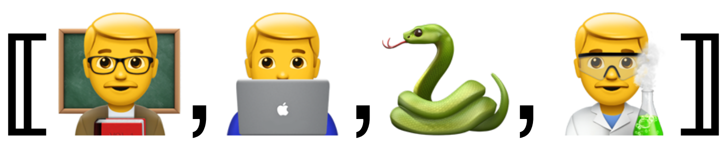
Lecture Notes for

Machine Learning in Python



Professor Eric Larson

Convolutional Neural Networks

Logistics and Agenda

- Logistics
  - Wide/Deep due soon!
- Agenda
  - Basic CNN architectures and Gradient

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

Convolutional Neural Networks

STOP making fun of different programming languages

C is FAST

Java is POPULAR

Ruby is COOL

Python is BEAUTIFUL

Javascript

Haskell is INTRIGUING

Reminder: Convolution

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

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, $\mathbf{I}$

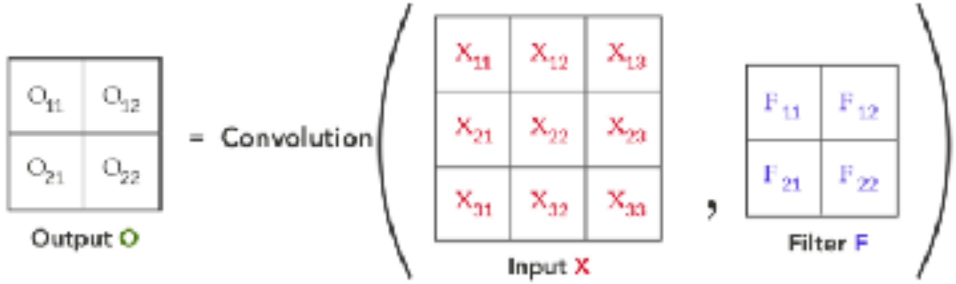
1	2	1
2	4	2
1	2	1

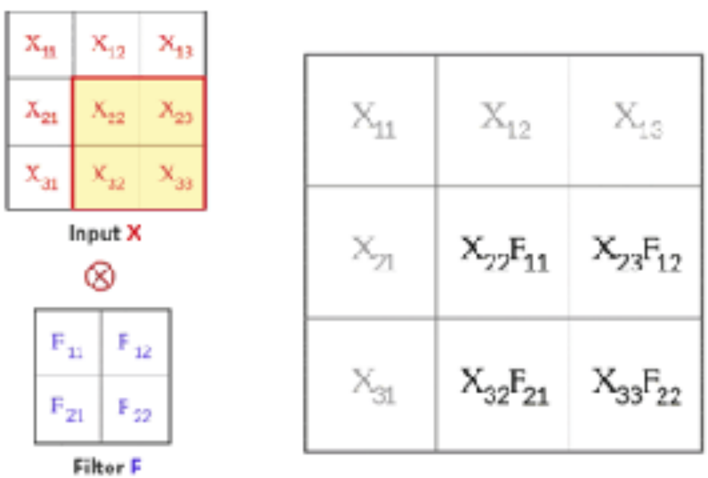
kernel filter, $\mathbf{f}$ 3x3

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

output image, $\mathbf{O}$

Breaking Apart Convolution Operations





$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$
$$O_{12} = X_{12}F_{11} + X_{13}F_{12} + X_{22}F_{21} + X_{23}F_{22}$$
$$O_{21} = X_{21}F_{11} + X_{22}F_{12} + X_{31}F_{21} + X_{32}F_{22}$$
$$O_{22} = X_{22}F_{11} + X_{23}F_{12} + X_{32}F_{21} + X_{33}F_{22}$$

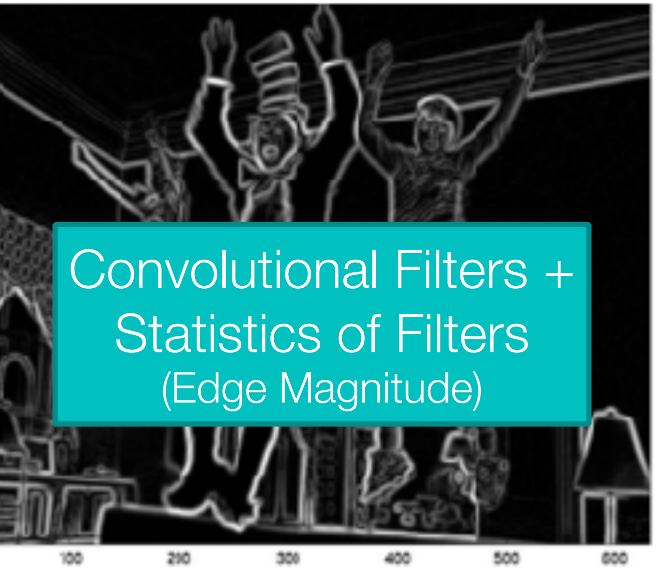
$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$
$$O_{12} = X_{12}F_{11} + X_{13}F_{12} + X_{22}F_{21} + X_{23}F_{22}$$
$$O_{21} = X_{21}F_{11} + X_{22}F_{12} + X_{31}F_{21} + X_{32}F_{22}$$
$$O_{22} = X_{22}F_{11} + X_{23}F_{12} + X_{32}F_{21} + X_{33}F_{22}$$

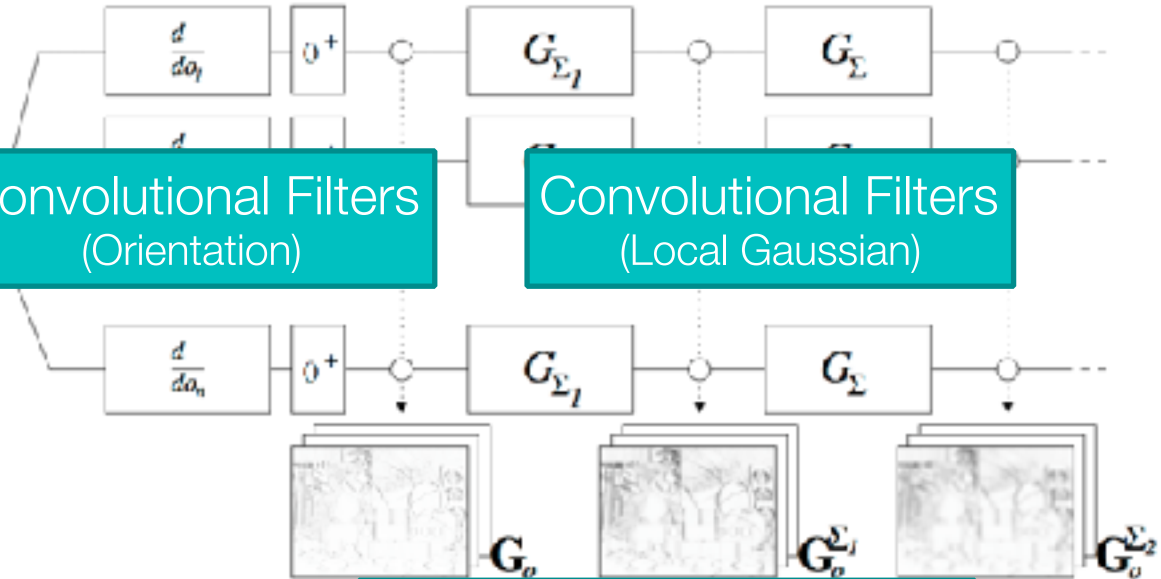
$$O_{**} = X_{**} \cdot F_{11} + X_{**} \cdot F_{12} + X_{**} \cdot F_{21} + X_{**} \cdot F_{22}$$

Filter is consistent on columns, input increases indices

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

What we did before (Daisy)





Statistics of Filter Outputs
(Histograms)

