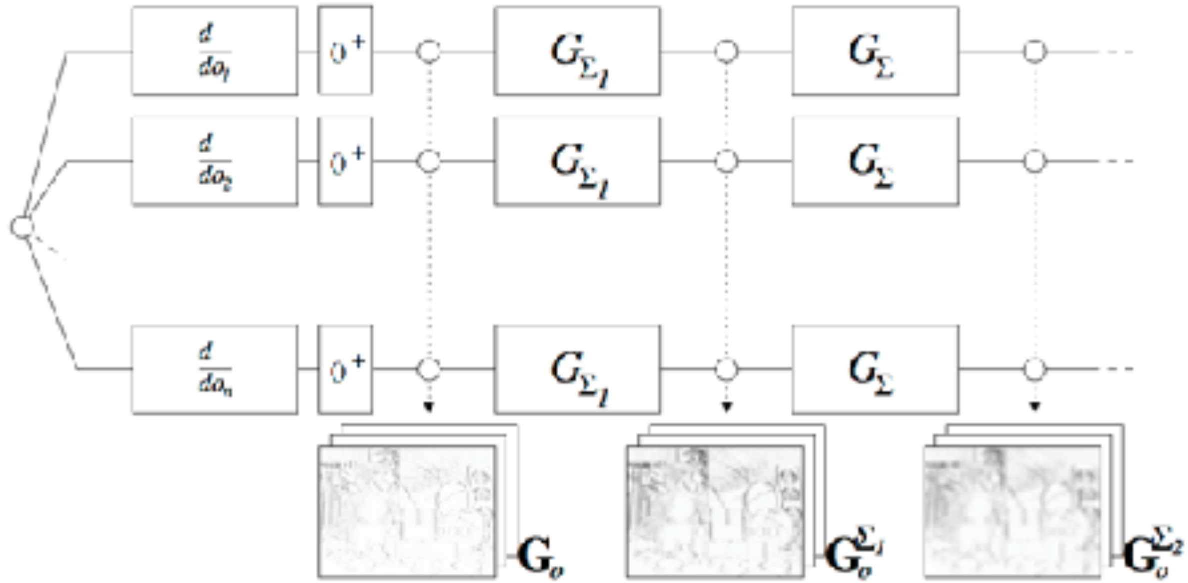


$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

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

images: Jianbo Shi, Upenn

What we did before



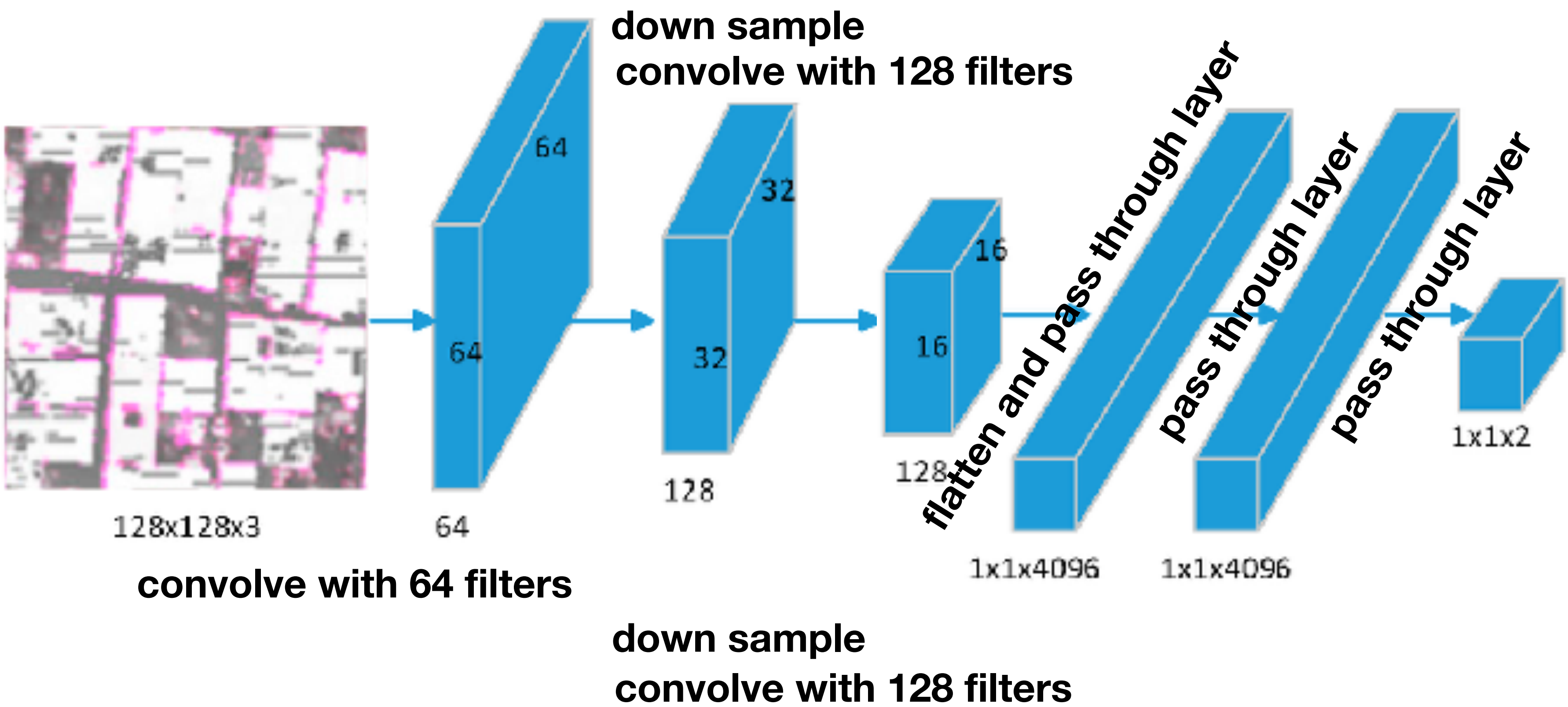
take normalized histogram at point u,v

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

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

