Lecture Notes for Machine Learning in Python

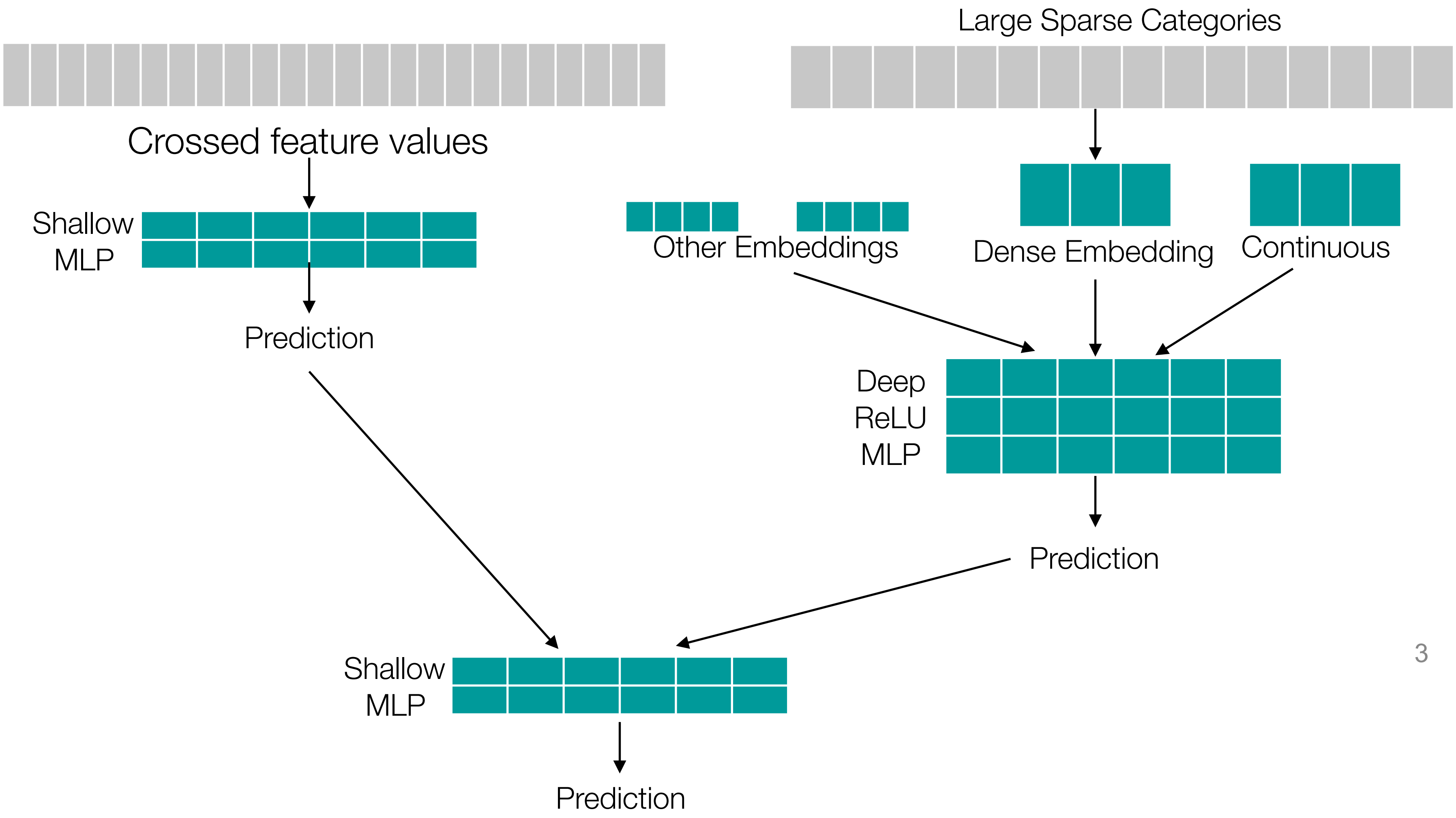
Professor Eric Larson

Basic Convolutional Neural Networks

Logistics and Agenda

- Logistics
  - Wide/Deep due this week
  - Next week: No Class on Tuesday
- Agenda
  - Review Wide and Deep Demo
  - Basic CNN architectures

Combining Memorization and Generalization



Demo

Wide and Deep

The awful dataset:
Toy Census Data Example

Other tutorials:

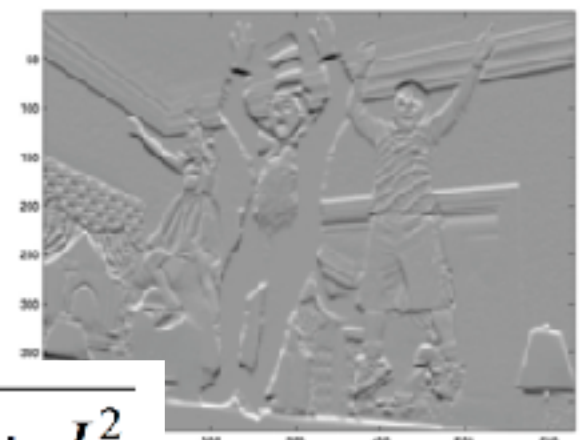
https://www.tensorflow.org/tutorials/wide_and_deep

Convolutional Neural Networks

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

What we did before

- the gradient (2D derivative)

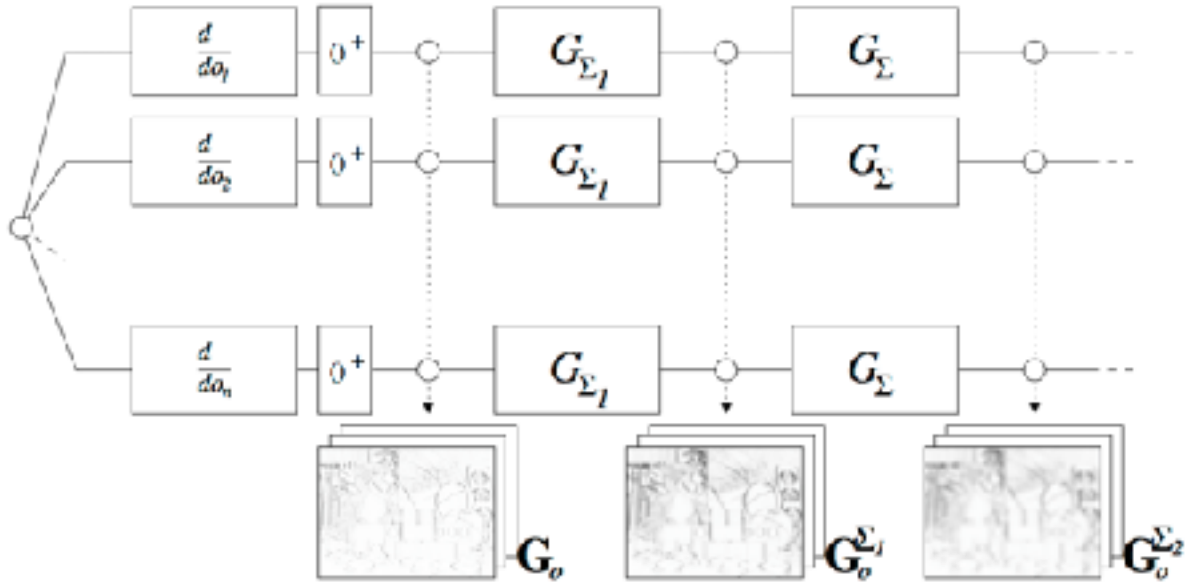


$$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$$

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

images: Jianbo Shi, Upenn

What we did before



take normalized histogram at point u,v

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

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

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

CNN Overview

- First layer(s):
  - convolution with different filters
  - nonlinearity
  - pooling
  - Each pooling layer *can* make the input image "smaller"
    - more summative explanations
- Final layers are densely connected
  - typically multi-layer perceptrons