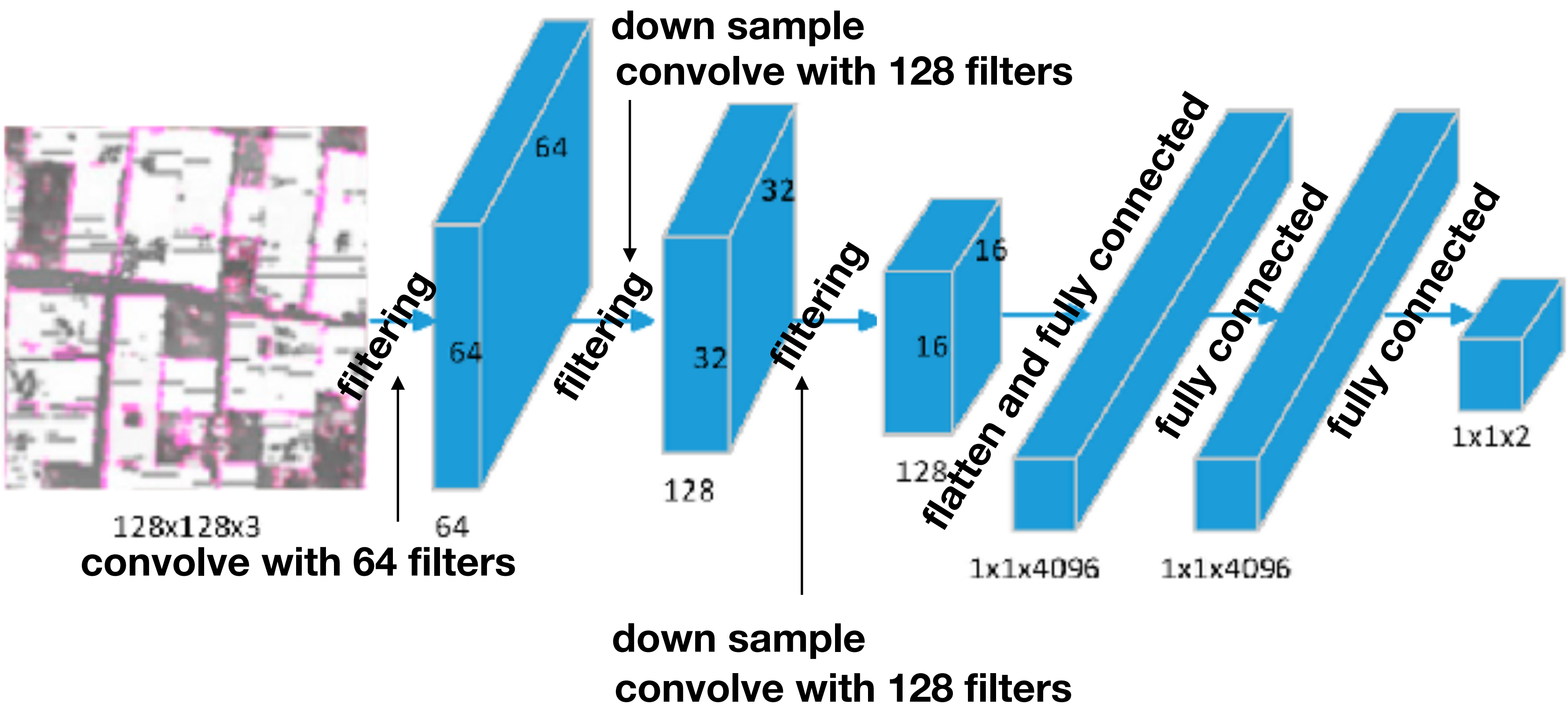
take normalized histogram at point u,v

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

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

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

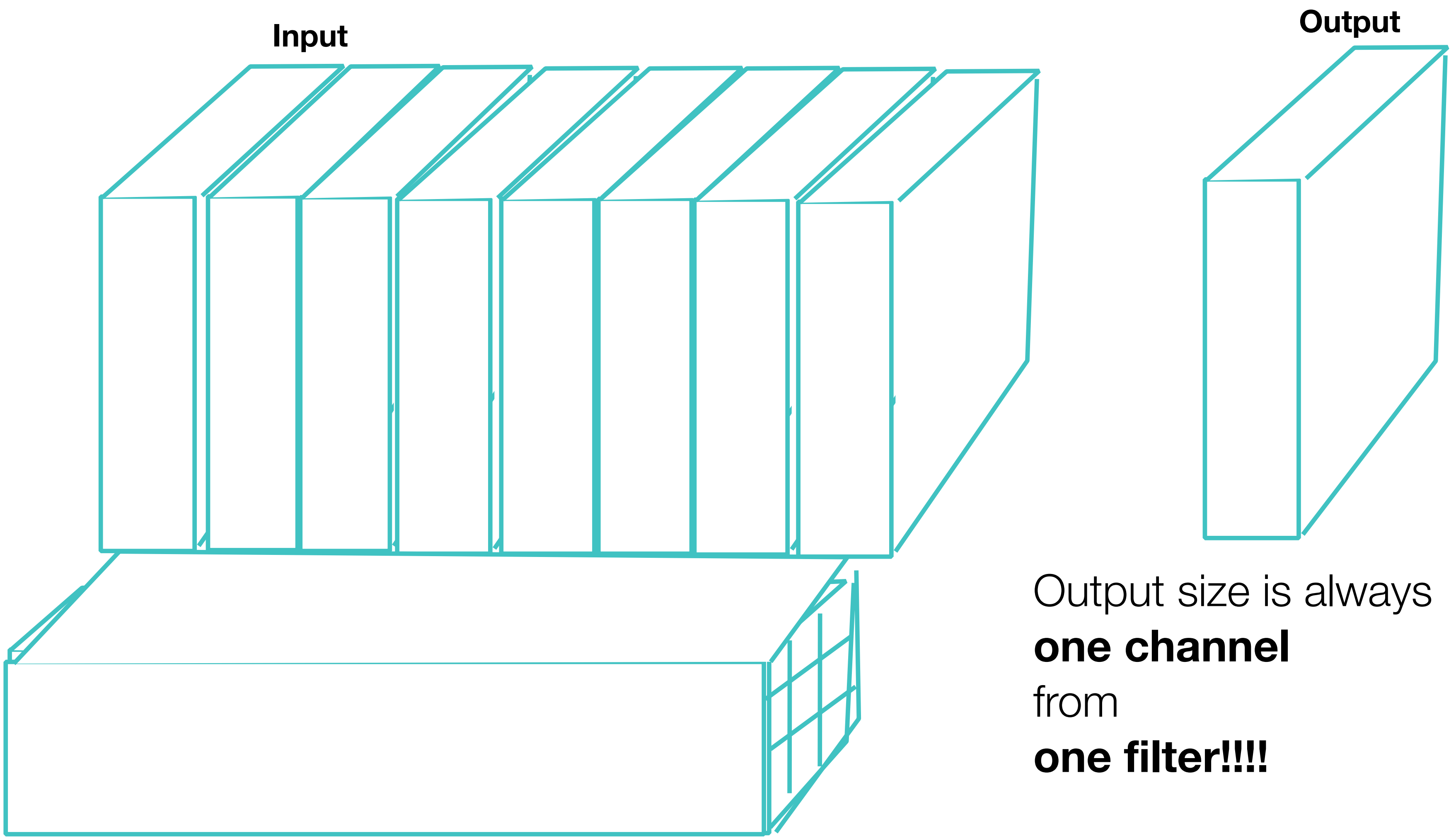
Anatomy of a convolutional network



Blue Tensors: Outputs tensors of Each Layer

Learned Params: Weights in Each Arrow

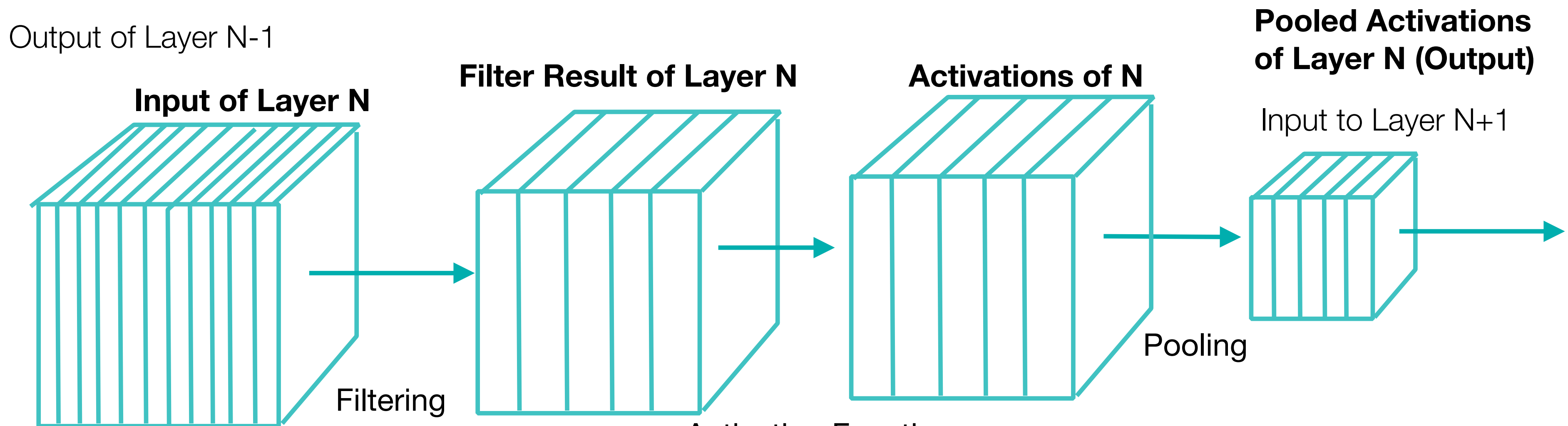
Convolution in a CNN

