Lecture Notes for

Machine Learning in Python

[👨‍🏫, 👨‍💻, 🐍, 👨‍🔬]

Professor Eric Larson

Basic Convolutional Neural Networks

Logistics and Agenda

- Logistics
  - Wide/Deep due soon!
  - Remember: late turn in…
- 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

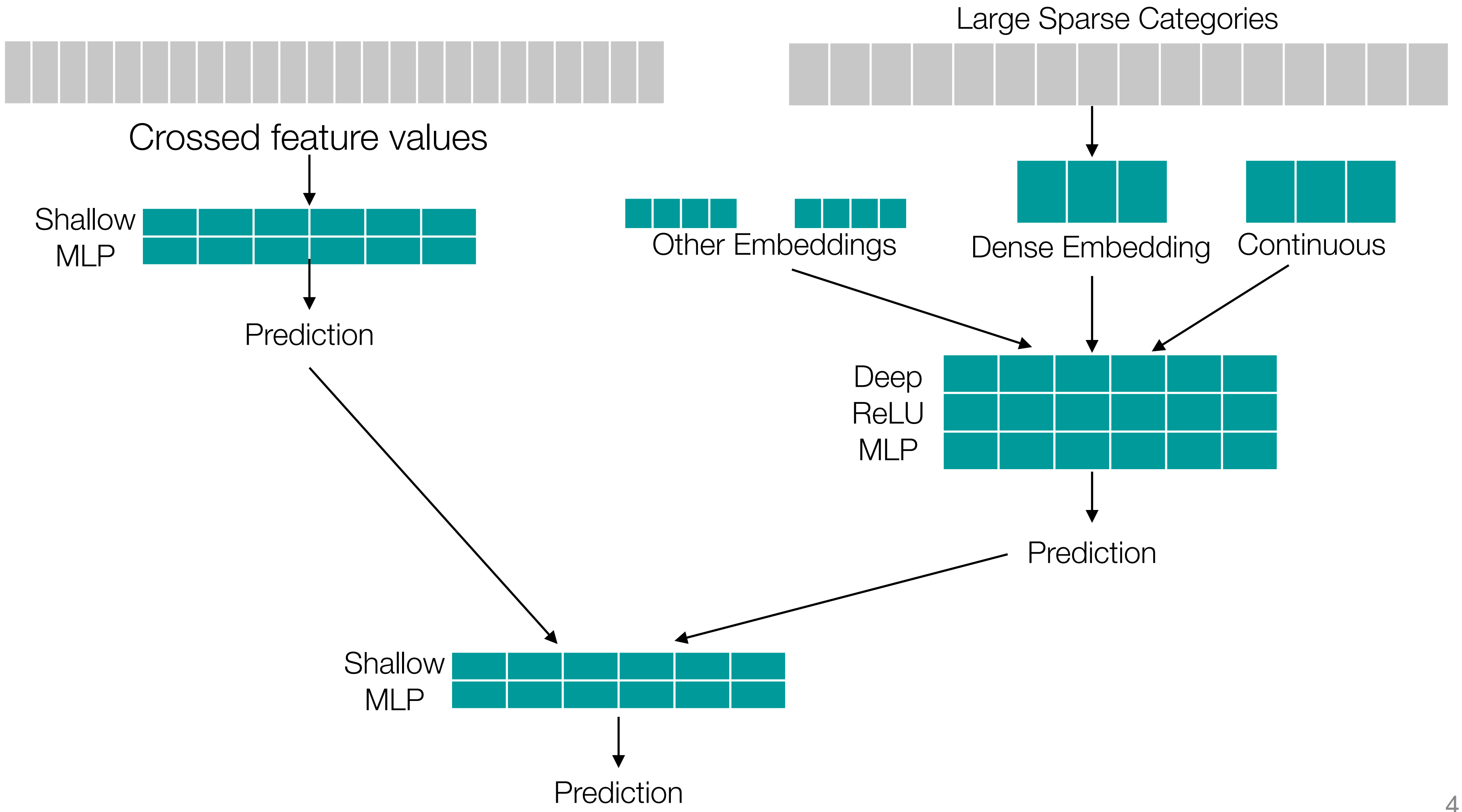
Wide and Deep Networks | **Convolutional Networks** | **Recurrent Networks**
Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models

ConceptNet
Case studies

Last Time:

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



Wide and Deep

“Finish” Demo

`10. Keras Wide and Deep.ipynb`

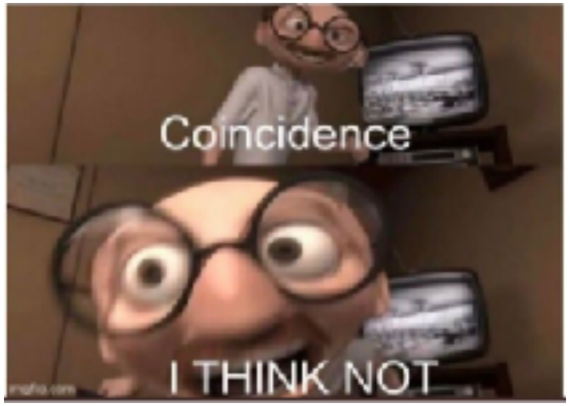
The awful dataset:
Toy Census Data Example

Other tutorials:

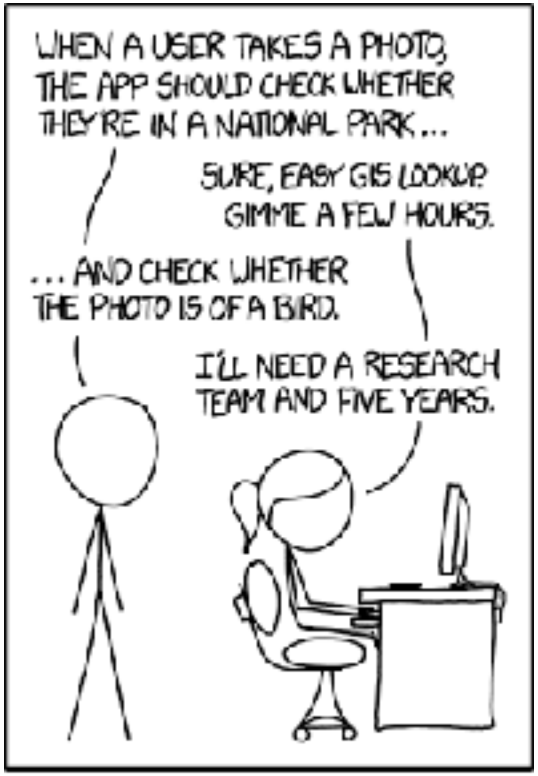
https://www.tensorflow.org/tutorials/wide_and_deep