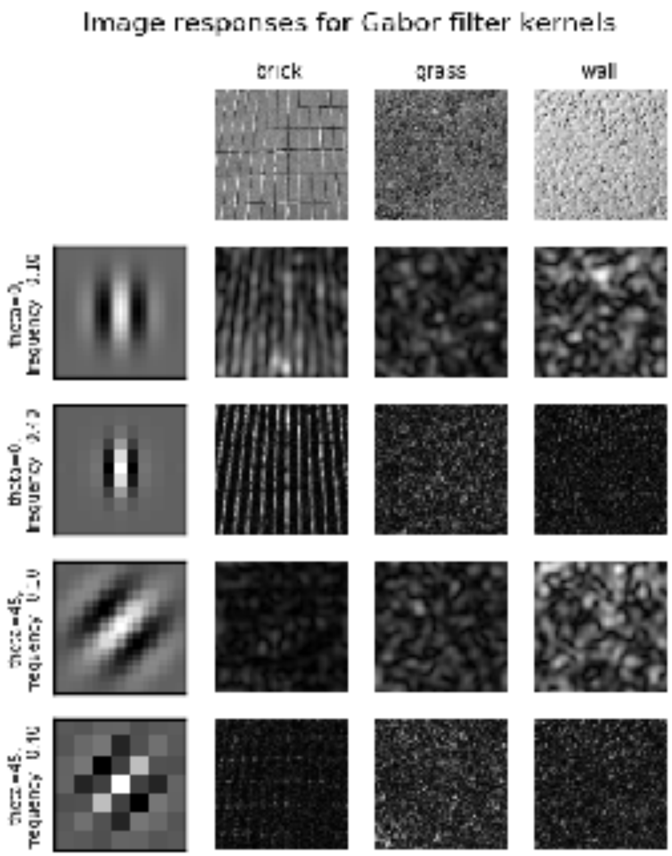
CNN Overview: Self Test

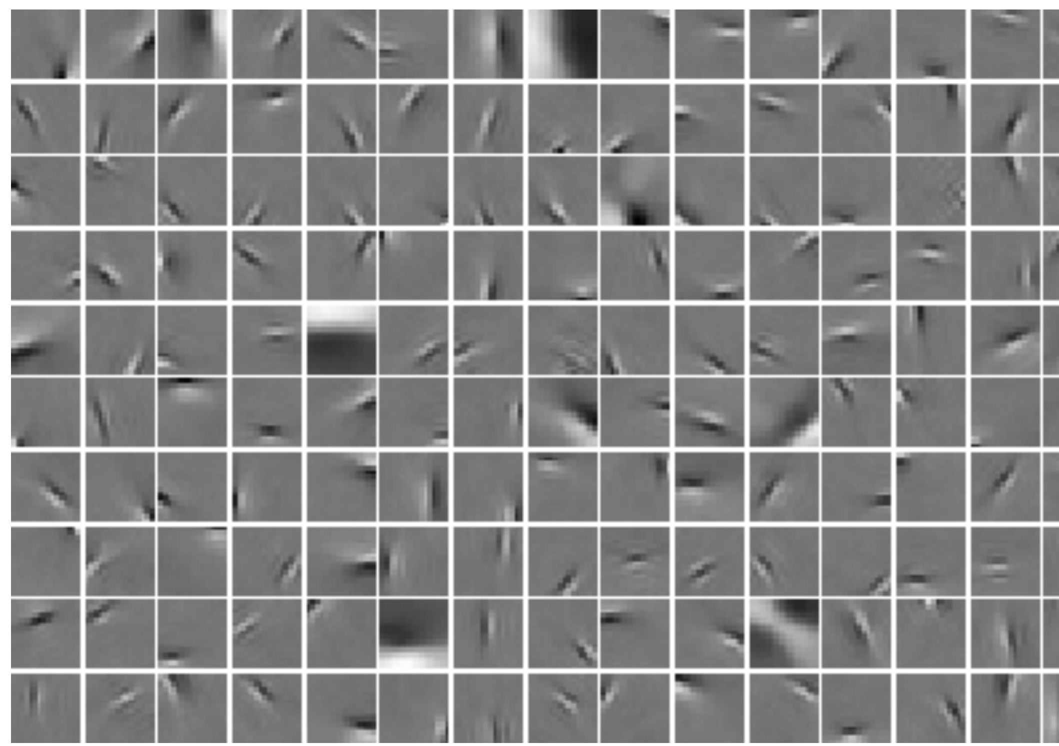
- First layer(s):
  - convolution with different filters
  - nonlinearity
  - pooling
  - Each pooling layer *can* make the input image "smaller"
    - more summative explanations
- Final layers are densely connected
  - typically multi-layer perceptrons

- 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

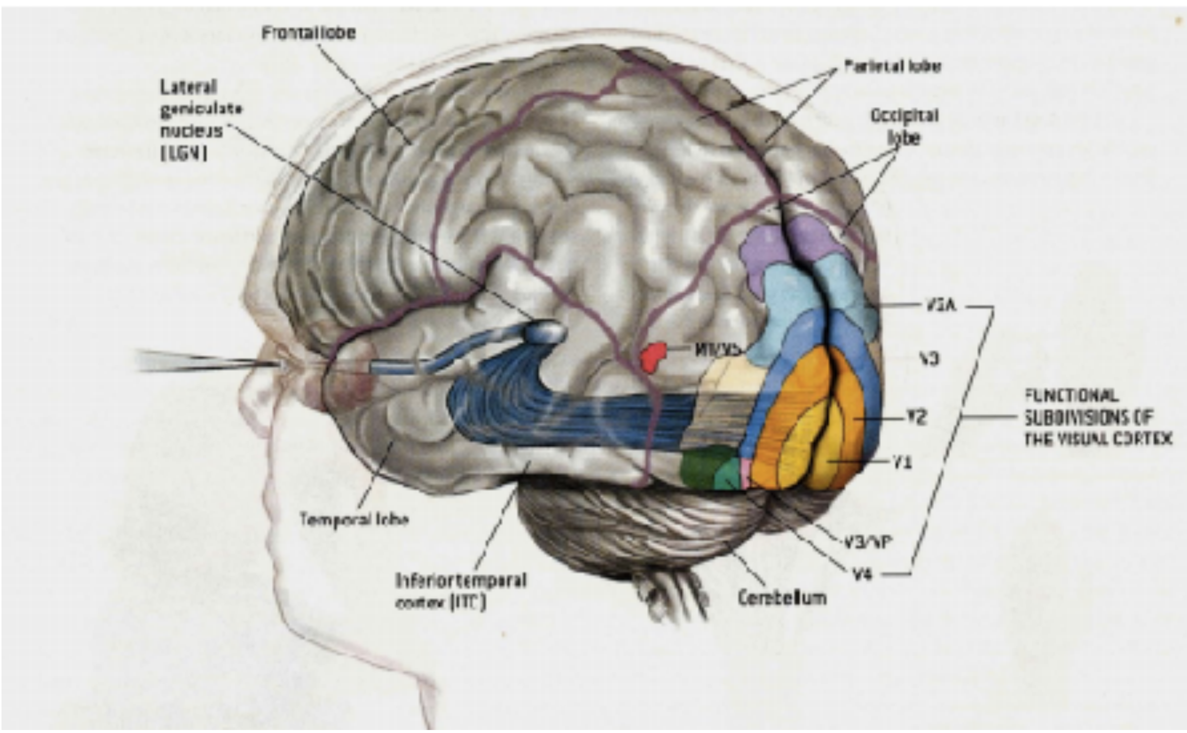
- Why perform lots of filtering?
    - recall gabor filtering?





CNN Filtering

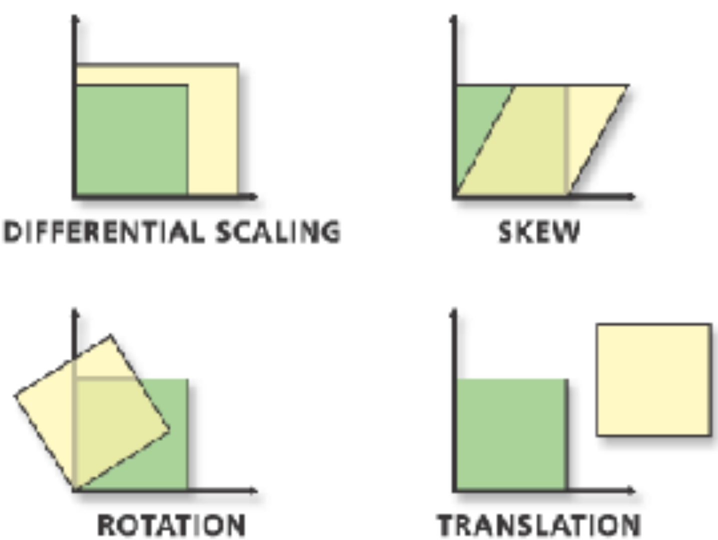
- Why perform lots of filtering?
  - recall gabor filtering?

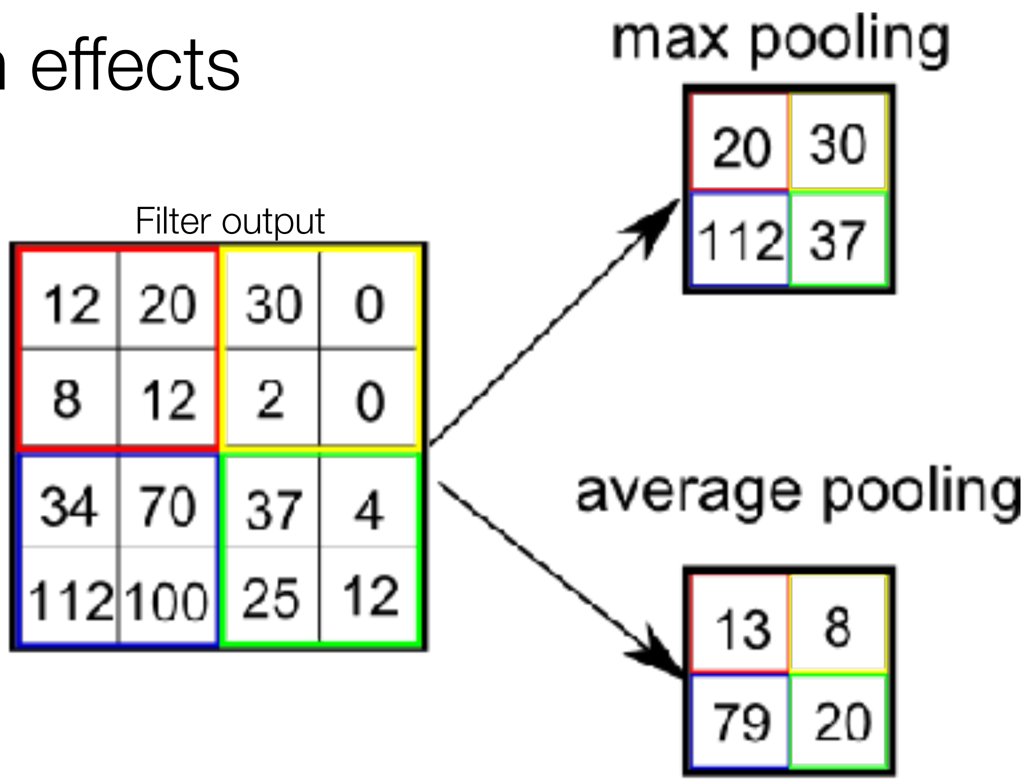


V1	Motion
V2	Stereo
V3	Color
V3a	Texture segregation
V3b	Segmentation, grouping
V4	Recognition
V7	Face recognition
MT	Attention
MST	Working memory/mental imagery

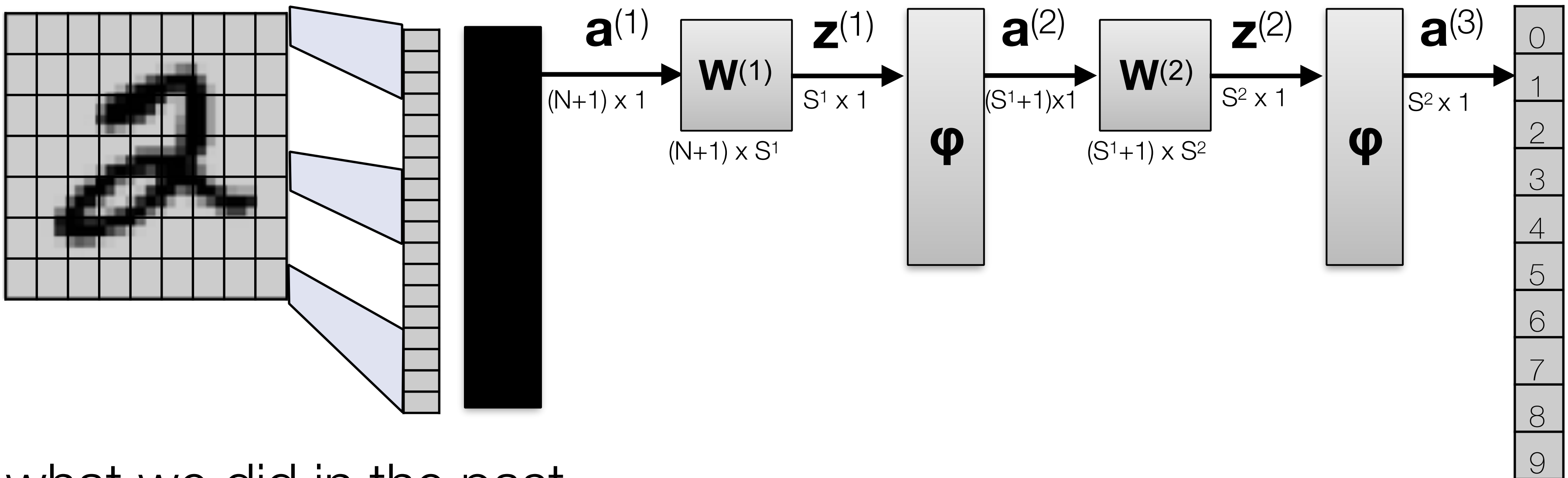
CNN Pooling

- Why perform pooling?
- Why max pooling?
  - reduce translation effects
  - param reduction