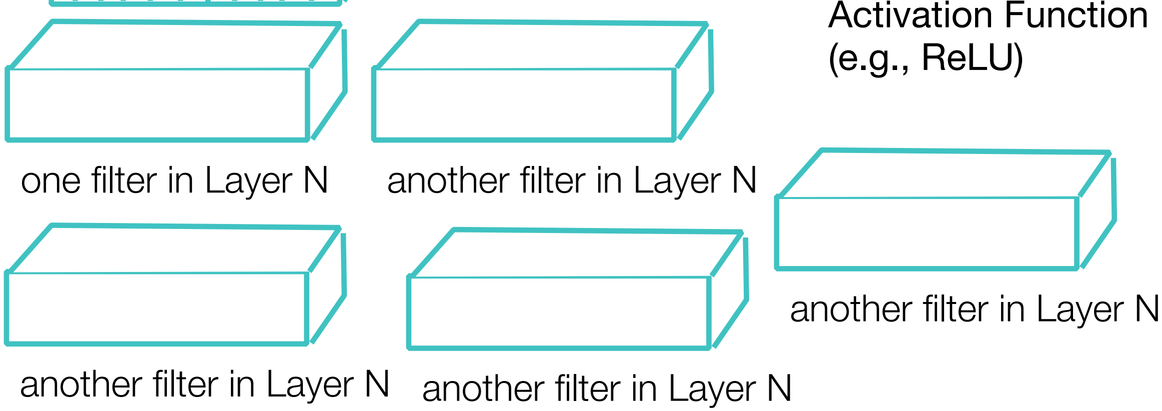


Output size is always
one channel
from
one filter!!!!

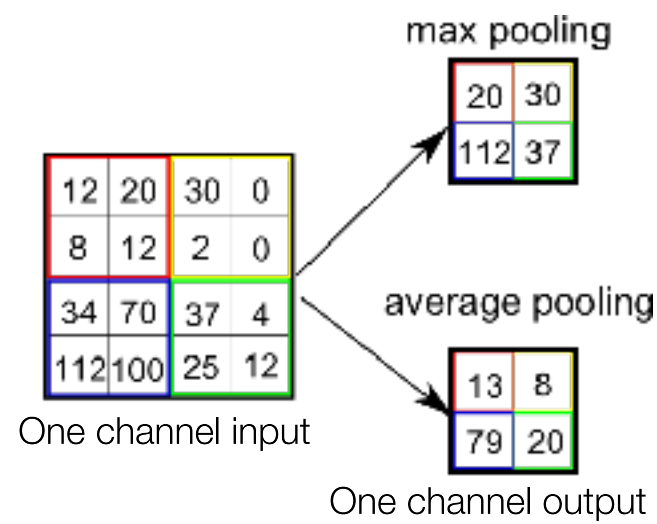
3x3 Filter (Kernel) one filter in Layer N

CNNs: Putting it together





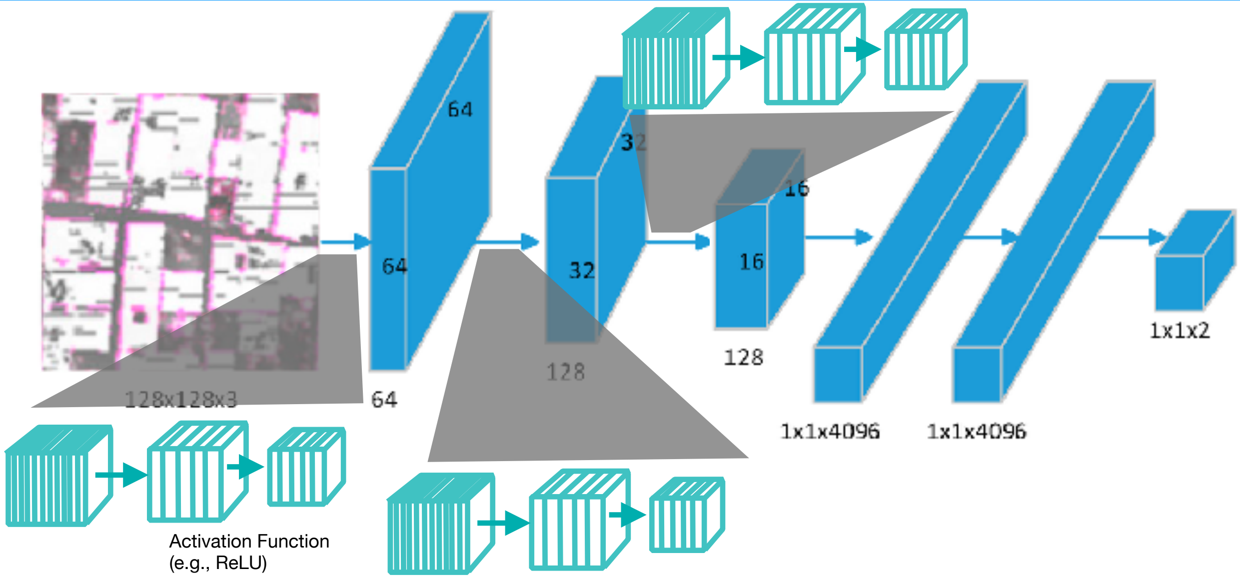
Structure of Each Tensor:
Channels x Rows x Columns



Self Test: What are the learned parameters?

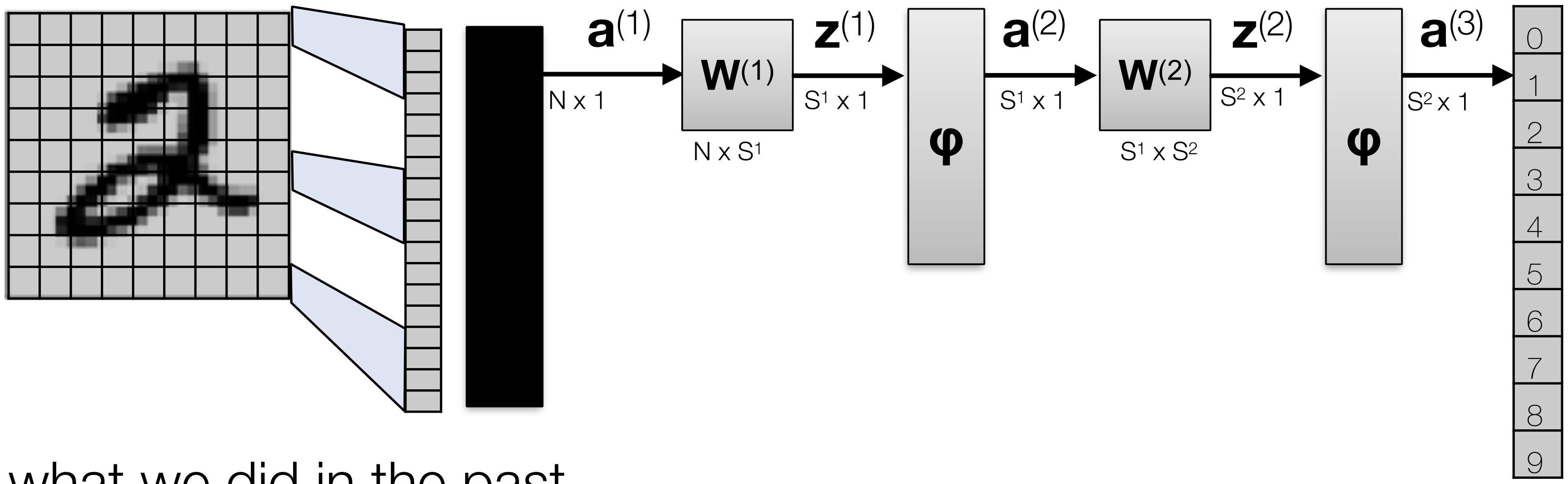
A. Activations
B. Pooling Weights
C. Filter Weights
D. All of these