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, $\mathbf{I}$

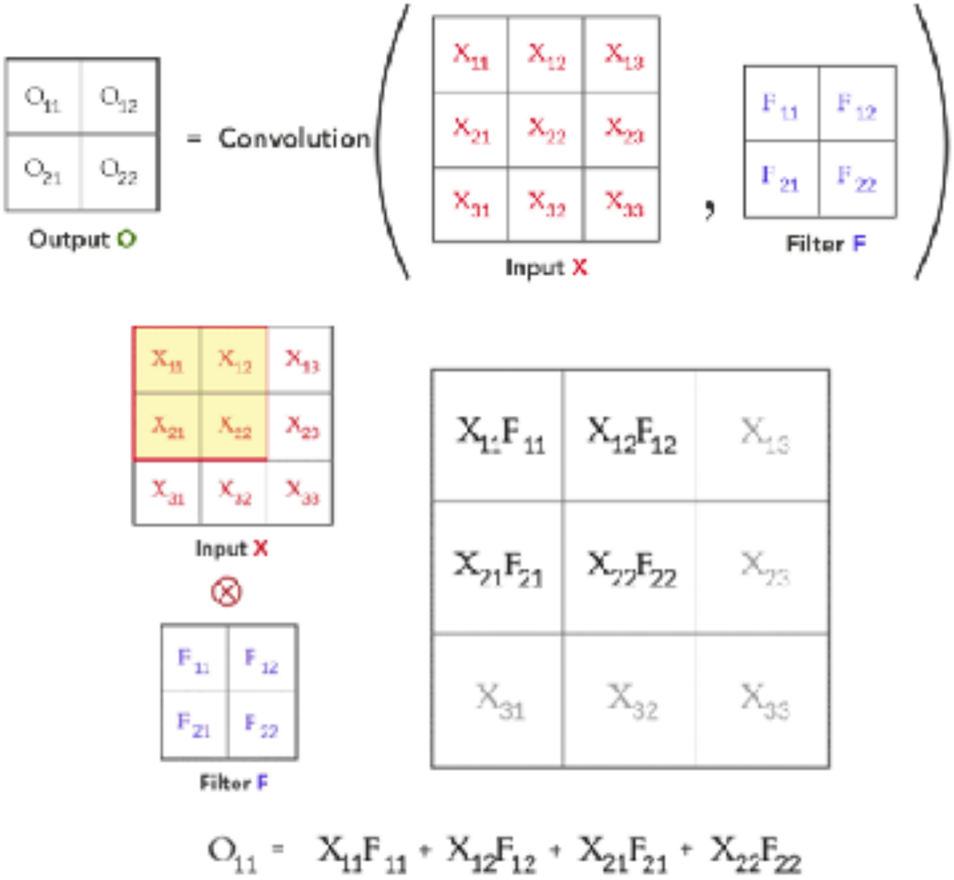
1	2	1
2	4	2
1	2	1

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

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

output image, $\mathbf{O}$

Reminder: Convolution



Observe: Can also express the convolution as matrix multiplication with a reshaped input and filter!

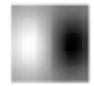
https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

What we did before

- the gradient (2D derivative)