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

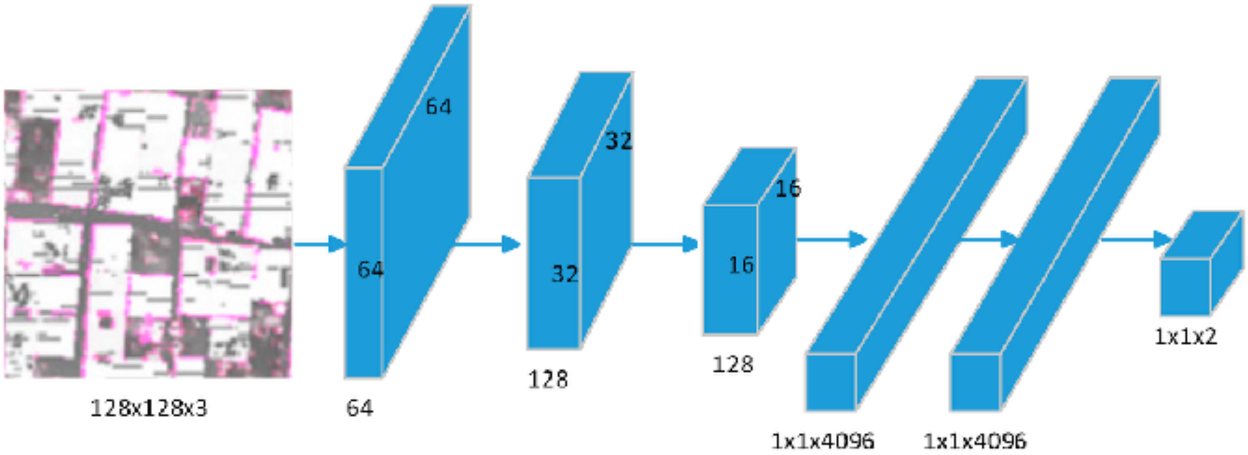
Anatomy of a convolution



Blue Tensors: Outputs of Each Layer

Learned Params: Weights in Each Filter and Fully Connected Layer

CNN Overview



- First layer(s):
  - convolution
  - nonlinearity
  - 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