From Fully Connected to CNN



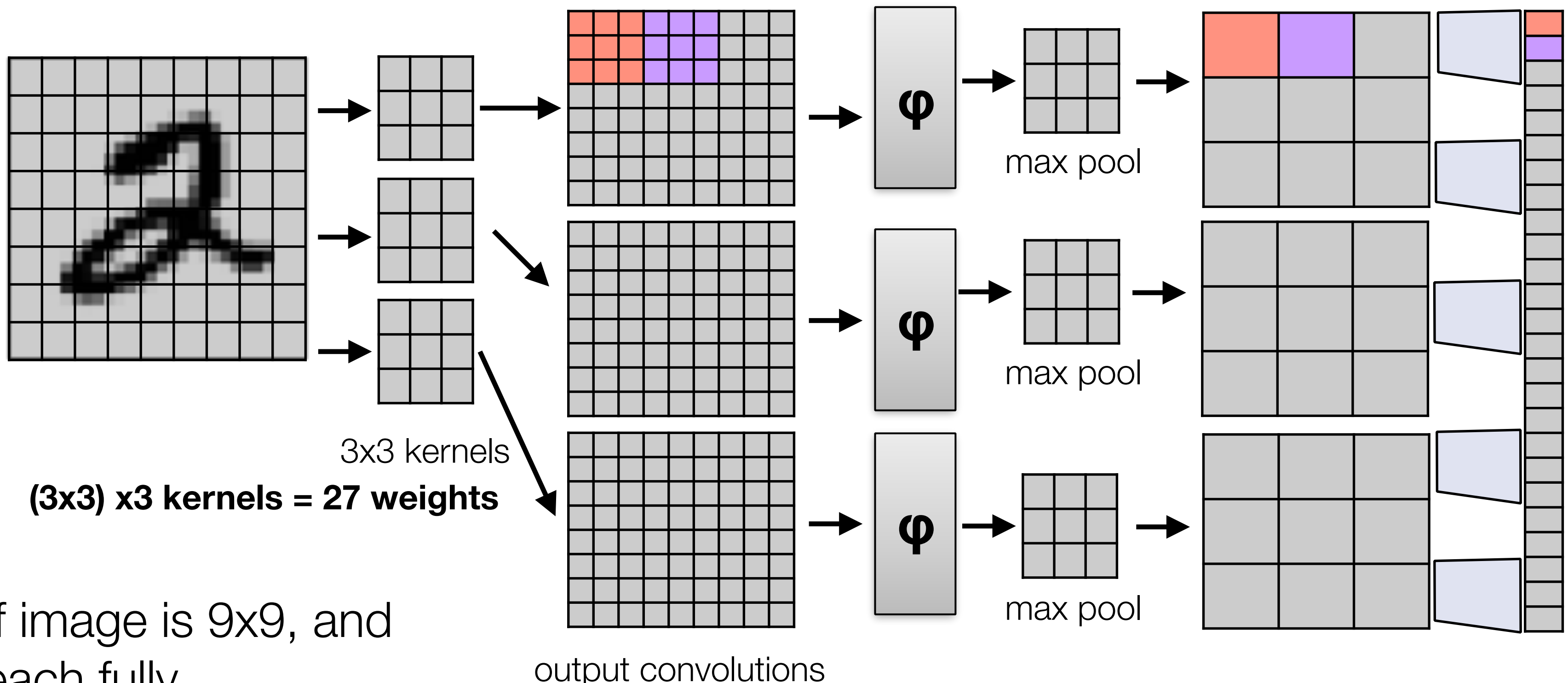
what we did in the past

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

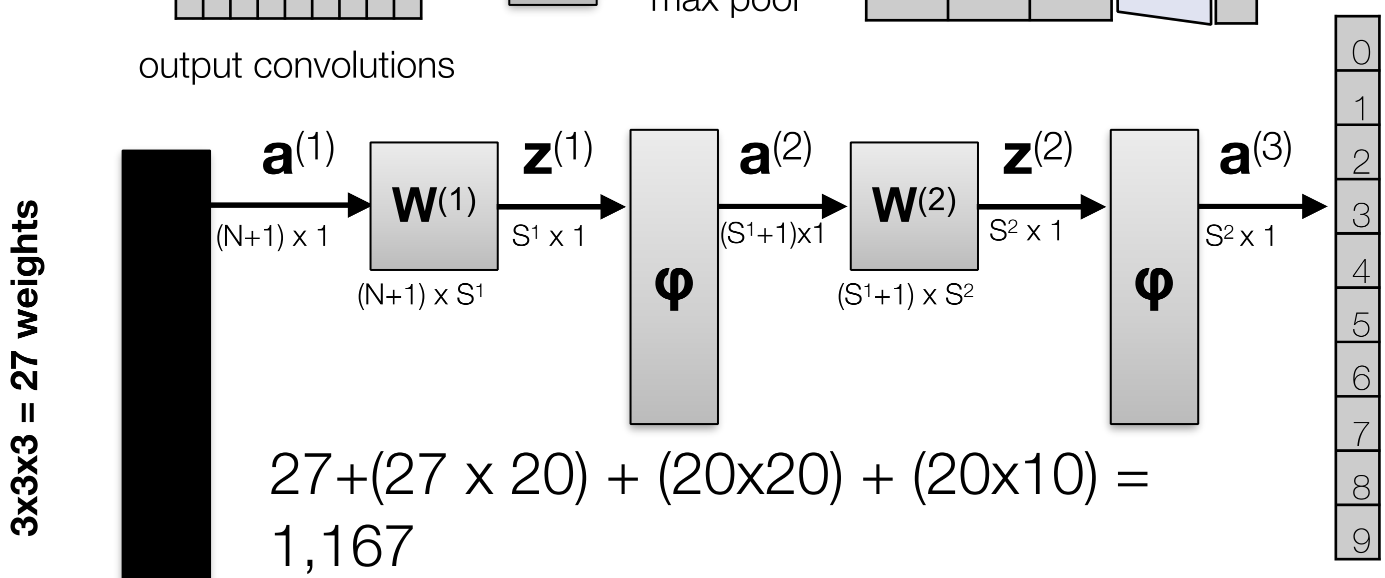
$(K^2 \times 20) + (20 \times 20) + (20 \times 10) = 600 + 20\ K^2$

for $9 \times 9 = 600 + 20 \times 9^2 =$ 2,220 parameters

From Fully Connected to CNN

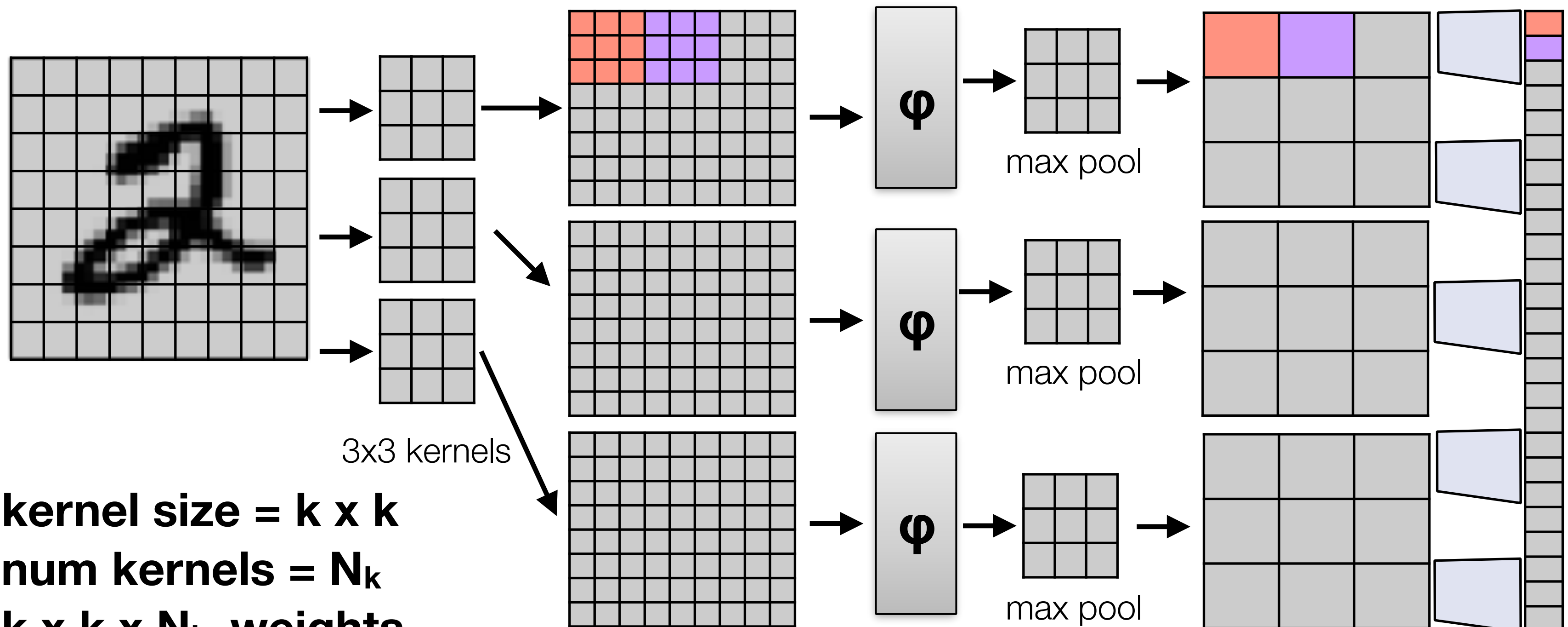


If image is 9x9, and each fully connected layer is 20 hidden neurons wide, how many parameters are in this NN (ignore bias)?