I

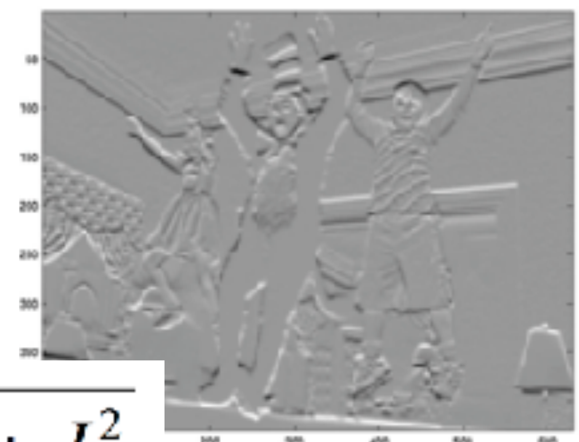
filter



filter

J_x

J_y

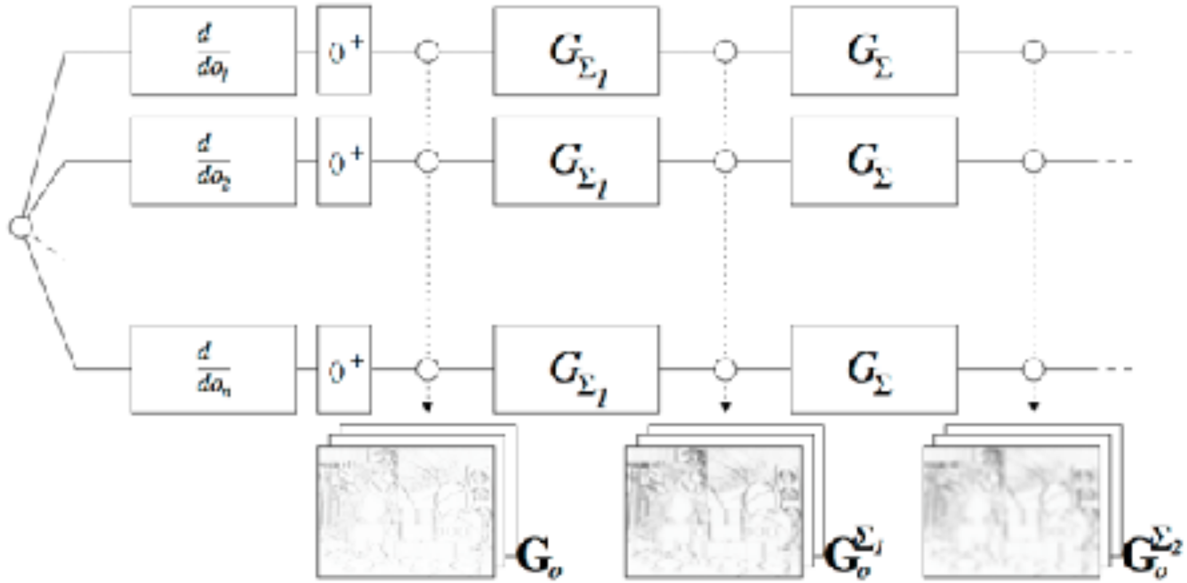


$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

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

images: Jianbo Shi, Upenn

What we did before



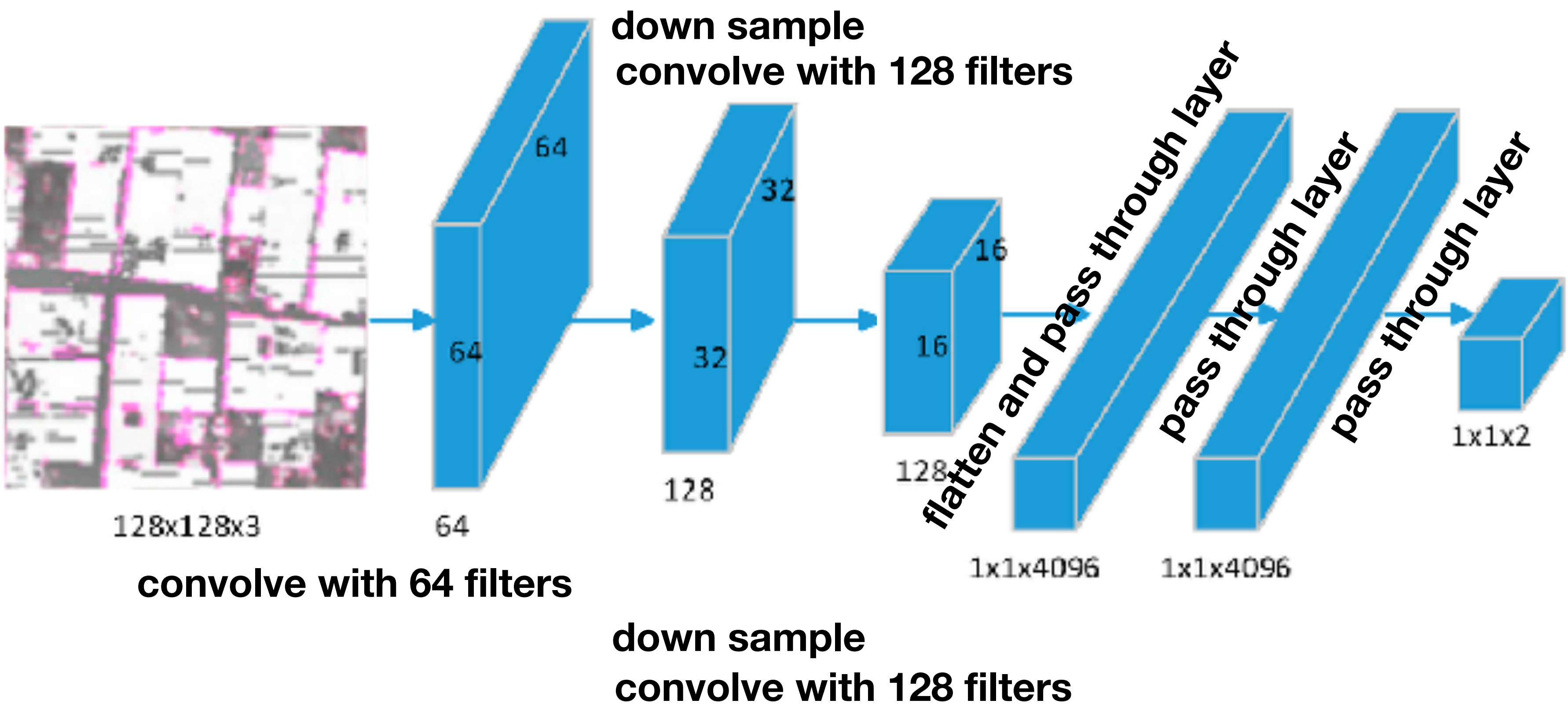
take normalized histogram at point u,v

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

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