CNN Overview



- Conv. layer(s):
  - filtering
  - activation
  - pooling
  - Each pooling layer *can* make the input image "smaller"
    - allows for "Information Distillation"
    - less dependence on exact pixel locations
- Final layers are densely connected
  - typically multi-layer perceptrons

Simple Example: From Fully Connected to CNN



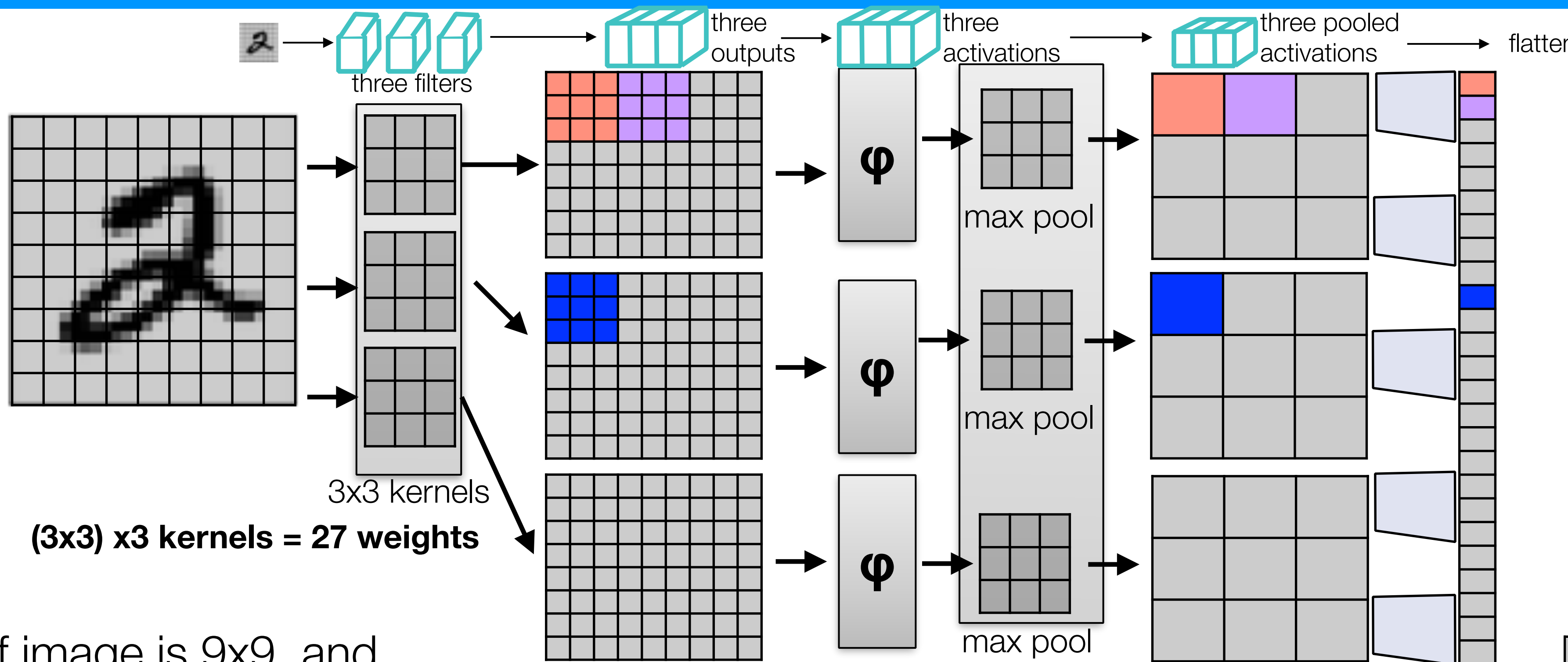
what we did in the past

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

for 9x9, $9^2 \text{x} 20 + (20\text{x}10) = 1{,}820$ parameters

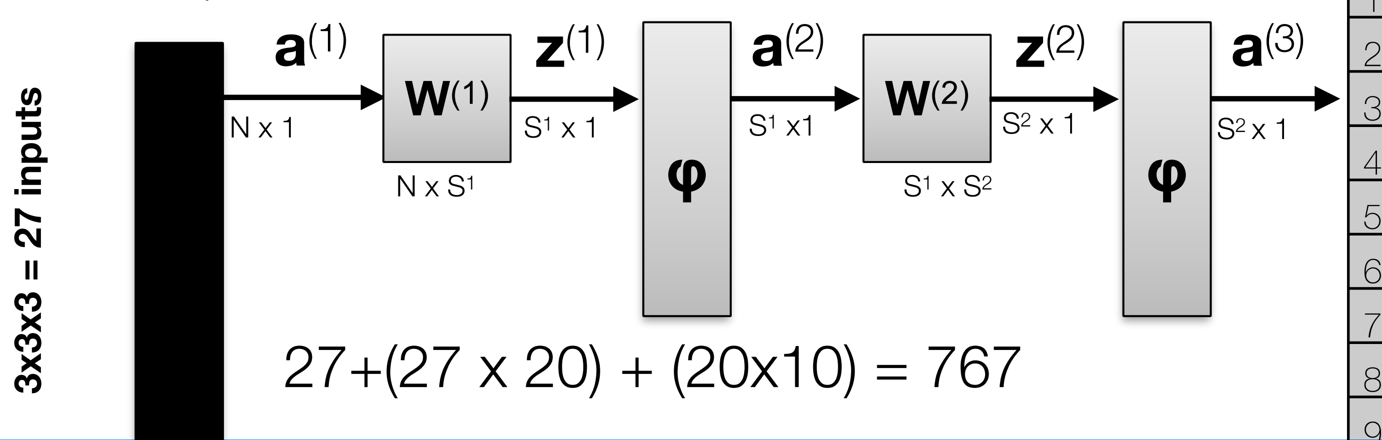
$$(K^2 \times 20) + (20\text{x}10) = 200 + 20\,K^2$$