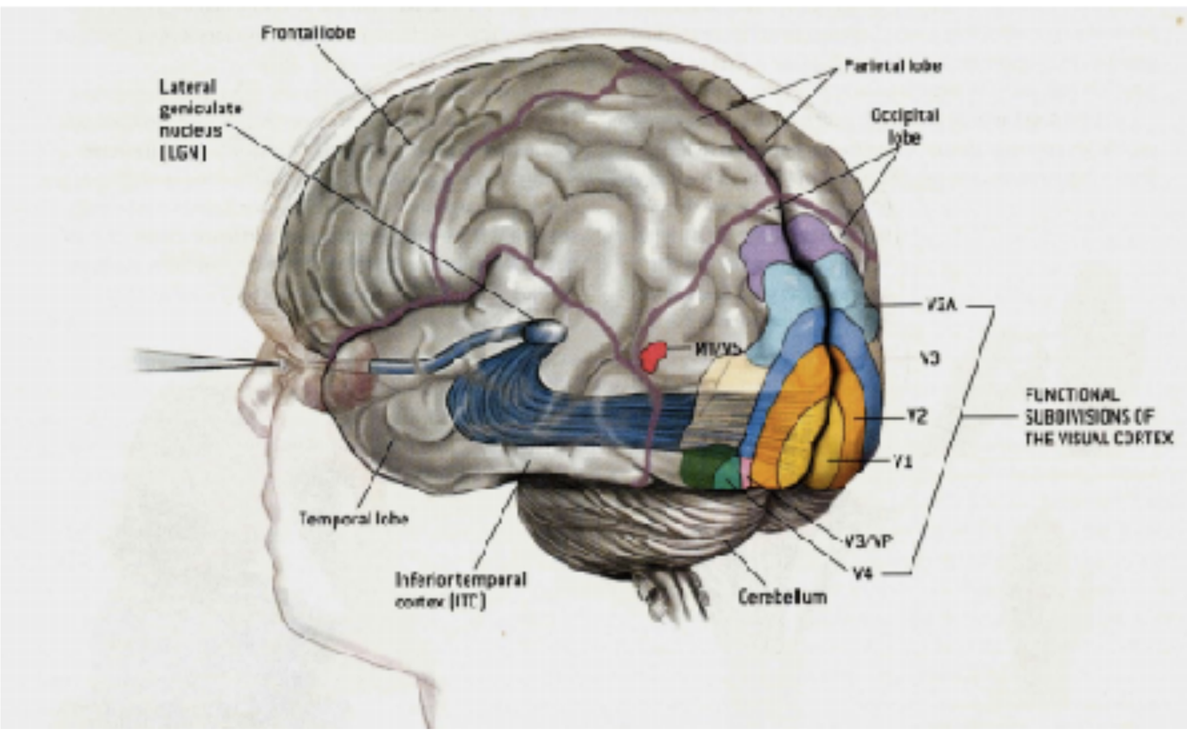
CNN Overview: Self Test

- 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

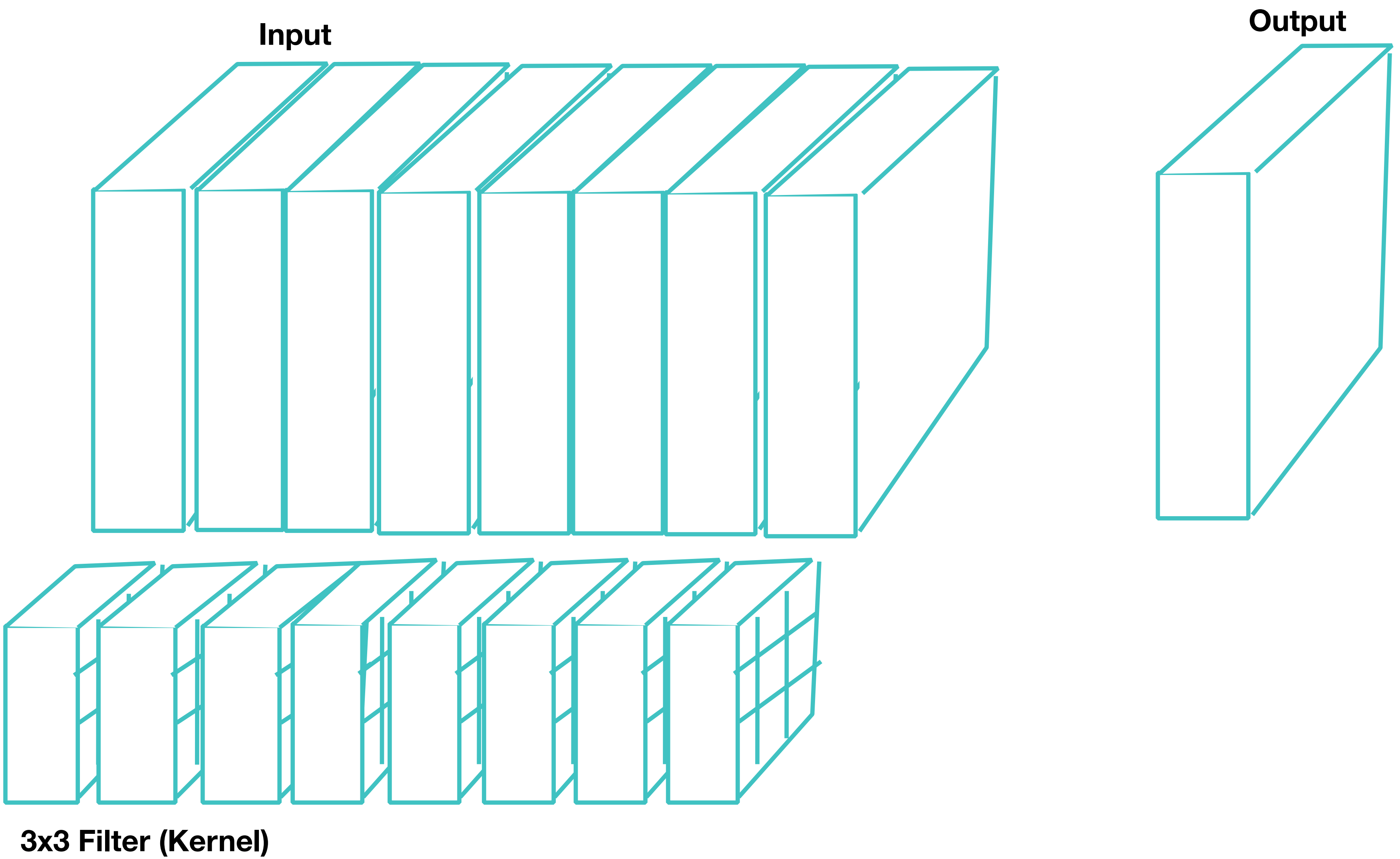
CNN Filtering

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