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

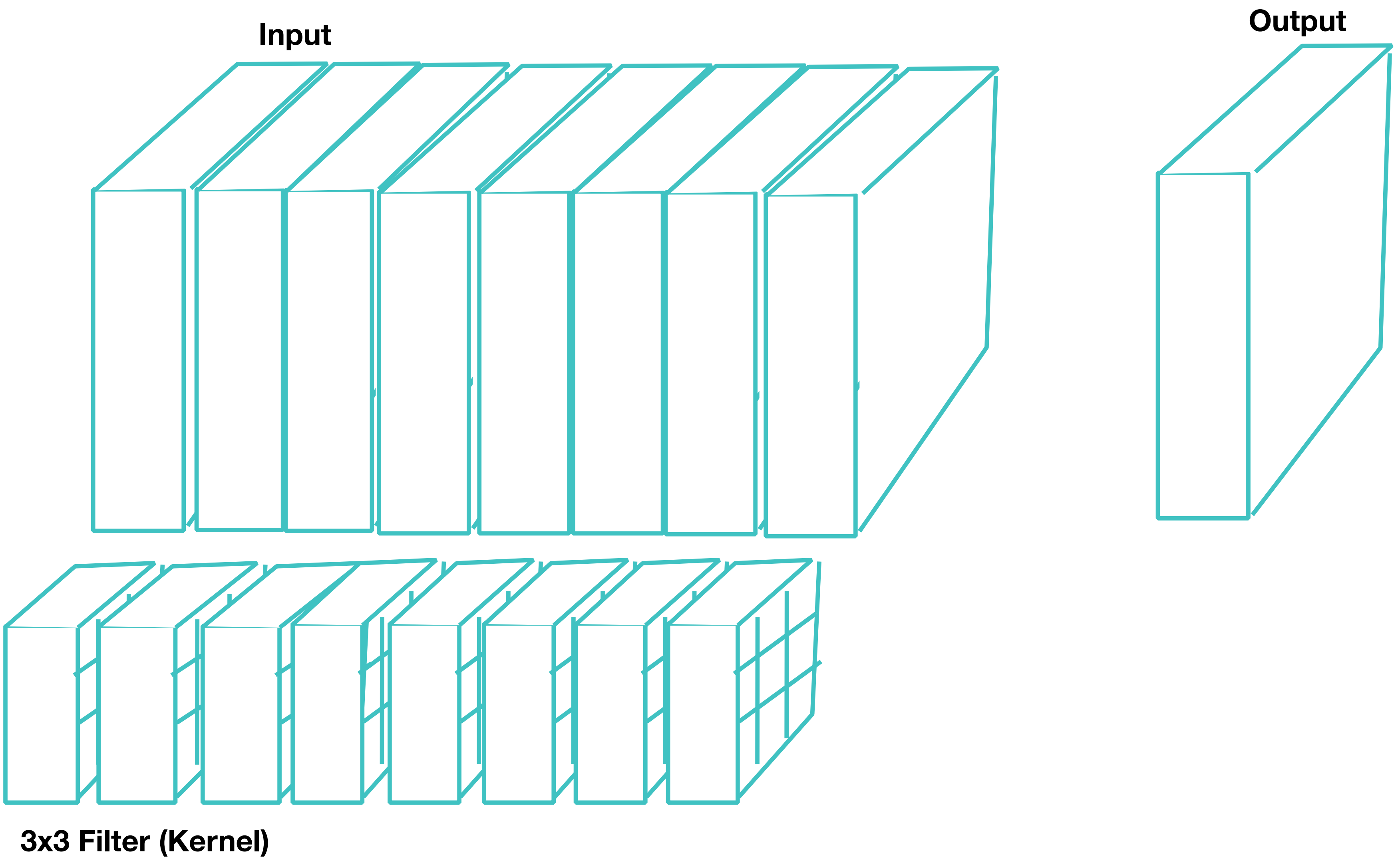
Anatomy of a convolutional network



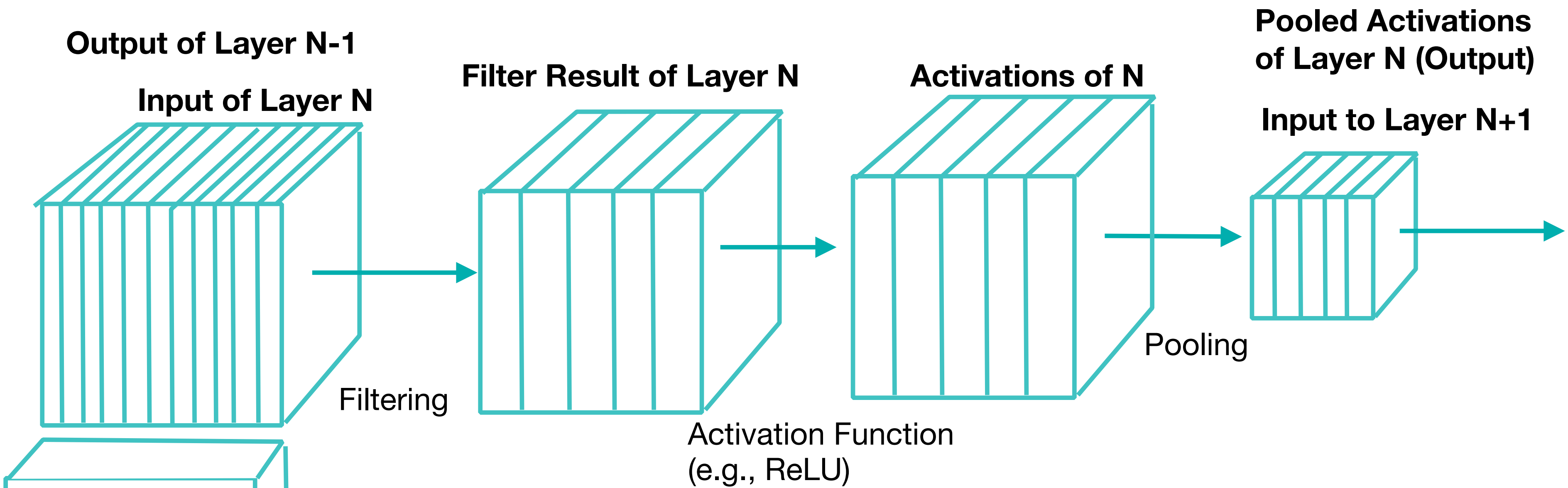
Blue Tensors: Outputs of Each Layer

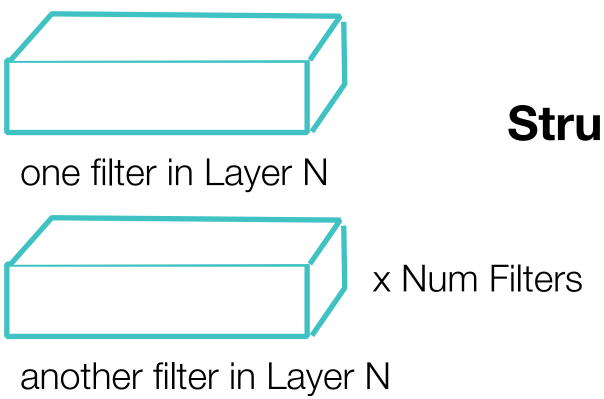
Learned Params: Weights in Each Filter and Fully Connected Layer

Convolution in a CNN



CNNs: Putting it together





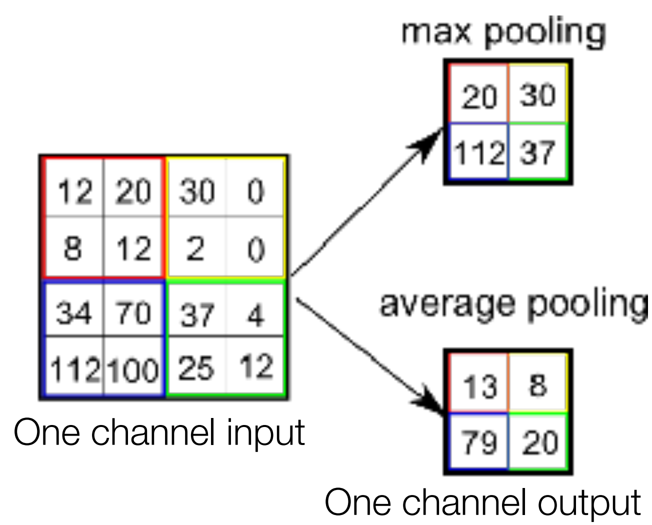
Structure of Each Tensor: Channels x Rows x Columns

What are the learned parameters?