27+(27 x 20) + (20x20) + (20x10) = 1,167

From Fully Connected to CNN

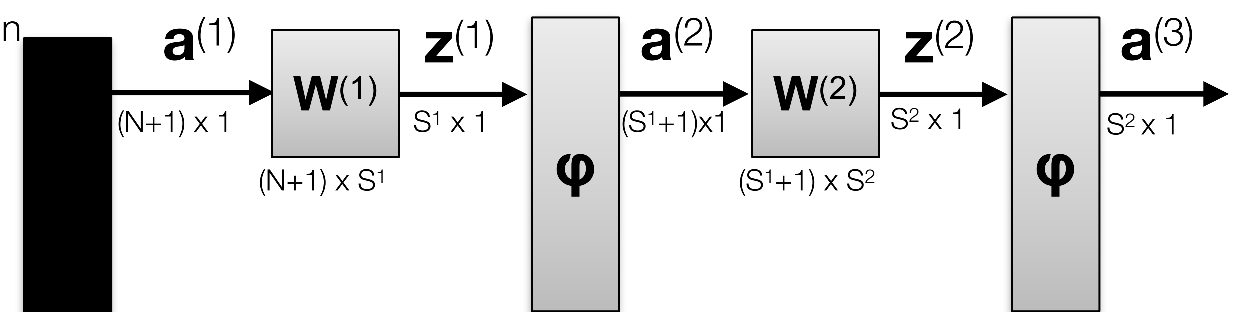


kernel size = k x k
num kernels = N_k
$k \times k \times N_k$ weights

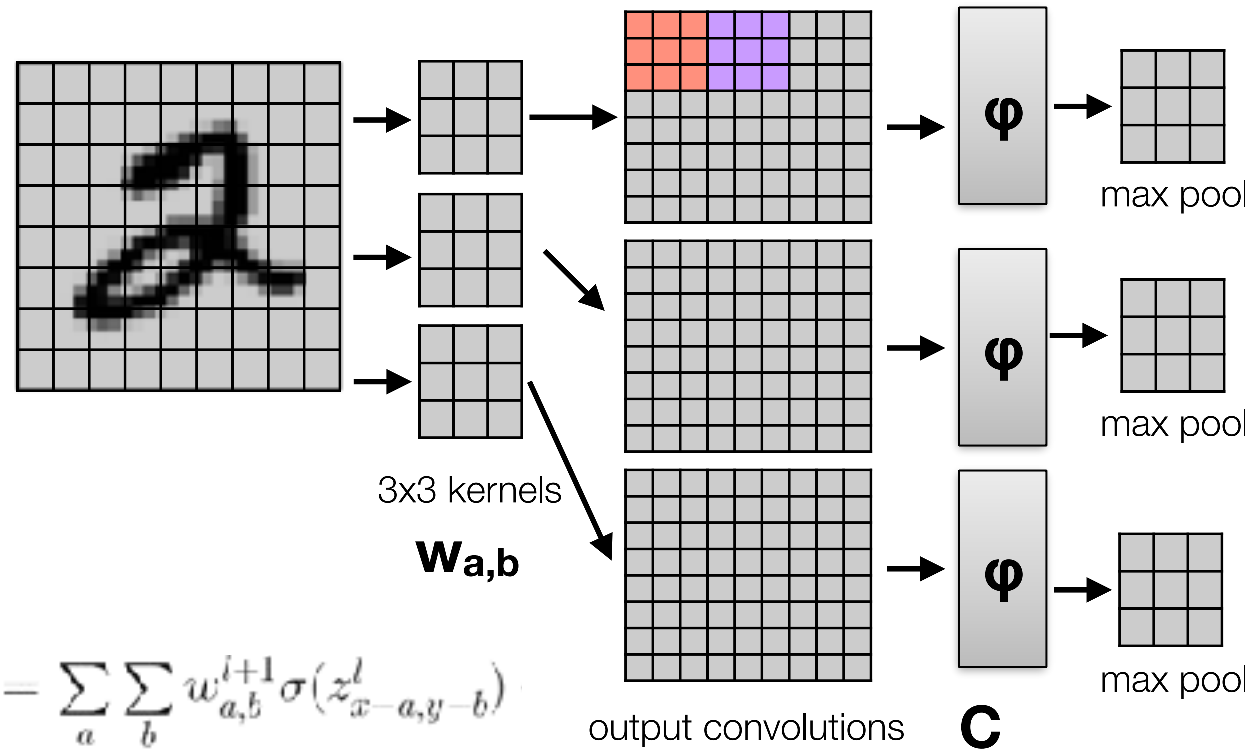
pooling outputs
if stride = k
$(K/k) \times (K/k) \times N_k$

total inputs
$N_k \times (k^2 + K^2/k^2)$

num filters
filter dimension
image dimension



CNN gradient



$$= \sum_a \sum_b w^{l+1}_{a,b} \sigma(z^l_{x-a,y-b})$$

Derivative of max pool is easy:

for each input x_i

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

Derivative of convolution is more involved:

$$\frac{\partial C}{\partial w^l_{a,b}} = \sum_x \sum_y \frac{\partial C}{\partial z^l_{x,y}} \frac{\partial z^l_{x,y}}{\partial w^l_{a,b}} = \sum_x \sum_y \delta^l_{x,y} \frac{\partial\left(\sum_{a'} \sum_{b'} w^l_{a',b'} \sigma(z^l_{x-a',y-b'}) + b^l_{x,y}\right)}{\partial w^l_{a,b}} =$$

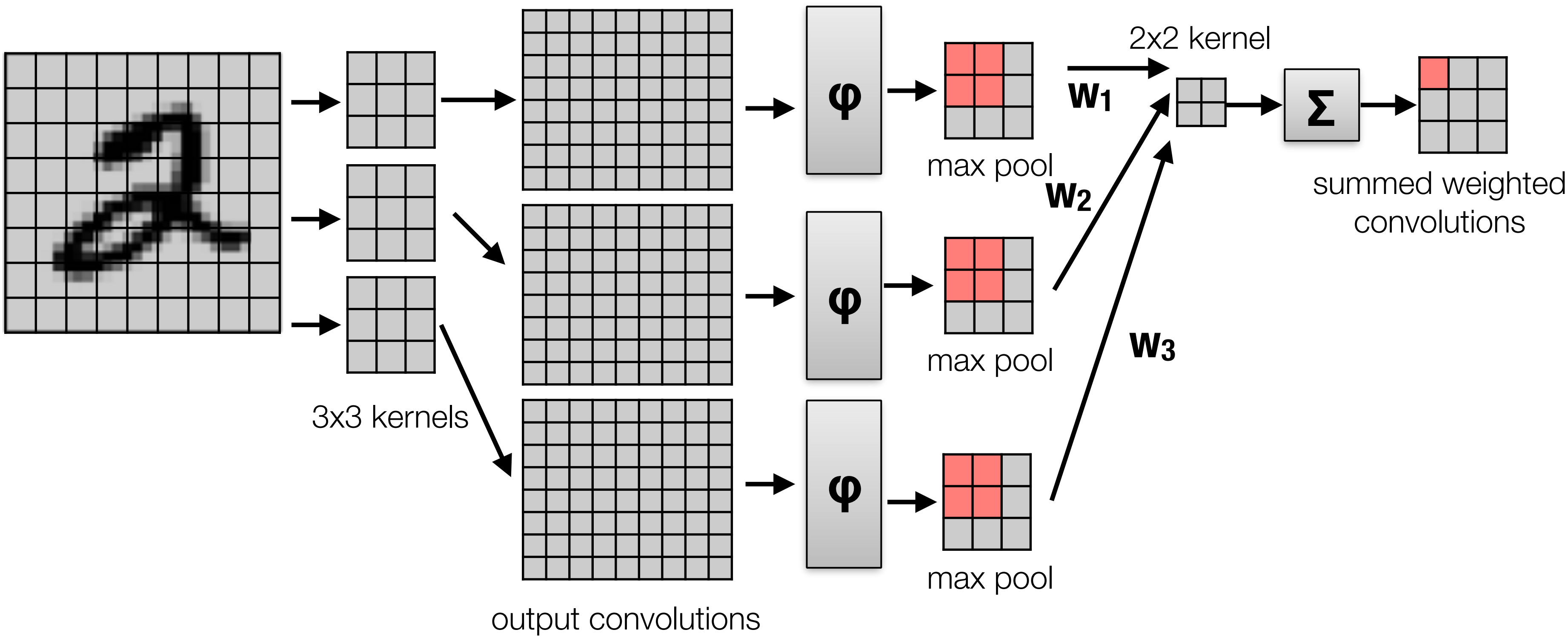
$$\sum_x \sum_y \delta^l_{x,y} \sigma(z^{l-1}_{x-a,y-b}) = \delta^l_{a,b} * \sigma(z^{l-1}_{-a,-b}) = \delta^l_{a,b} * \sigma(ROT180(z^{l-1}_{a,b}))$$

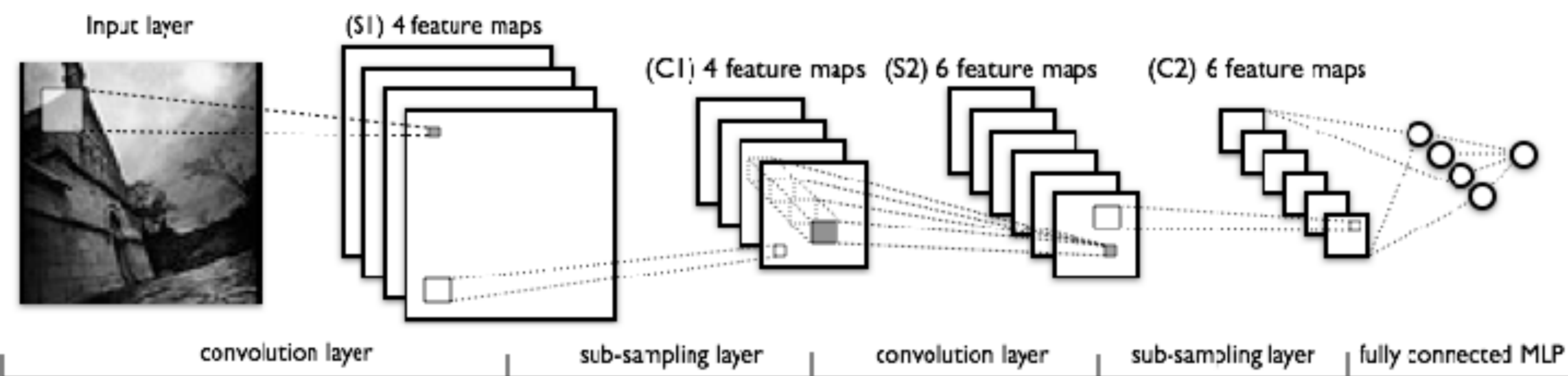
https://grzegorzgwardys.wordpress.com/2016/04/22/8/