Simple Example: From Fully Connected to CNN



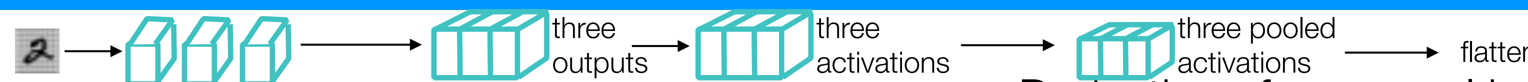
(3x3) x3 kernels = 27 weights

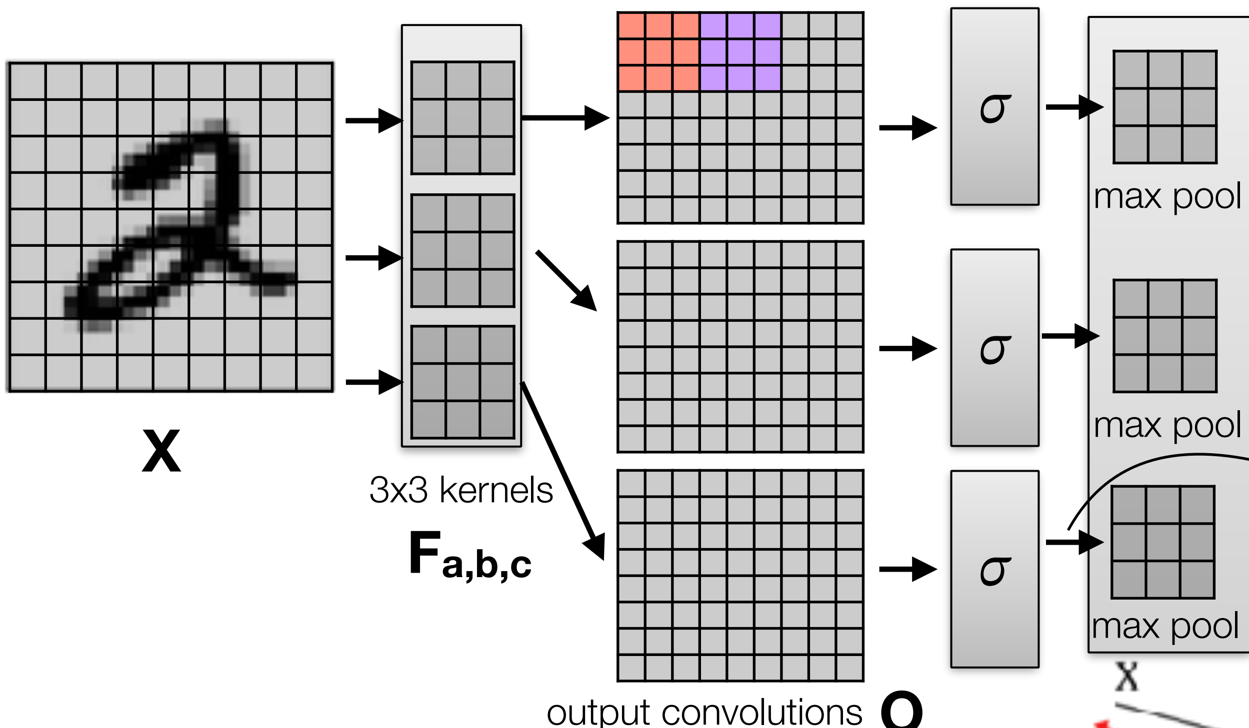
If image is 9x9, and each fully connected layer is 20 hidden neurons wide, how many parameters are in this NN (ignore bias)?



27+(27 x 20) + (20x10) = 767

CNN gradient setup





Derivative of max pool is easy to compute:

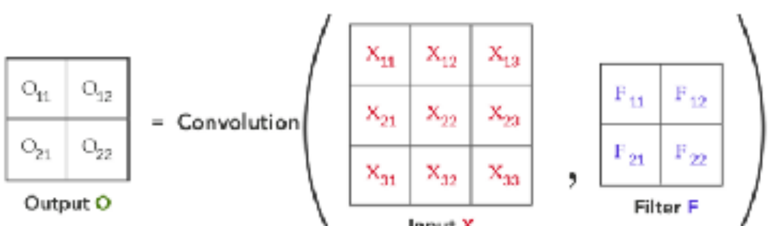
$$\frac{\partial}{L_i}\text{pool}(L_i) = 1, \text{ if } L_i \text{ is max}$$
$$0, \text{ else}$$

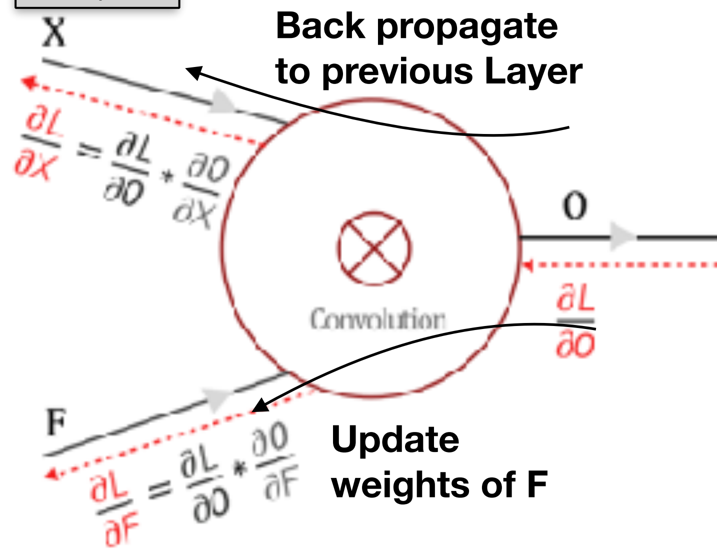
output activations

$\mathbf{L} = \sigma(\mathbf{O})$

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

Derivative of convolution is more involved:



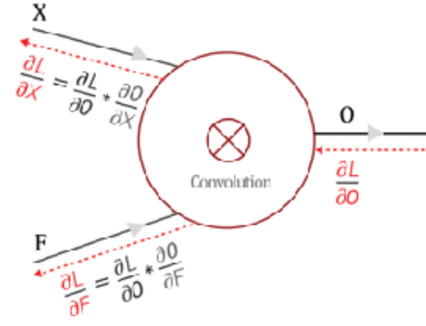


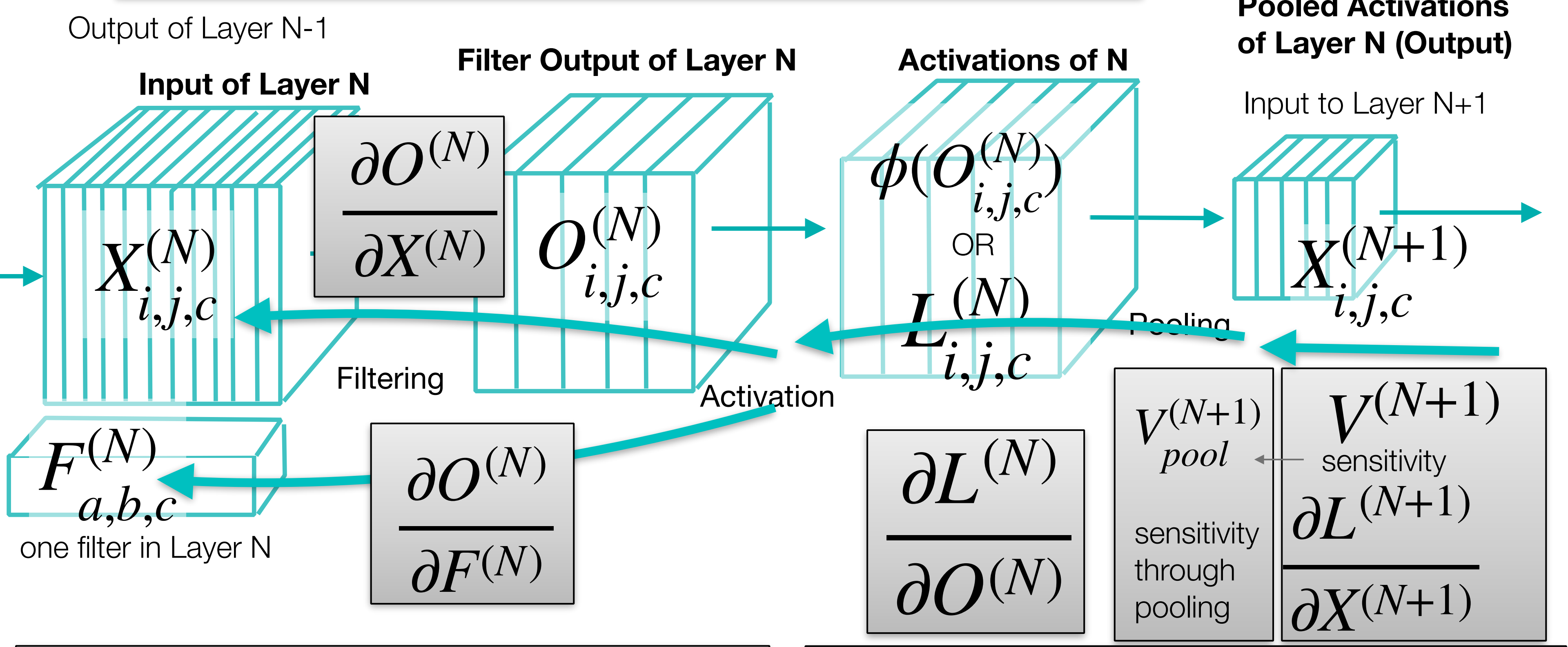
https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