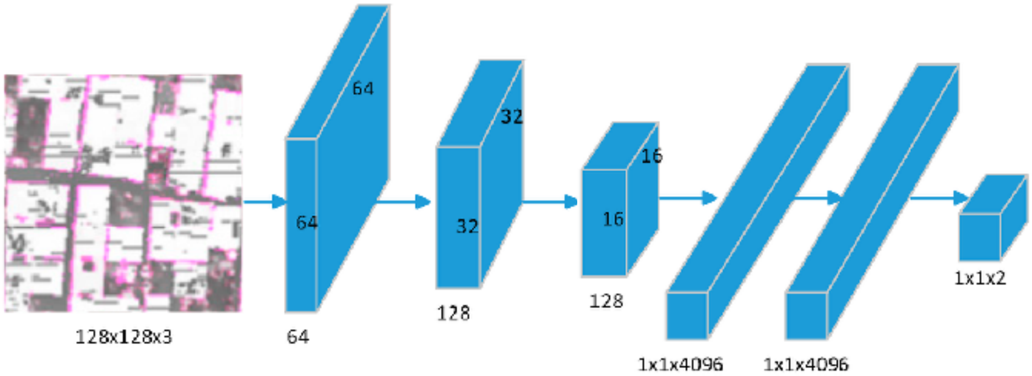
A. Activations

B. Pooling

C. Filters

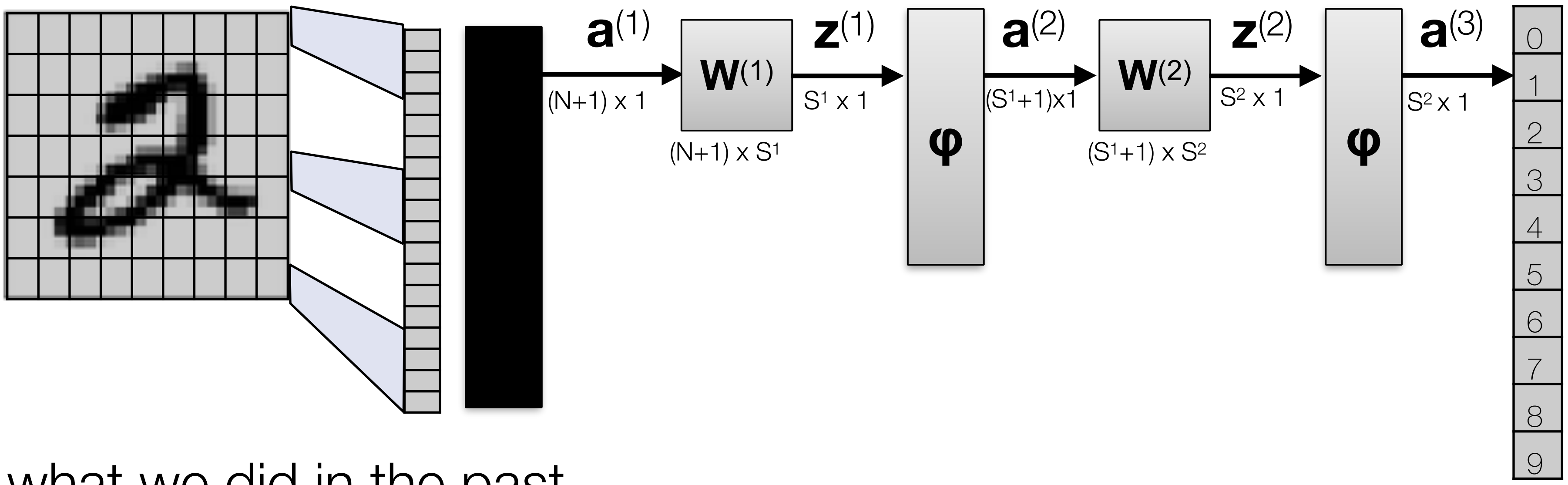


CNN Overview



- Initial 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

Simple Example: From Fully Connected to CNN



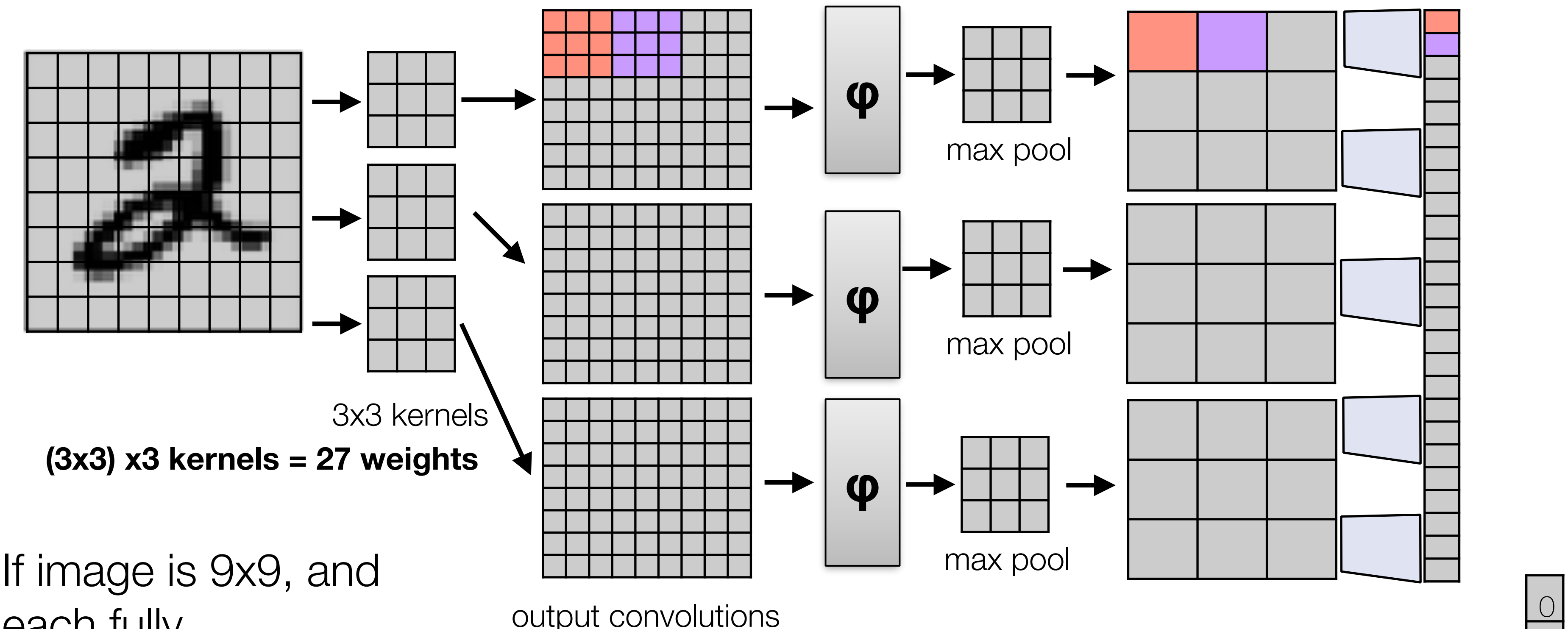
what we did in the past

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

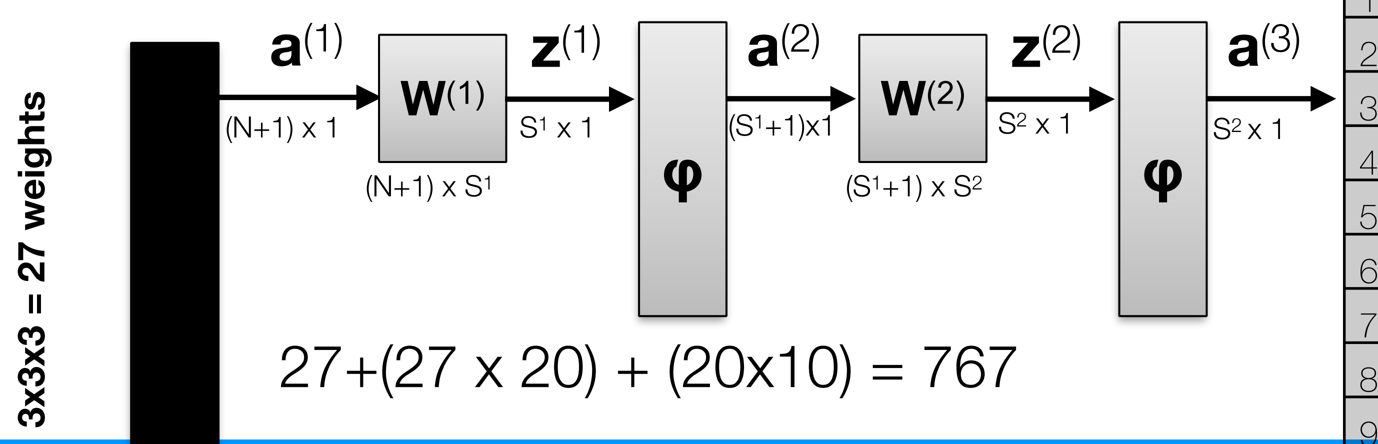
$(K^2 \times 20) + (20 \times 10) = 200 + 20\,K^2$

for 9x9 = $200 + 20 \times 9^2 =$ 1,820 parameters

Simple Example: From Fully Connected to CNN

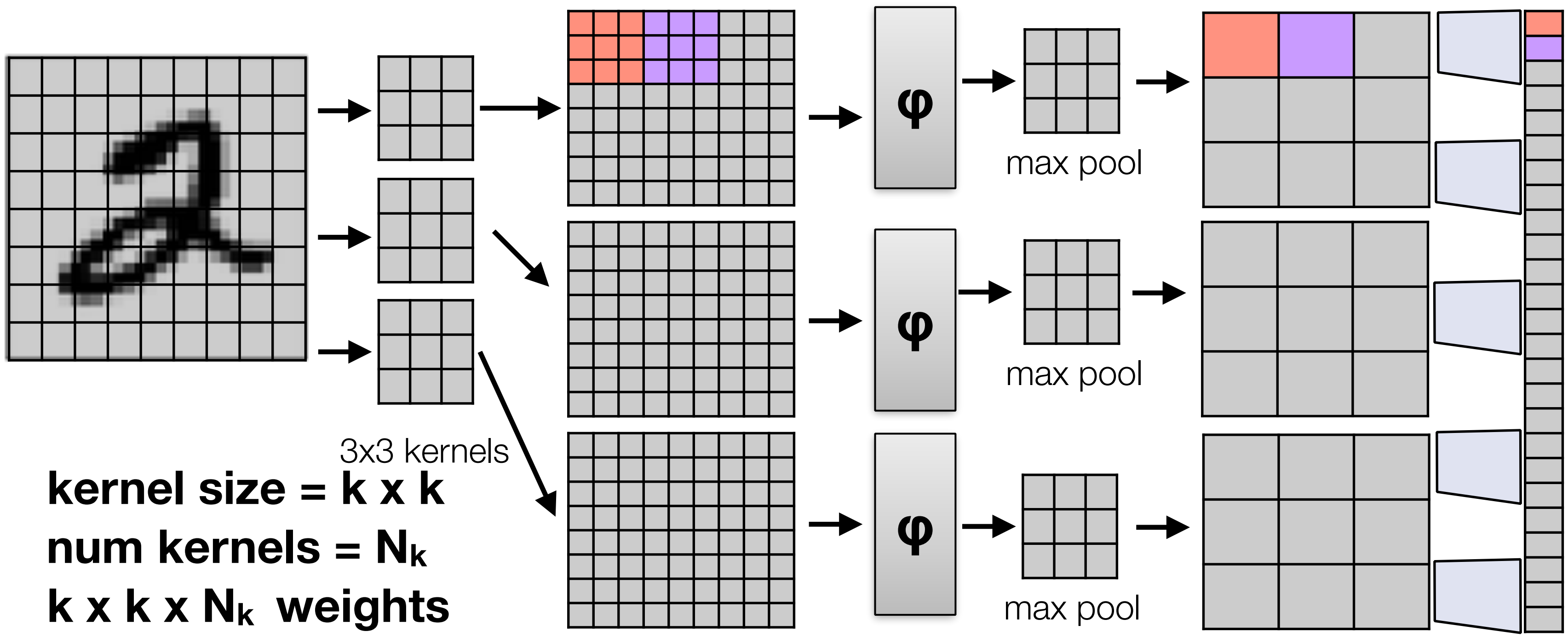


If image is 9x9, and each fully connected layer is 20 hidden neurons wide, how many parameters are in this NN (ignore bias)?