CNN gradient

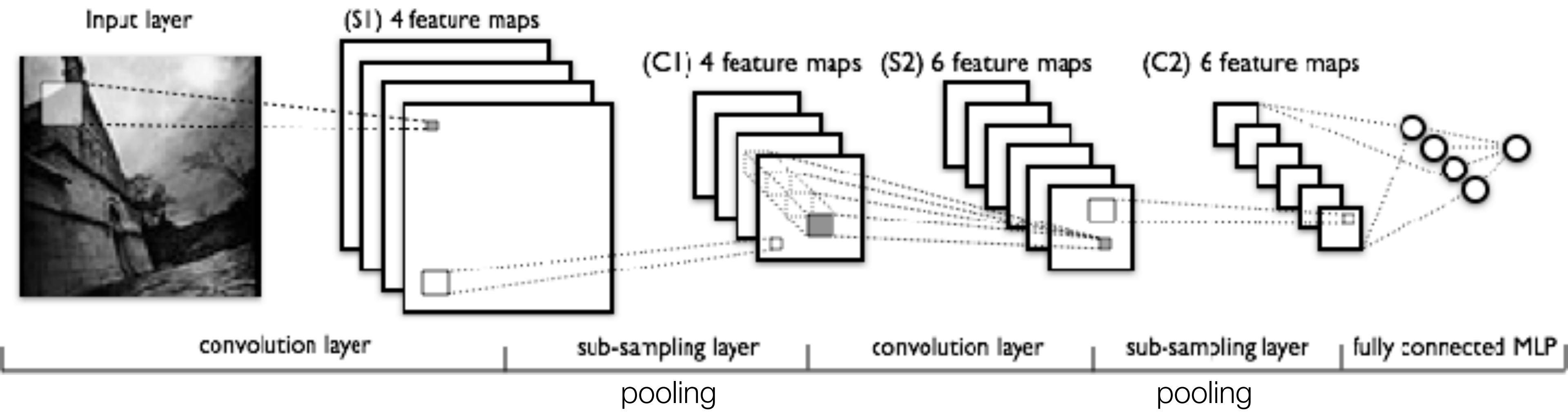
- But we really want to understand the process!
- These are great guides:
  - https://grzegorzgwardys.wordpress.com/2016/04/22/8/
  - http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/

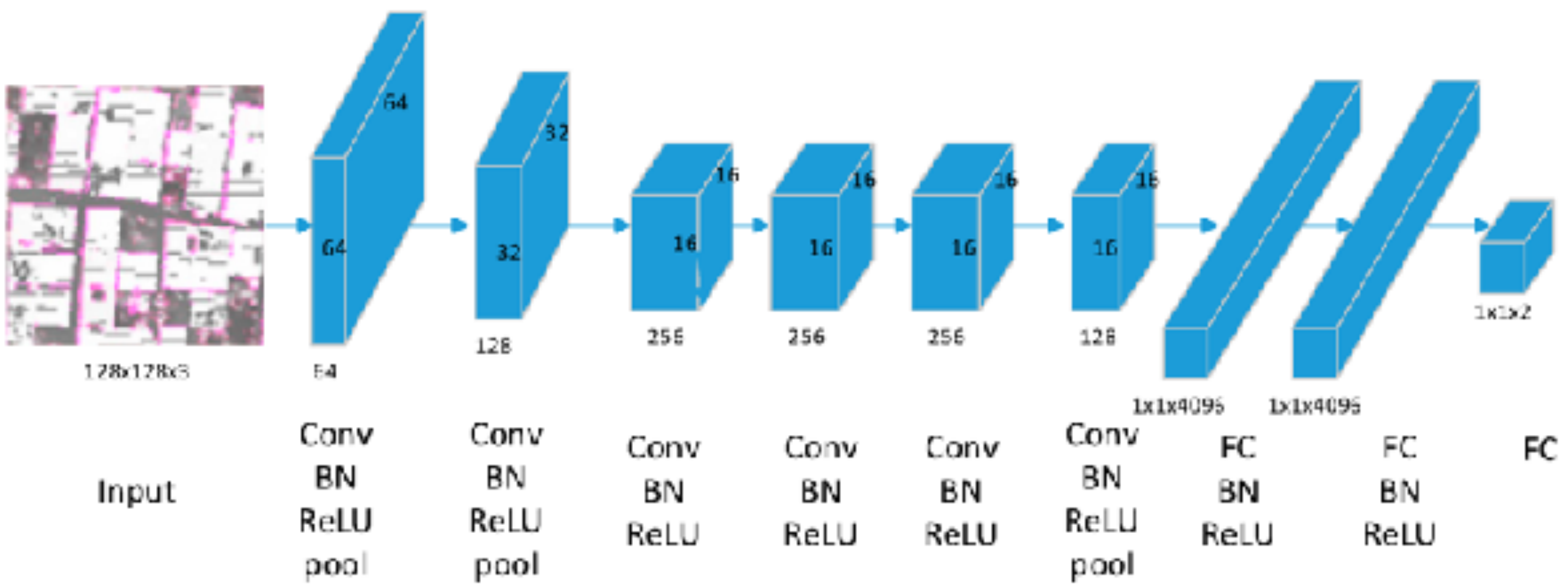
CNN adding more convolutional layers





Some Example CNN Architectures





CNN: What does it all mean?

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

Demo

TensorFlow and Basic CNNs

Convolutional Neural Networks
in TensorFlow
with Keras

Next Lecture

- More CNN architectures and CNN history

Lecture Notes for

Machine Learning in Python

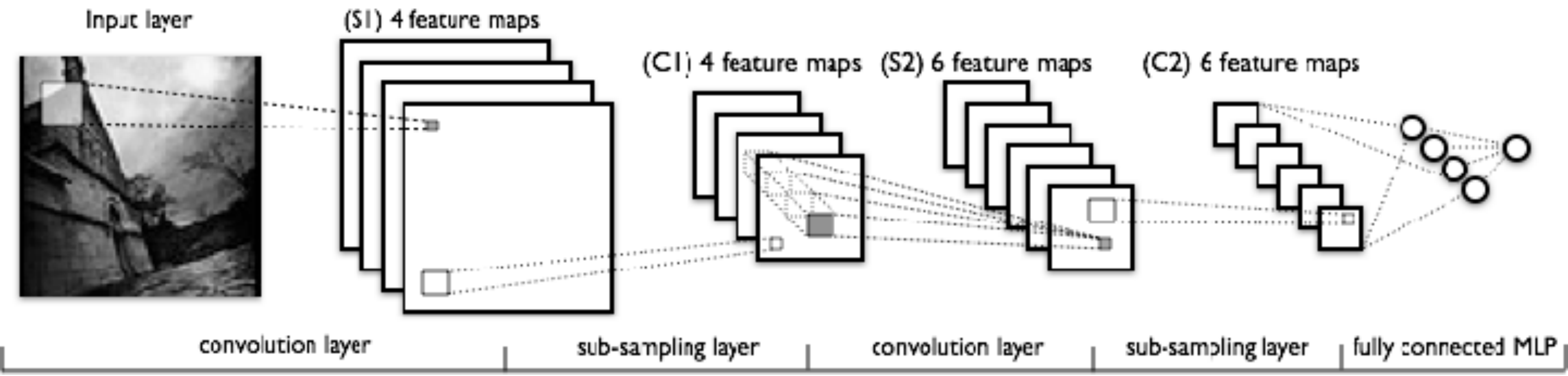
Professor Eric Larson

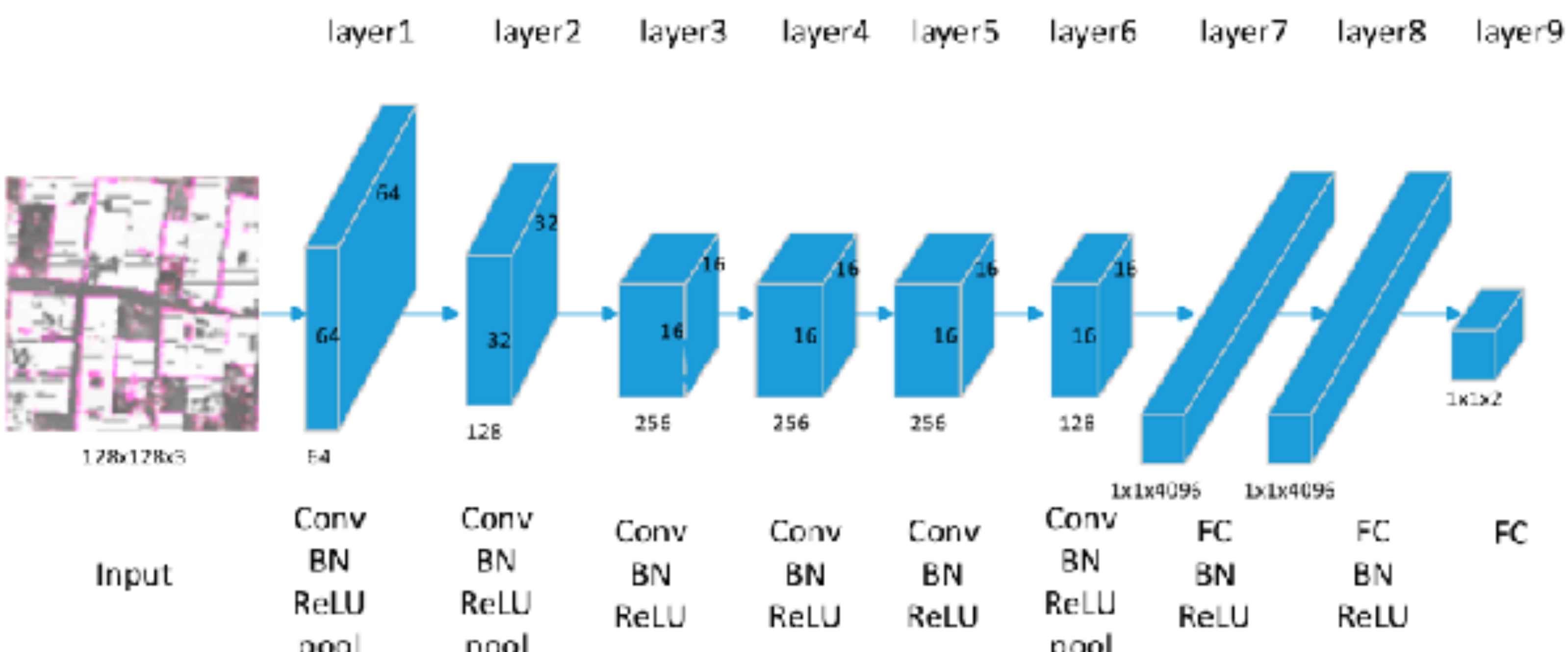
More Advanced Convolutional Networks

Class logistics and Agenda

- CNN Lab due next week
- But we will start RNN next time
- Agenda:
  - History of CNNs
  - with Modern CNN Architectures