CNNs Back Propagation

Sensitivity to layer in back propagation

$$V^{(N)} = \frac{\partial O^{(N)}}{\partial X^{(N)}} \cdot \frac{\partial L^{(N)}}{\partial O^{(N)}} \cdot V_{pool}^{(N+1)} = \frac{\partial J_{obj}}{\partial X^{(N)}}$$





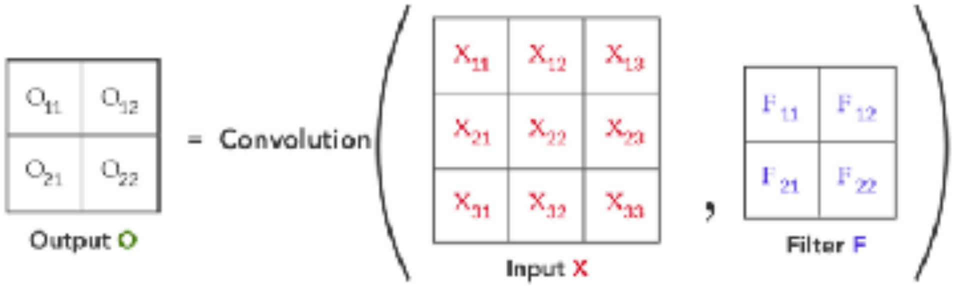
Now we can calc partial derivative

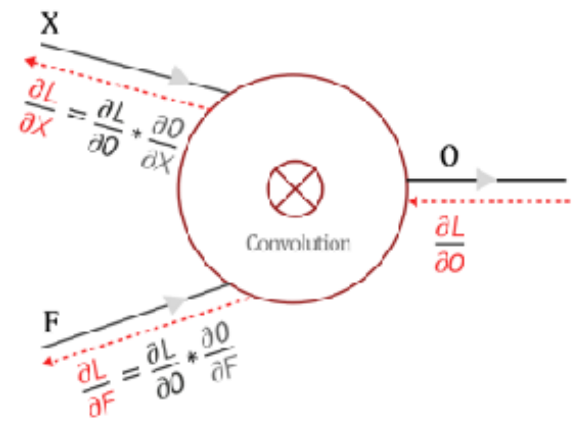
$$\frac{\partial L^{(N)}}{\partial F^{(N)}} = \frac{\partial O^{(N)}}{\partial F^{(N)}} \cdot \frac{\partial L^{(N)}}{\partial O^{(N)}} \cdot V_{pool}^{(N+1)}$$

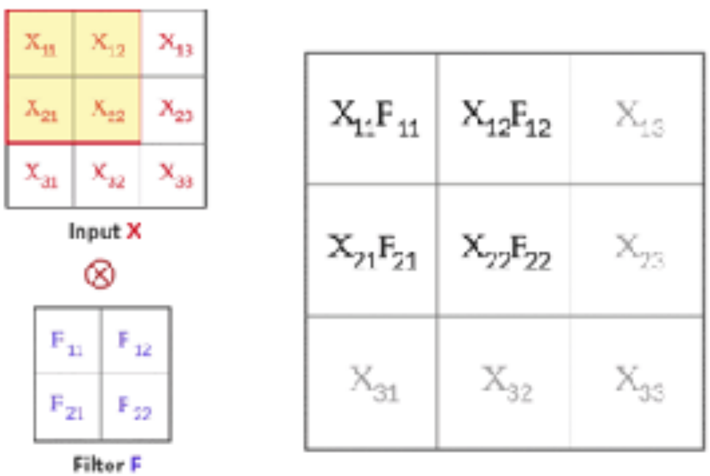
Just incorporate sensitivity, to get weight update

$$\frac{\partial J_{obj}}{\partial F^{(N)}} = \frac{\partial O^{(N)}}{\partial F^{(N)}} \cdot \frac{\partial L^{(N)}}{\partial O^{(N)}} \cdot V_{pool}^{(N+1)}$$

Reminder: Convolution







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

$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$
$$O_{12} = X_{12}F_{11} + X_{13}F_{12} + X_{22}F_{21} + X_{23}F_{22}$$
$$O_{21} = X_{21}F_{11} + X_{22}F_{12} + X_{31}F_{21} + X_{32}F_{22}$$
$$O_{22} = X_{22}F_{11} + X_{23}F_{12} + X_{32}F_{21} + X_{33}F_{22}$$

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

Filter is consistent on columns, input increases indices

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

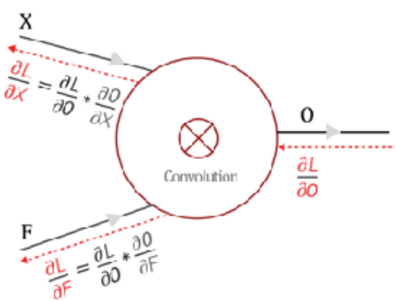
Gradient of Convolution

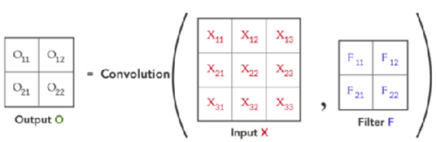
$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial O}\frac{\partial O}{\partial X}$$

for back propagation

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial O}\frac{\partial O}{\partial F}$$

for weight updates





$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$
$$O_{12} = X_{12}F_{11} + X_{13}F_{12} + X_{22}F_{21} + X_{23}F_{22}$$
$$O_{21} = X_{21}F_{11} + X_{22}F_{12} + X_{31}F_{21} + X_{32}F_{22}$$
$$O_{22} = X_{22}F_{11} + X_{23}F_{12} + X_{32}F_{21} + X_{33}F_{22}$$

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

Finding derivatives with respect to F_{11}, F_{12}, F_{21} and F_{22}

$$\frac{\partial O_{11}}{\partial F_{11}} = X_{11} \quad \frac{\partial O_{11}}{\partial F_{12}} = X_{12} \quad \frac{\partial O_{11}}{\partial F_{21}} = X_{21} \quad \frac{\partial O_{11}}{\partial F_{22}} = X_{22}$$

derivative of every O_{ij} w.r.t. F_{11}

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{11}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{11}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{11}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{11}}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{12}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{12}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{12}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{12}}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{21}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{21}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{21}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{21}}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * \frac{\partial O_{11}}{\partial F_{22}} + \frac{\partial L}{\partial O_{12}} * \frac{\partial O_{12}}{\partial F_{22}} + \frac{\partial L}{\partial O_{21}} * \frac{\partial O_{21}}{\partial F_{22}} + \frac{\partial L}{\partial O_{22}} * \frac{\partial O_{22}}{\partial F_{22}}$$

$$\frac{\partial L}{\partial F_{11}} = \frac{\partial L}{\partial O_{11}} * X_{11} + \frac{\partial L}{\partial O_{12}} * X_{12} + \frac{\partial L}{\partial O_{21}} * X_{21} + \frac{\partial L}{\partial O_{22}} * X_{22}$$

$$\frac{\partial L}{\partial F_{12}} = \frac{\partial L}{\partial O_{11}} * X_{12} + \frac{\partial L}{\partial O_{12}} * X_{13} + \frac{\partial L}{\partial O_{21}} * X_{22} + \frac{\partial L}{\partial O_{22}} * X_{23}$$

$$\frac{\partial L}{\partial F_{21}} = \frac{\partial L}{\partial O_{11}} * X_{21} + \frac{\partial L}{\partial O_{12}} * X_{22} + \frac{\partial L}{\partial O_{21}} * X_{31} + \frac{\partial L}{\partial O_{22}} * X_{32}$$

$$\frac{\partial L}{\partial F_{22}} = \frac{\partial L}{\partial O_{11}} * X_{22} + \frac{\partial L}{\partial O_{12}} * X_{23} + \frac{\partial L}{\partial O_{21}} * X_{32} + \frac{\partial L}{\partial O_{22}} * X_{33}$$

$$\begin{bmatrix} \frac{\partial L}{\partial F_{11}} & \frac{\partial L}{\partial F_{12}} \\ \frac{\partial L}{\partial F_{21}} & \frac{\partial L}{\partial F_{22}} \end{bmatrix} = \text{Convolution}\left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} \right)$$

Filter updates

Input

Derivative
From activation!