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

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

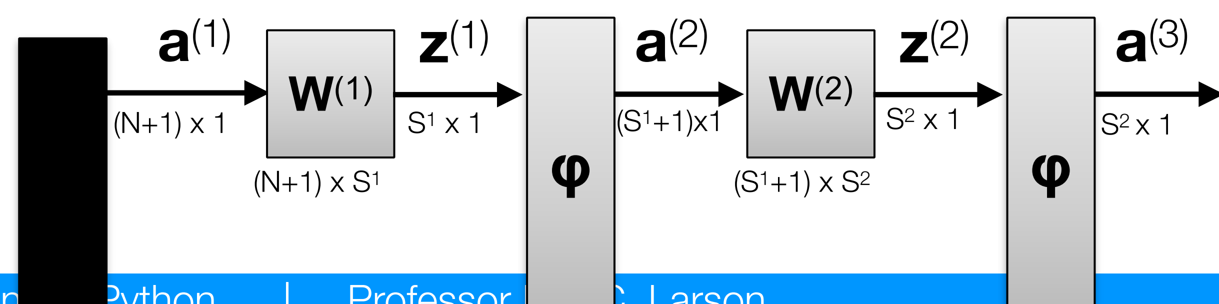
$N_k \times k^2$ ← filter dimension

num filters

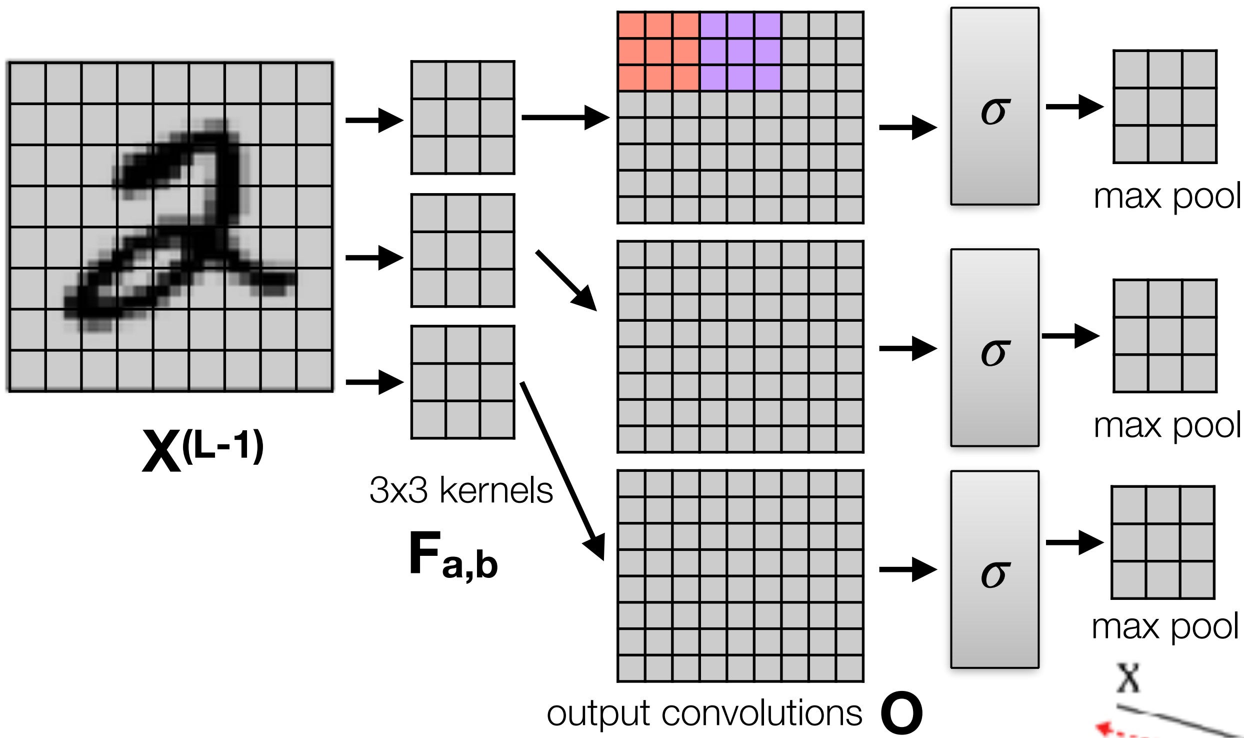
Input to MLP

$N_k \times (K^2/k^2)$

image dimension



CNN gradient



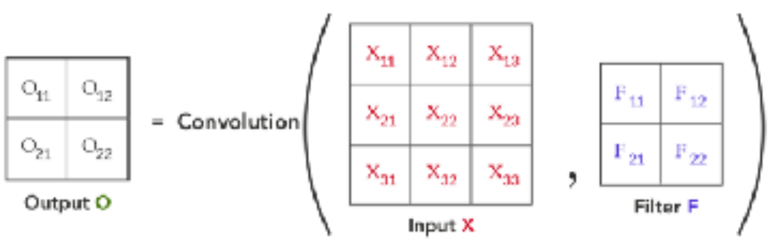
Derivative of max pool is easy:

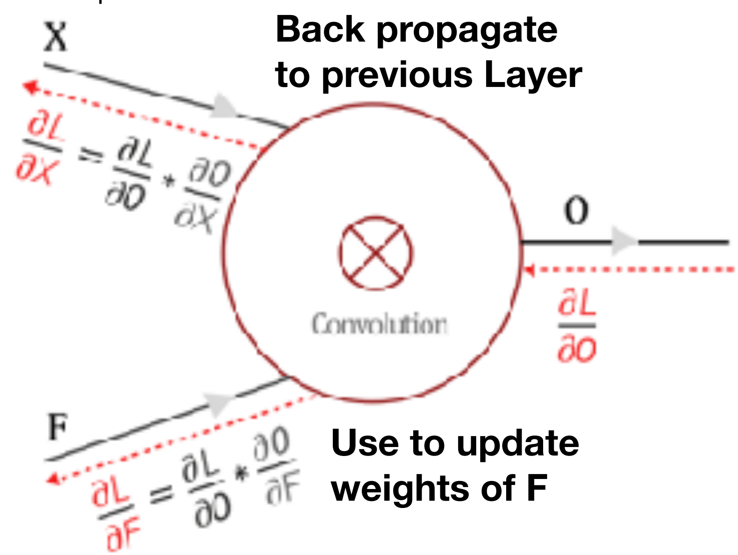
for each input X_i

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

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

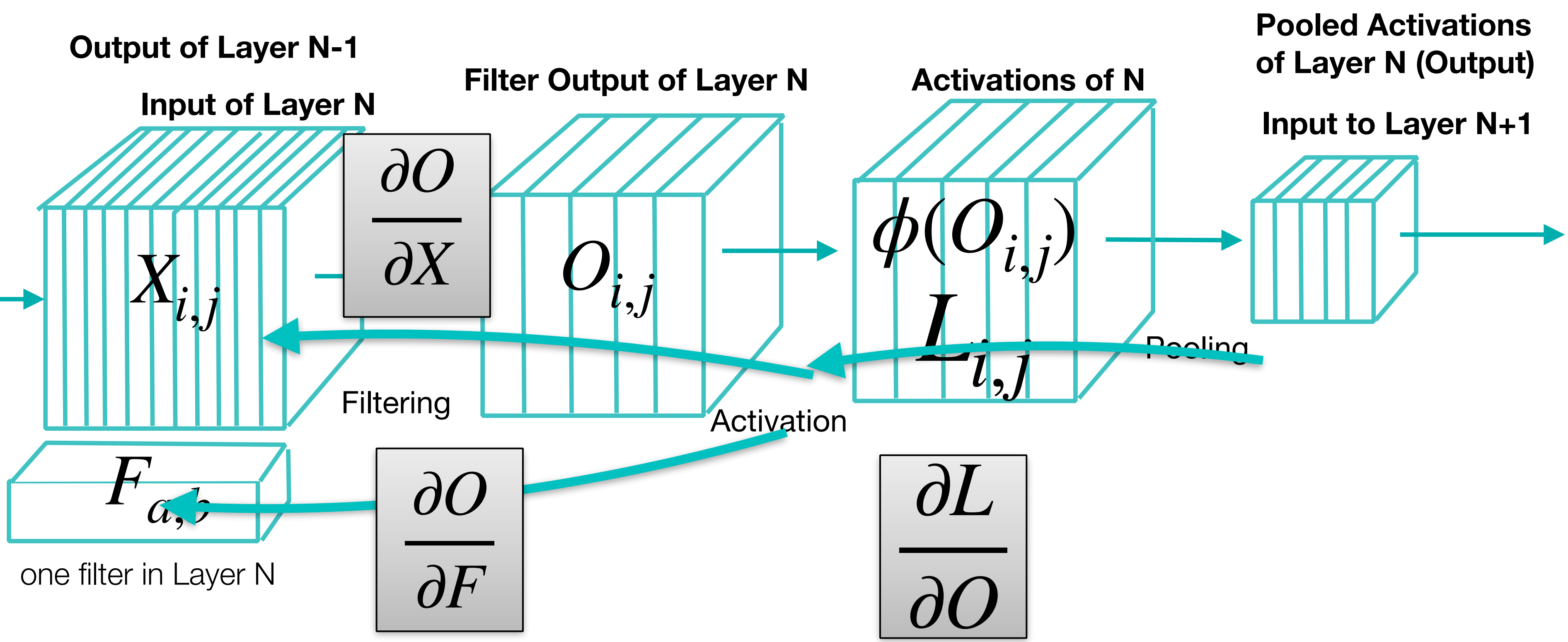
Derivative of convolution is more involved:





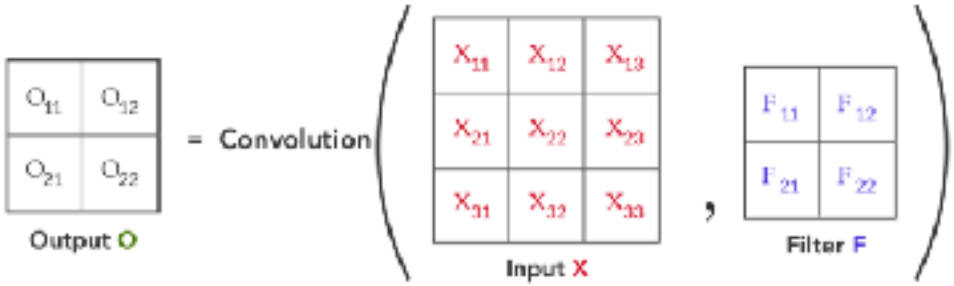
https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

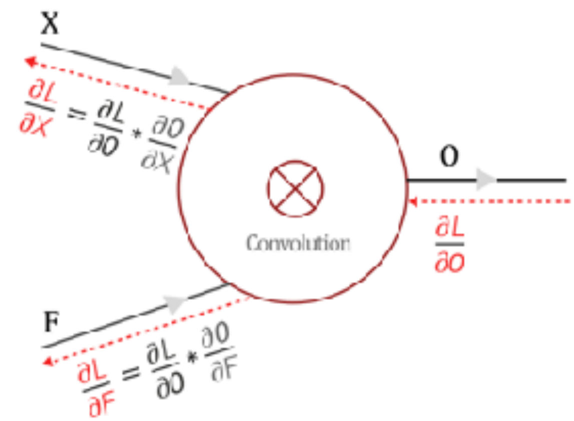
Last Time: CNNs, Putting it together

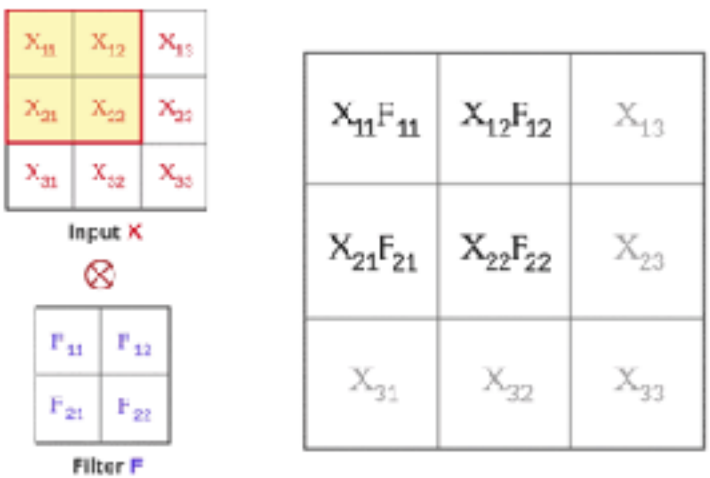


Structure of Each Tensor: Channels x Rows x Columns

Reminder: Convolution







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

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

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

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

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

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

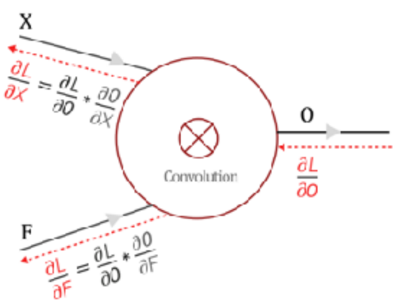
Gradient of Convolution

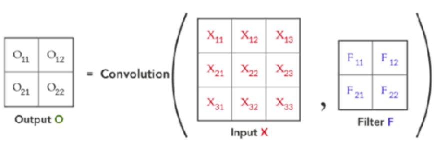
$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial O}\frac{\partial O}{\partial X}$$

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}

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

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

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

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

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

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

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

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

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

$$\underbrace{\begin{bmatrix} \frac{\partial L}{\partial F_{11}} & \frac{\partial L}{\partial F_{12}} \\ \frac{\partial L}{\partial F_{21}} & \frac{\partial L}{\partial F_{22}} \end{bmatrix}}_{\text{Filter updates}} = \text{Convolution}\left(\underbrace{\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}}_{\text{Input}}, \underbrace{\begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix}}_{\text{Sensitivity from next layer}} \right)$$

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

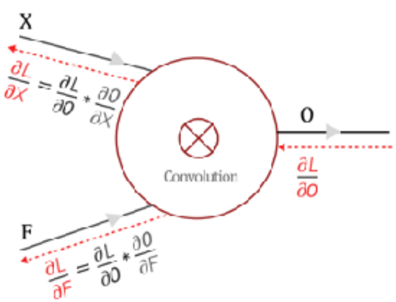
Gradient of Convolution

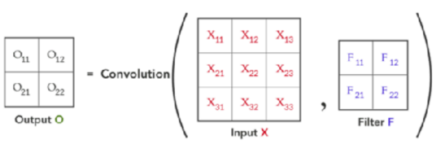
$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial O}\frac{\partial O}{\partial X}$$

for back propagation

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

for weight updates





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

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

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

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

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

Differentiating with respect to X_{11}, X_{12}, X_{21} *and* X_{22}

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

Similarly, we can find local gradients for O_{12}, O_{21} *and* O_{22}

$$\frac{\partial L}{\partial X_{11}} = \frac{\partial L}{\partial O_{11}} * F_{11}$$