Last Time:


Input layer
(S1) 4 feature maps
(C1) 4 feature maps
(S2) 6 feature maps
(C2) 6 feature maps
convolution layer
sub-sampling layer
convolution layer
sub-sampling layer
fully connected MLP
pooling
pooling

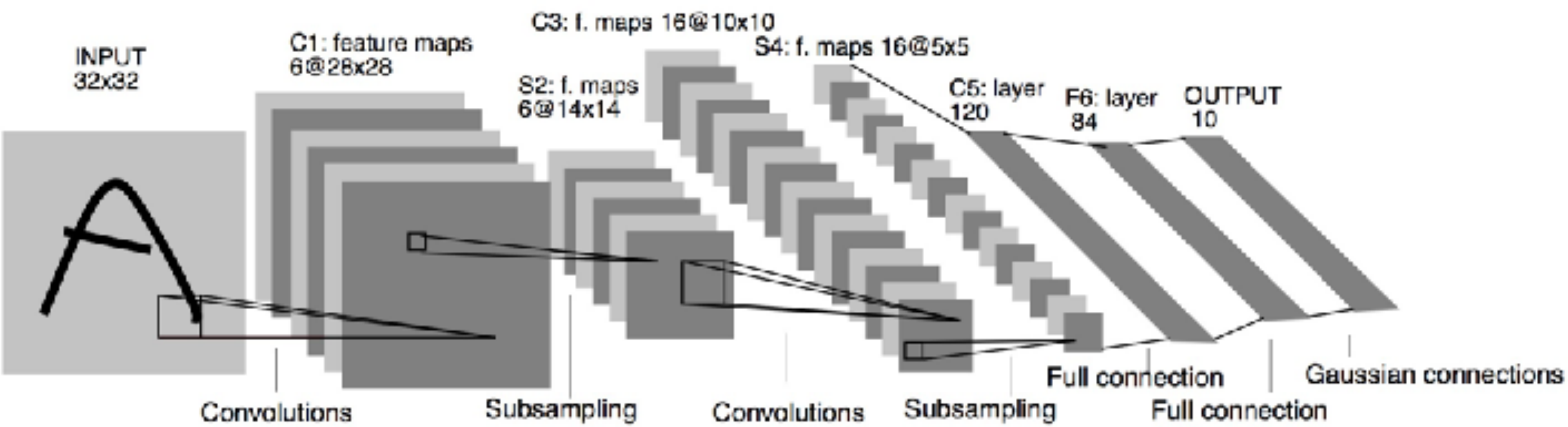

layer1
layer2
layer3
layer4
layer5
layer6
layer7
layer8
layer9
128x128x3
64
128
256
256
256
128
1x1x4096
1x1x4096
1x1x2
Input
Conv BN ReLU pool
Conv BN ReLU pool
Conv BN ReLU
Conv BN ReLU
Conv BN ReLU
Conv BN ReLU pool
FC BN ReLU
FC BN ReLU
FC

Types of CNN, 1988-1998

Heads Facebook AI Team

- **LeNet-1** (1988)
  - ~2600 params, not many layers
- **LeNet-5** (1998)
  - 7 layers, gets excellent MNIST performance
- Major contribution, general structure:
  - conv=>pool=>non-linearity=> …=>MLP

avg **tanh or sigmoid**



https://culurciello.github.io/tech/2016/06/04/nets.html

CNN History

- List of major breakthroughs from 1998 through 2010 in convolutional networks:

¯_(ツ)_/¯

- 2010

Types of CNN, 2010

AI Researcher
IDSA, Switzerland

- **Circesan Net**
- Publishes code for running CNN via GPU
  - Subsequently wins 5 international competitions
  - from stop signs => cancer detection
- Major contribution: NVIDIA parallelized training algorithms

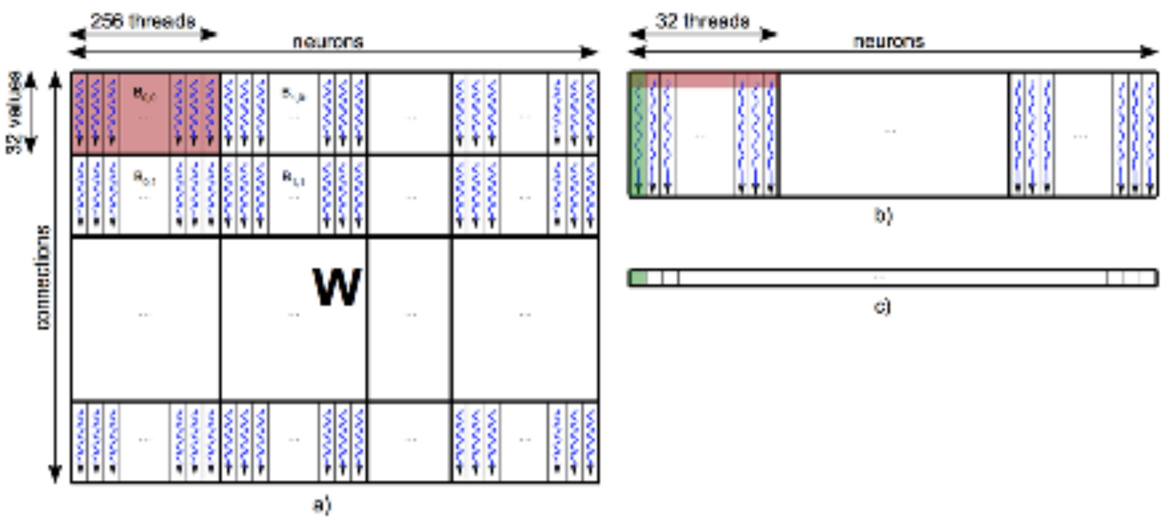
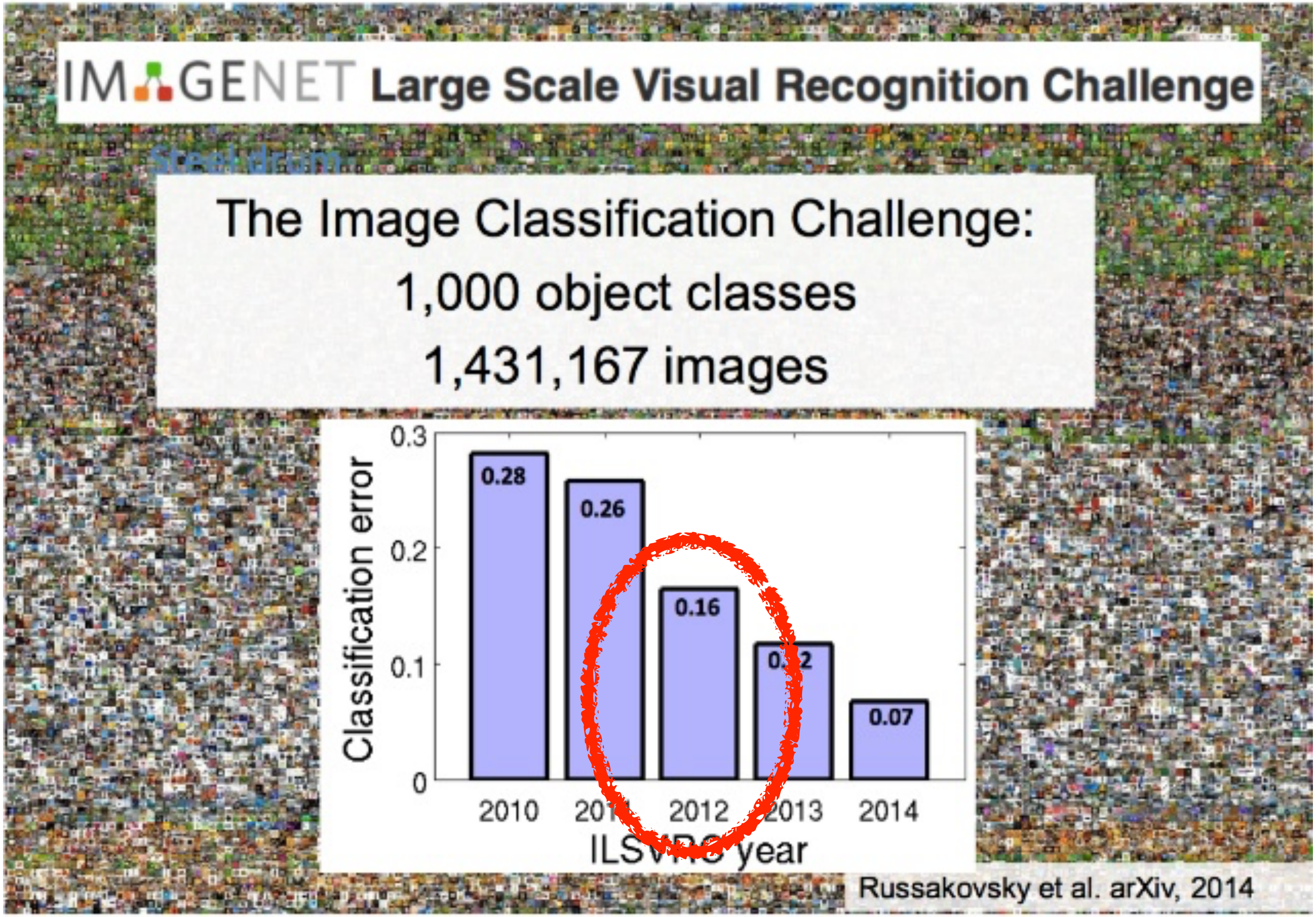


Figure 2: Forward propagation: a) mapping of kernel 1 grid onto the padded weight matrix; b) mapping the kernel 2 grid onto the partial dot products matrix; c) output of forward propagation.