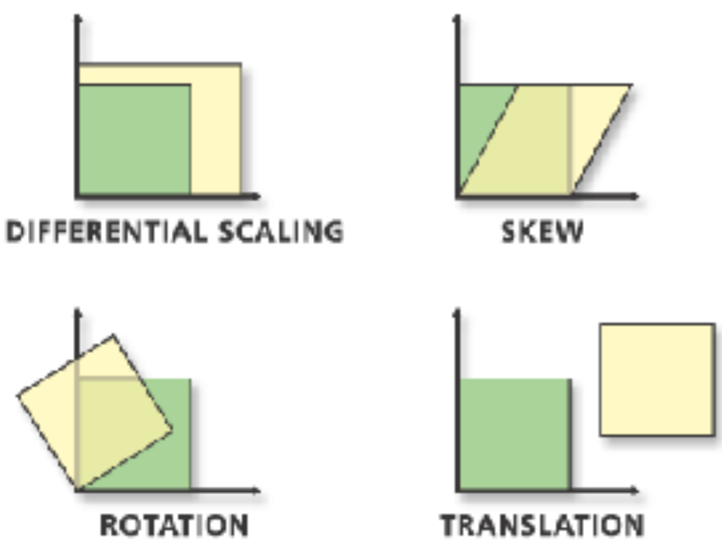
V1	Motion
V2	Stereo
V3	Color
V3a	Texture segregation
V3b	Segmentation, grouping
V4	Recognition
V7	Face recognition
MT	Attention
MST	Working memory/mental imagery