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

$$\frac{\partial L}{\partial X_{13}} = \frac{\partial L}{\partial O_{12}} * F_{12}$$

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

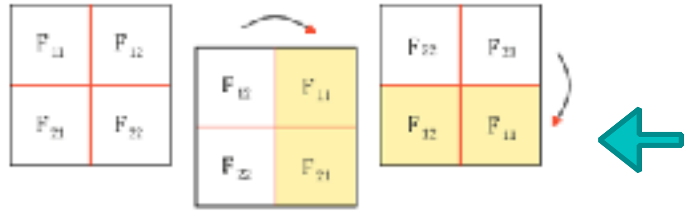
$$\frac{\partial L}{\partial X_{22}} = \frac{\partial L}{\partial O_{11}} * F_{22} + \frac{\partial L}{\partial O_{12}} * F_{21} + \frac{\partial L}{\partial O_{21}} * F_{12} + \frac{\partial L}{\partial O_{22}} * F_{11}$$

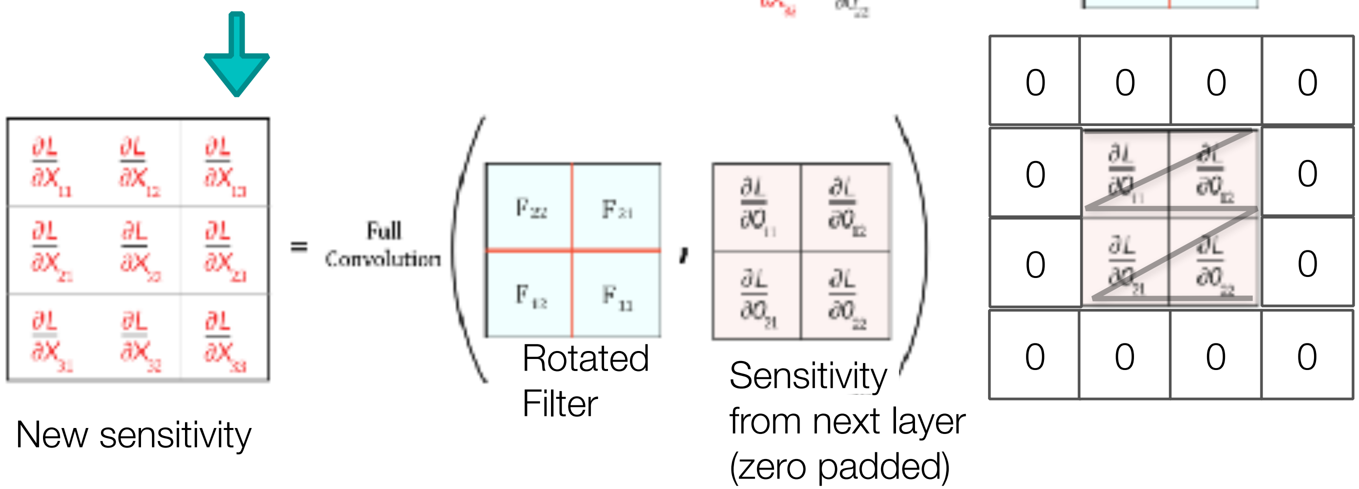
$$\frac{\partial L}{\partial X_{23}} = \frac{\partial L}{\partial O_{12}} * F_{22} + \frac{\partial L}{\partial O_{22}} * F_{12}$$

$$\frac{\partial L}{\partial X_{31}} = \frac{\partial L}{\partial O_{21}} * F_{21}$$

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

$$\frac{\partial L}{\partial X_{33}} = \frac{\partial L}{\partial O_{22}} * F_{22}$$

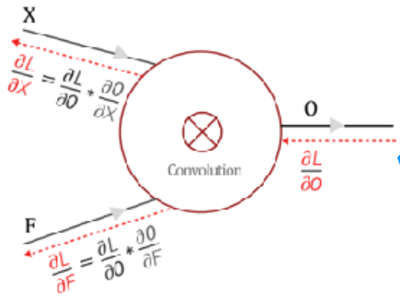




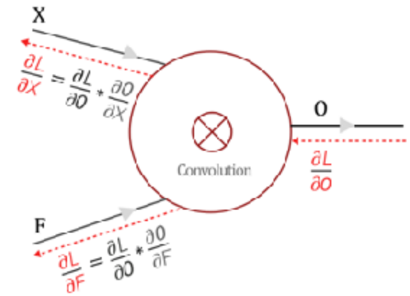
https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

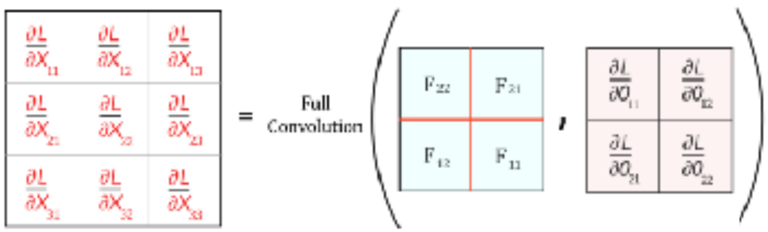
Summary

Filters at layer L-1



Filters at layer L

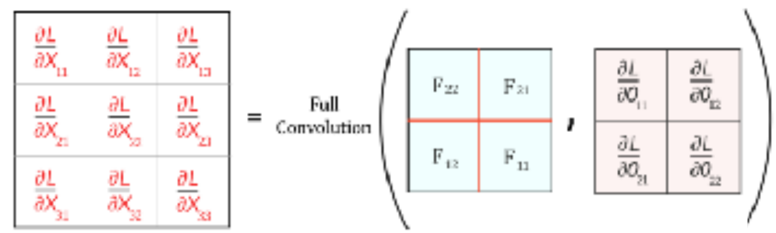




New sensitivity

Rotated Filter

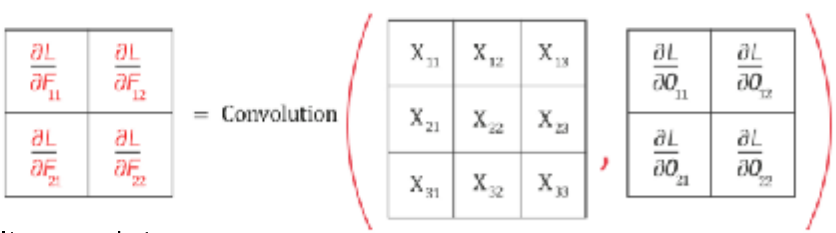
Sensitivity from next layer



New sensitivity

Rotated Filter

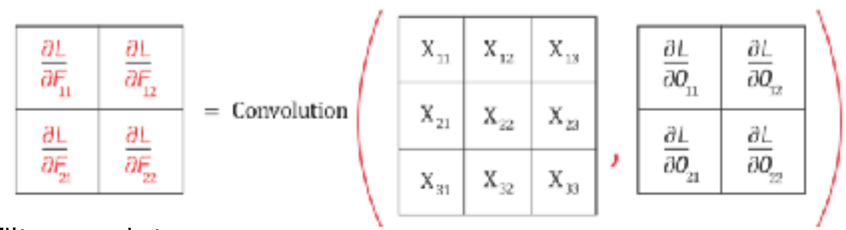
Sensitivity from next layer



Filter updates

Input

Sensitivity from next layer



Filter updates

Input

Sensitivity from next layer

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

Next Lecture

- More CNN architectures and CNN history