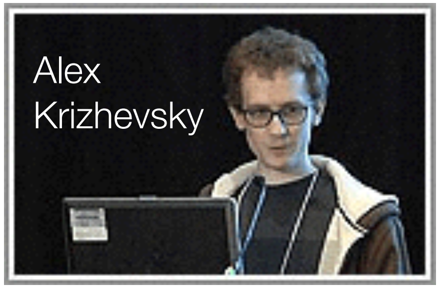
https://arxiv.org/pdf/1003.0358.pdf

ImageNet Competition (2010)



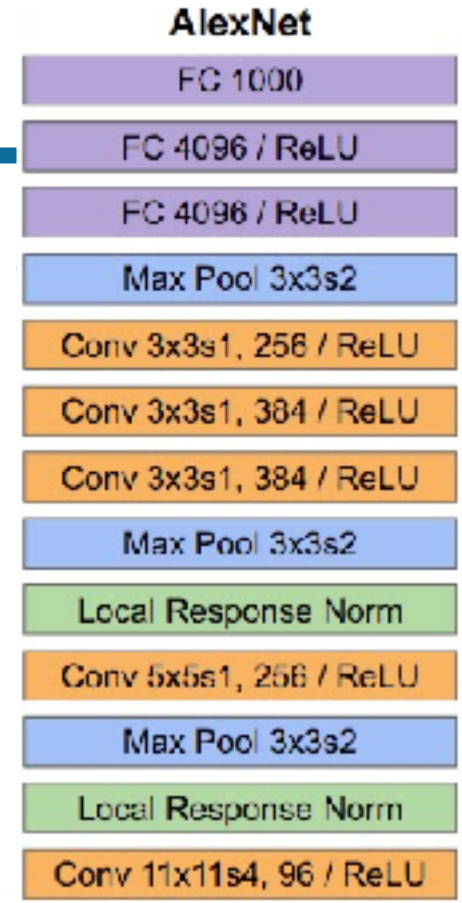
https://www.slideshare.net/nmhkahn/case-study-of-convolutional-neural-network-61556303

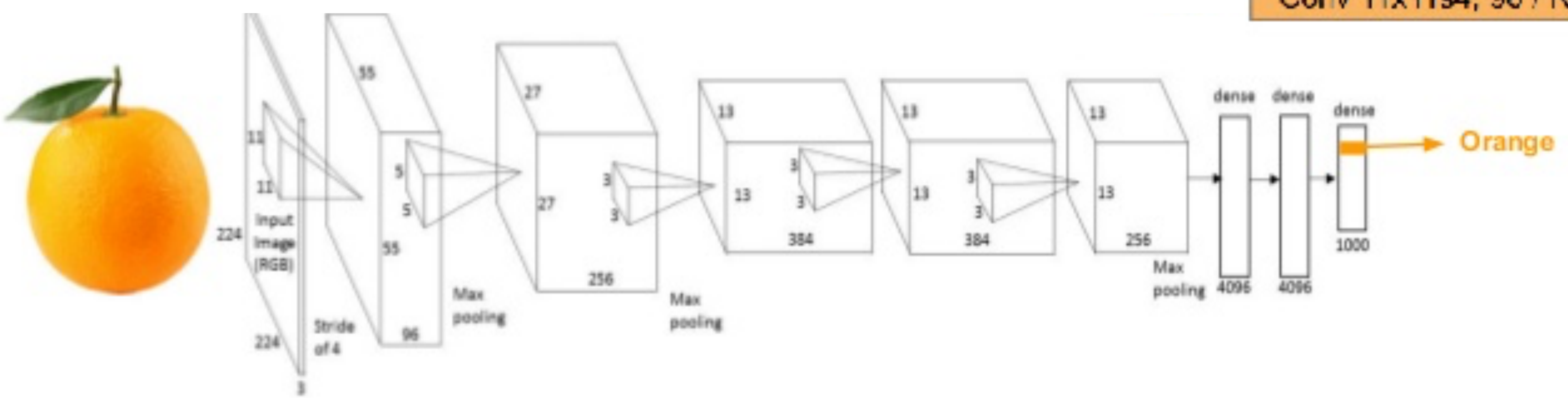
Types of CNN, 2012



Google

- **AlexNet**, Hinton is mentor
  - wins ImageNet competition
- Major contributions:
  - dropout for regularization
  - systematic use of ReLU
  - data expansion
  - overlapping max pool



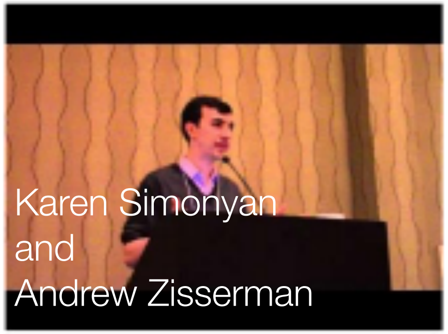


https://culurciello.github.io/tech/2016/06/04/nets.html

Warning


DON'T TRY WHAT YOU'RE
ABOUT TO SEE AT HOME
WE'RE WHAT YOU CALL
EXPERTS

Types of CNN, 2013



UNIVERSITY OF OXFORD

Google DeepMind

- Oxford **VGG Net** (Visual Geometry Group)
- Major contributions:
  - small cascaded kernels
  - way more layers (19 versus ~7)
  - "emulates" biology "better"
  - trained on NVIDIA GPUs for 2-3 weeks

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

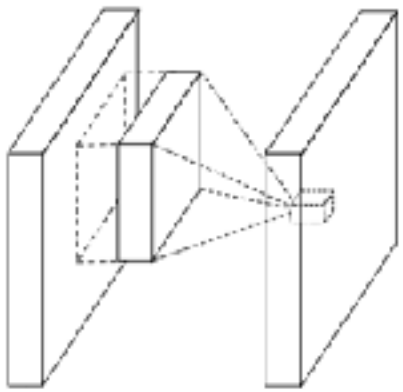
http://www.robots.ox.ac.uk/~vgg/practicals/cnn/

Types of CNN, 2014

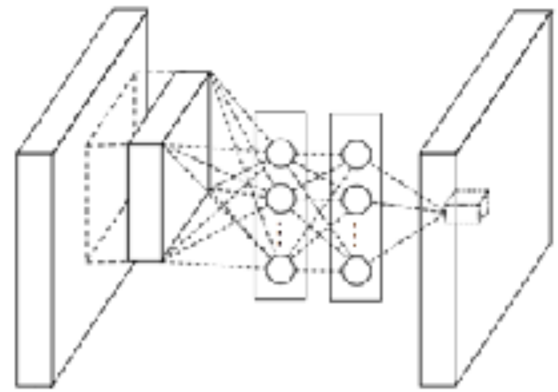
- Network in Network **NiN**
  - or MLPConv

Network In Network

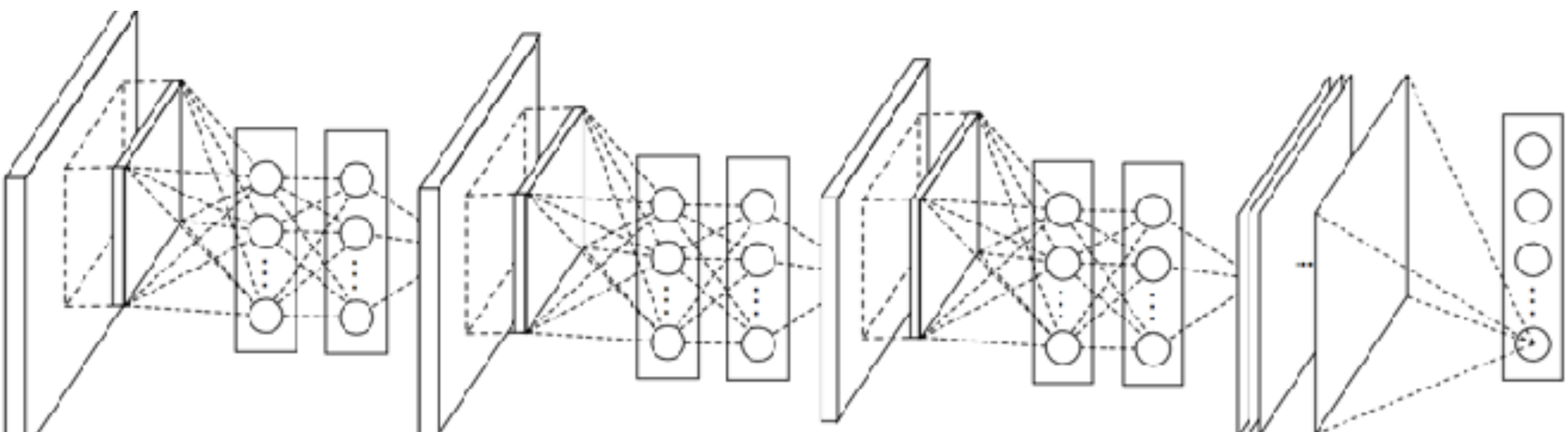
Min Lin[1,2], Qiang Chen[2], Shuicheng Yan[2]
[1]Graduate School for Integrative Sciences and Engineering
[2]Department of Electronic & Computer Engineering
National University of Singapore, Singapore
{linmin,chenqiang,eleyans}@nus.edu.sg