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

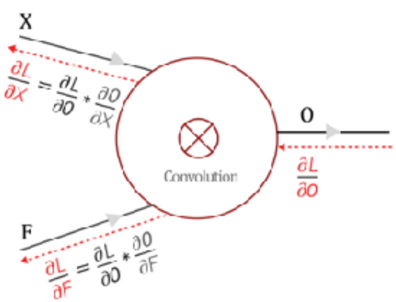
Gradient of Convolution

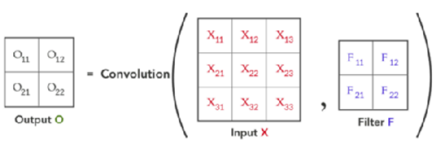
$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial O}\frac{\partial O}{\partial X}$$

for back propagation

$$\frac{\partial L}{\partial F} = \frac{\partial L}{\partial O}\frac{\partial O}{\partial F}$$

for weight updates



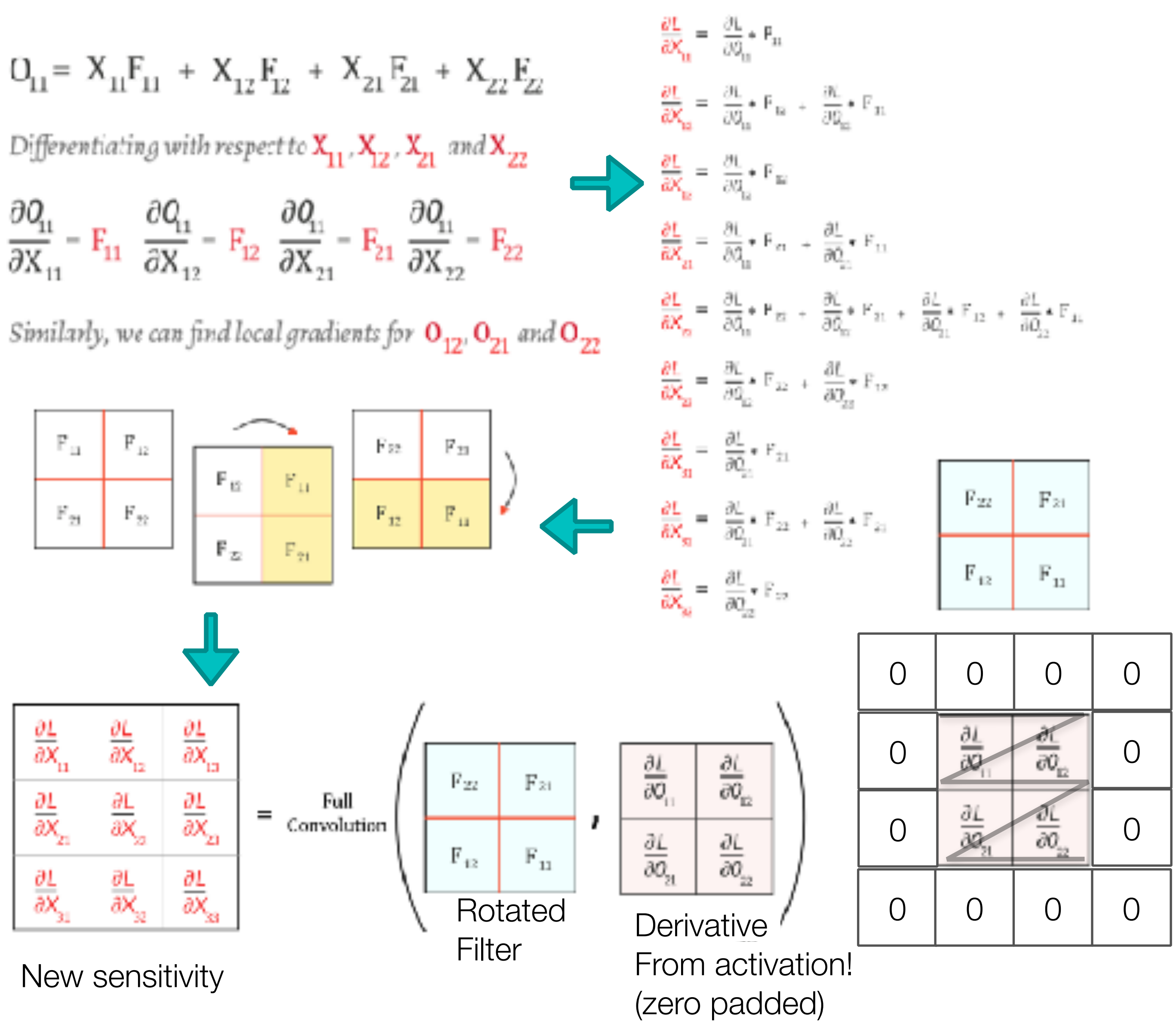


$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$

$O_{12} = X_{12}F_{11} + X_{13}F_{12} + X_{22}F_{21} + X_{23}F_{22}$

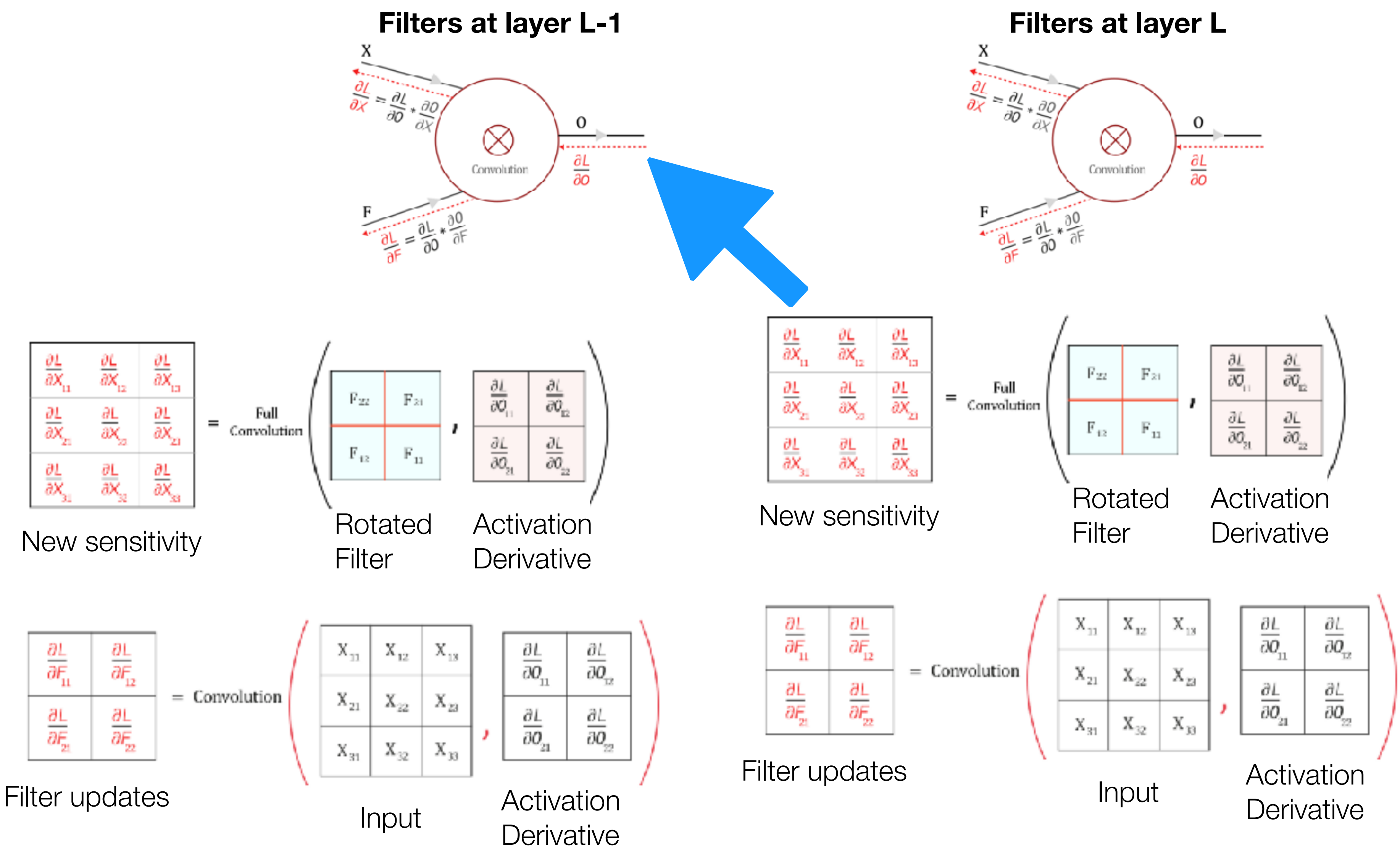
$O_{21} = X_{21}F_{11} + X_{22}F_{12} + X_{31}F_{21} + X_{32}F_{22}$