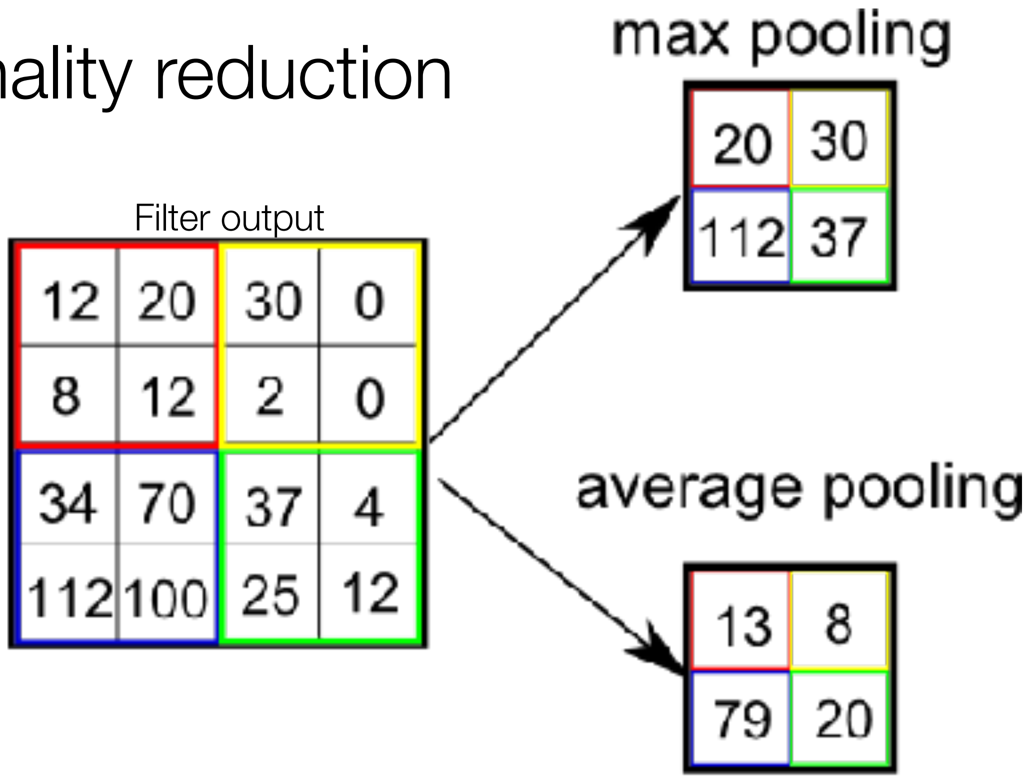
Convolution in a CNN



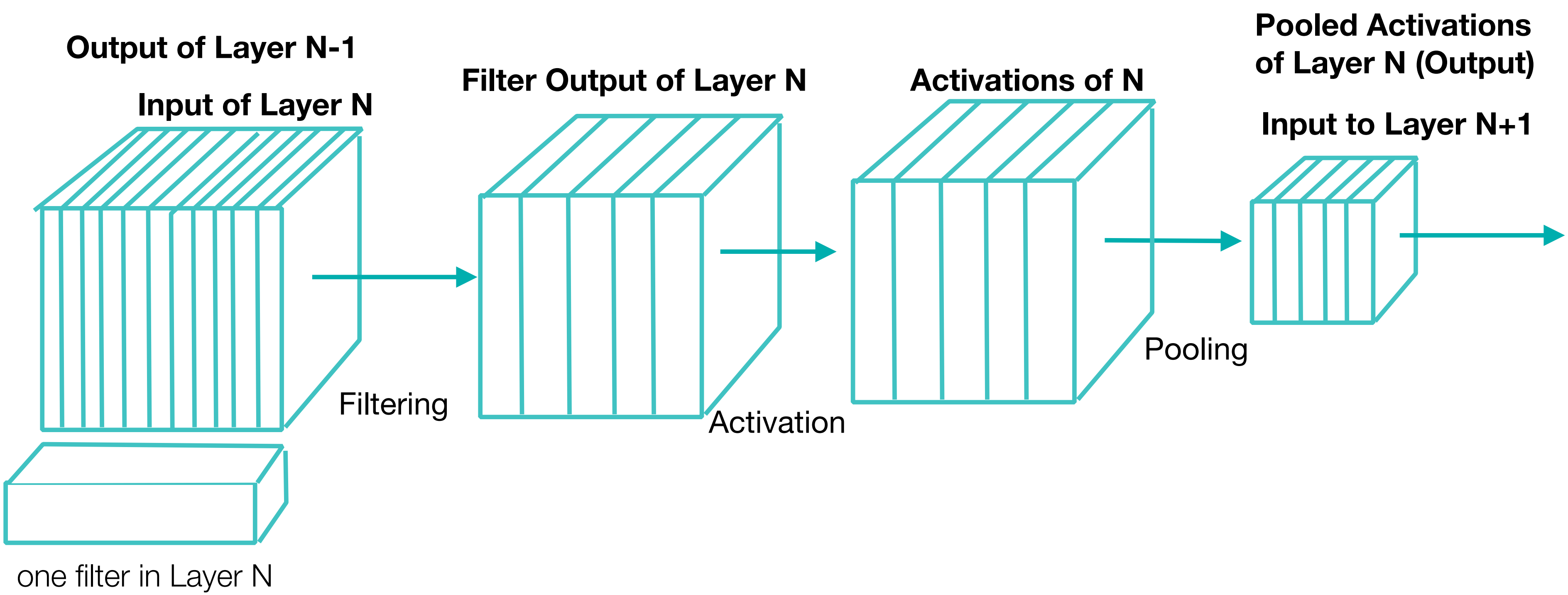
CNN Pooling

- 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