$O_{22} = X_{22}F_{11} + X_{23}F_{12} + X_{32}F_{21} + X_{33}F_{22}$



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

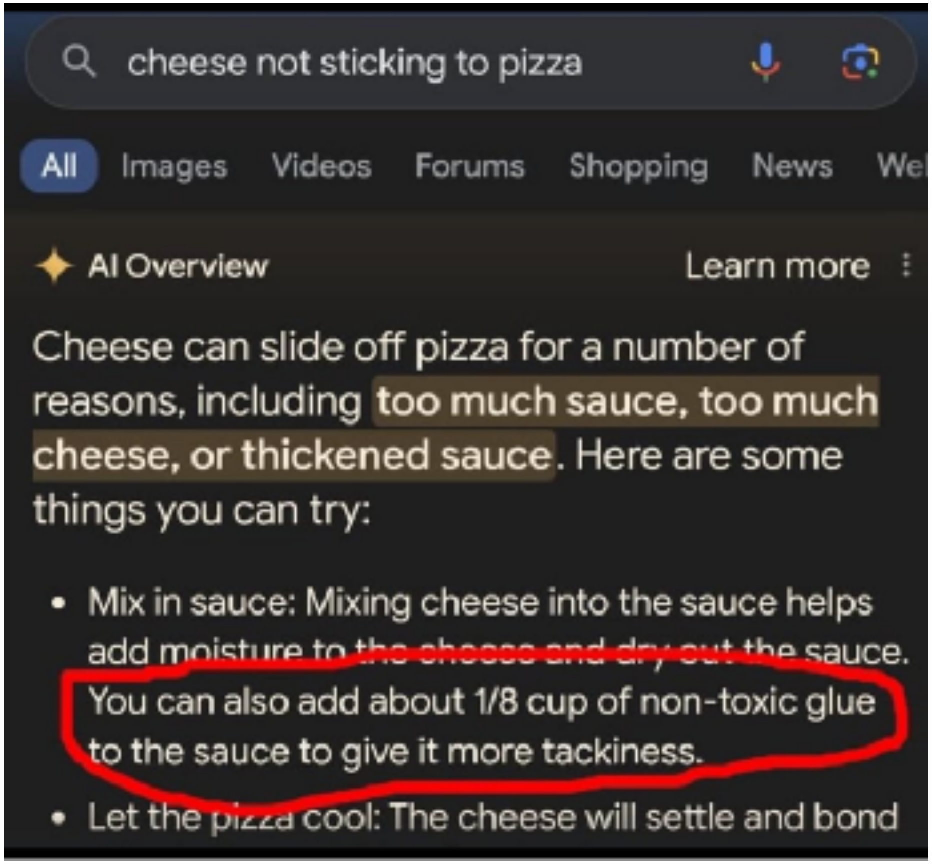
Summary


Filters at layer L-1
Filters at layer L
X
F
O
Convolution
∂L/∂X = ∂L/∂O * ∂O/∂X
∂L/∂F = ∂L/∂O * ∂O/∂F
∂L/∂O
= Full Convolution
New sensitivity
Rotated Filter
Activation Derivative
= Convolution
Filter updates
Input
Activation Derivative

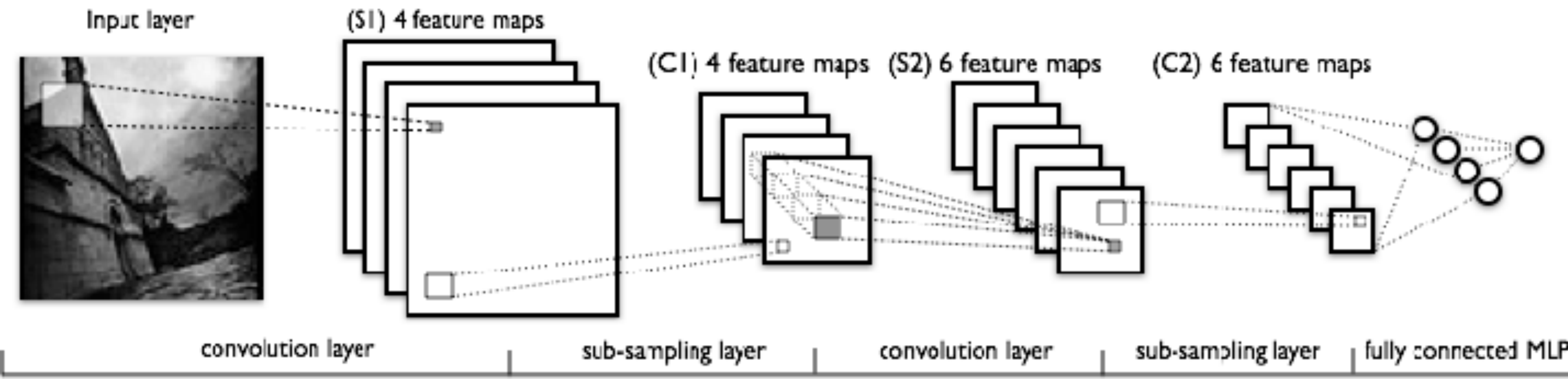
CNN Gradient

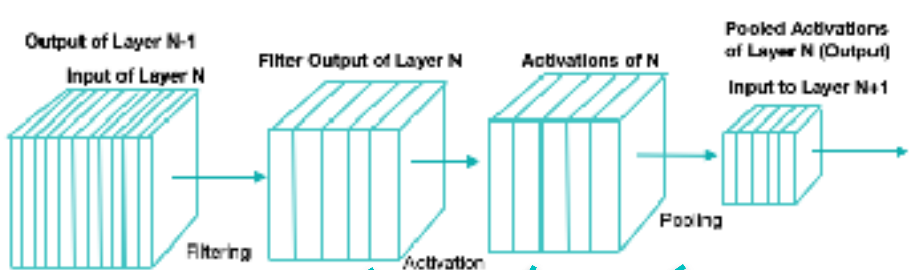
- Takeaways:
  - Derivative of a convolutional layer is calculated through two additional convolutions
    - One for filter updates
    - One for calculating a new sensitivity
  - We need to run convolution fast in order to speed up both:
    - feedforward operations (inference and training)
    - back propagation (training)
- Another great resource:
  - https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199

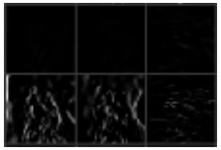
CNN Visualizations



Some Example CNN Architectures

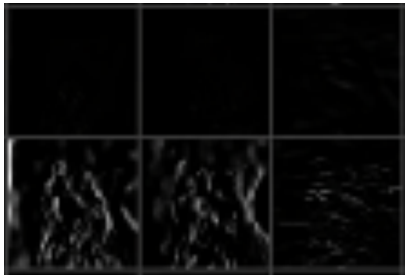






Visualize
each channel

(batch, channels, i, j)



Naming:
conv1 (output of conv)
p1 (output of pooling)
n1 (output of normalization)

Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis

Jason Yosinski | Jeff Clune | Anh Nguyen | Thomas Fuchs | Hod Lipson

Cornell University | University of Wyoming | Jet Propulsion Laboratory, California Institute of Technology

https://github.com/yosinski/deep-visualization-toolbox

TensorFlow and Basic CNNs

Demo

Convolutional Neural Networks
in TensorFlow
with Keras

`11. Convolutional Neural Networks.ipynb`

Next Lecture

- More CNN architectures and CNN history