(a) Linear convolution layer

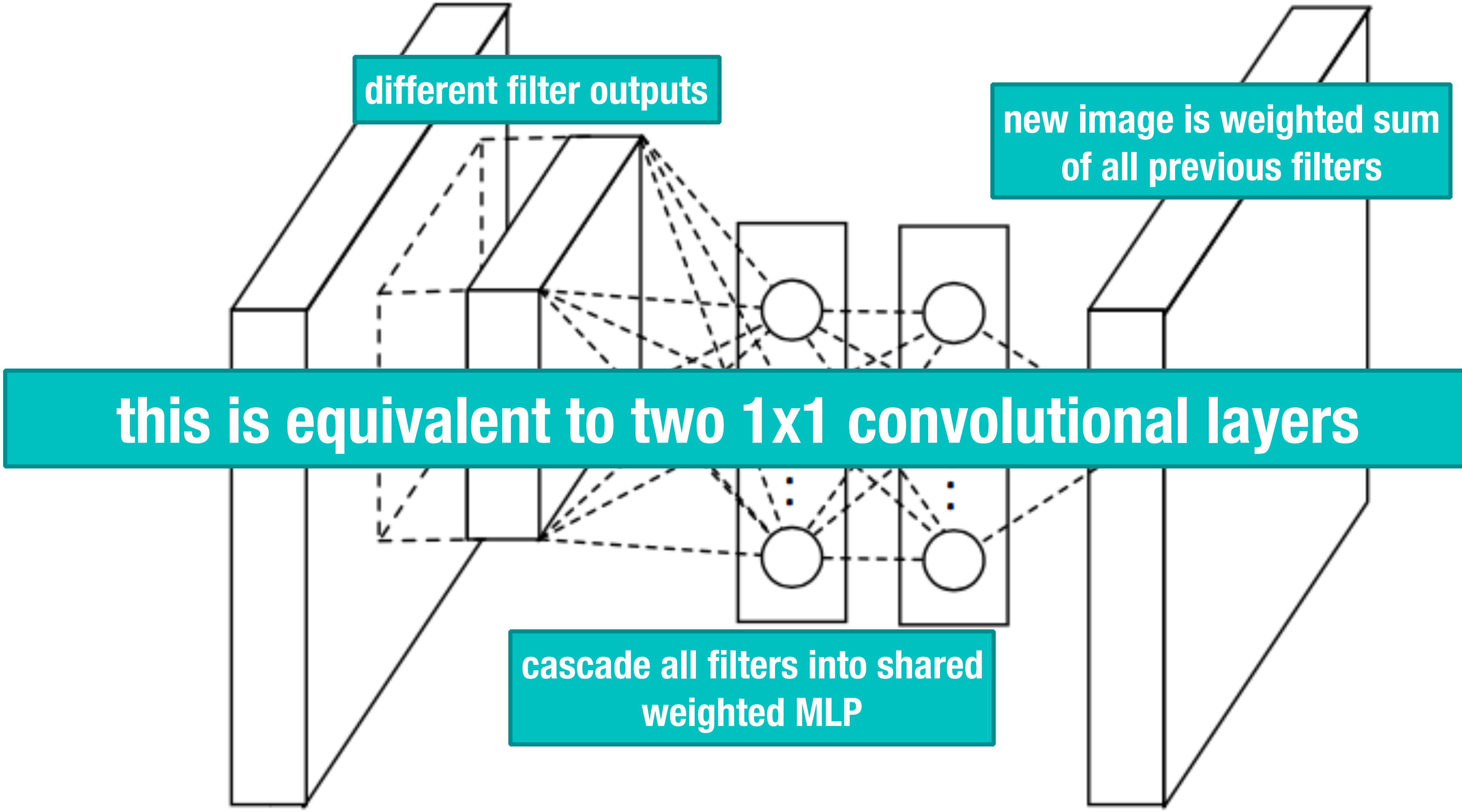


(b) Mlpconv layer



https://arxiv.org/pdf/1312.4400.pdf

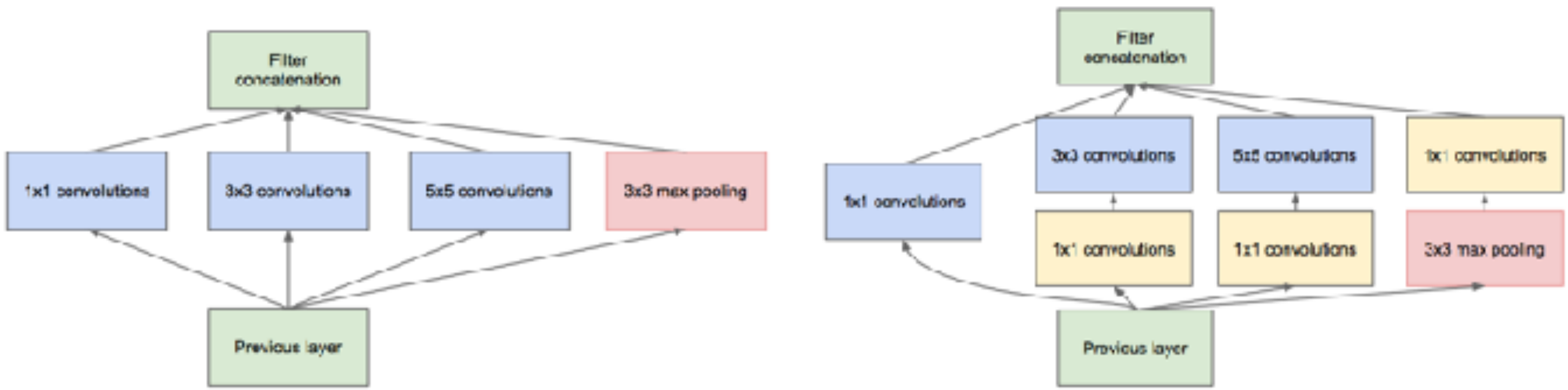
Types of CNN, 2014



https://arxiv.org/pdf/1312.4400.pdf

Types of CNN, 2014

- **GoogLeNet**
  - or **Inception V1**
- Major contribution:
  - bottleneck layering
  - parallel NiN



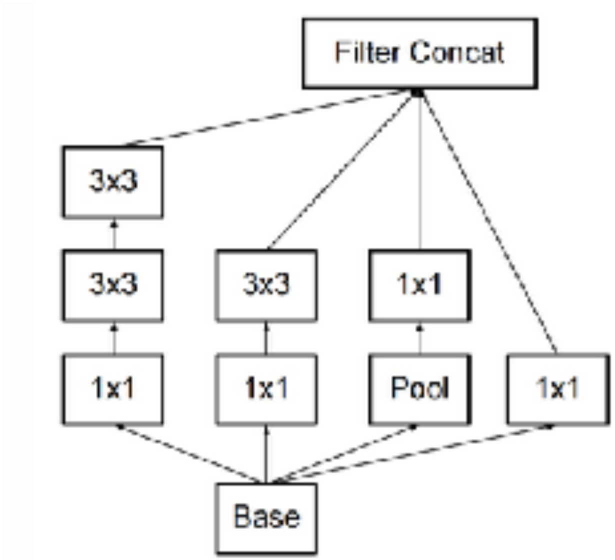
(a) Inception module, naïve version (b) Inception module with dimension reductions

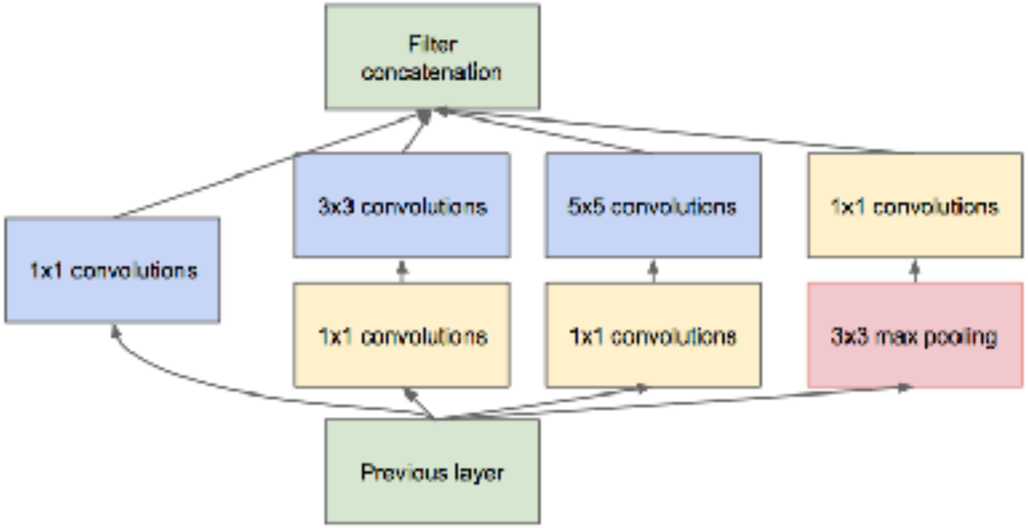
Figure 2: Inception module

https://arxiv.org/pdf/1409.4842.pdf

Types of CNN, 2015 February and December

- **Inception V2**, Inception V1 with batch normalization
- **Inception V3**:
    - replace 5x5 with multiple 3x3



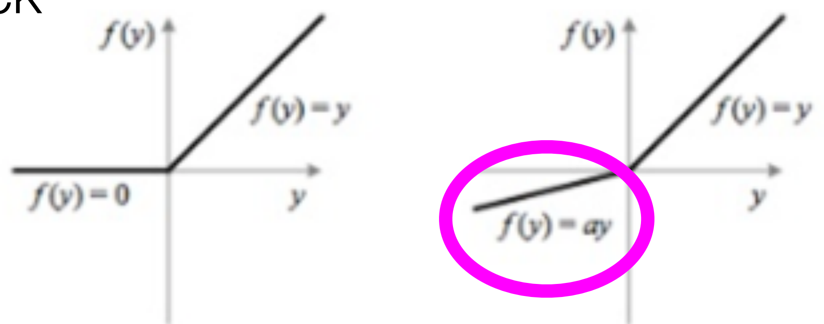


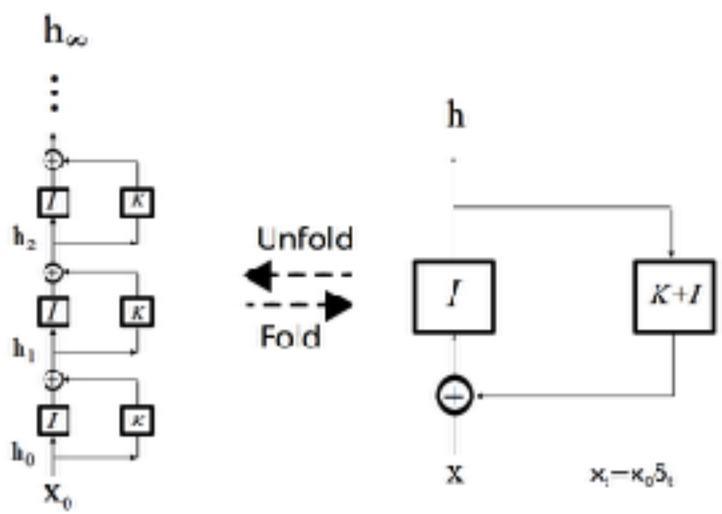
https://arxiv.org/pdf/1512.00567.pdf

Types of CNN, 2015 December

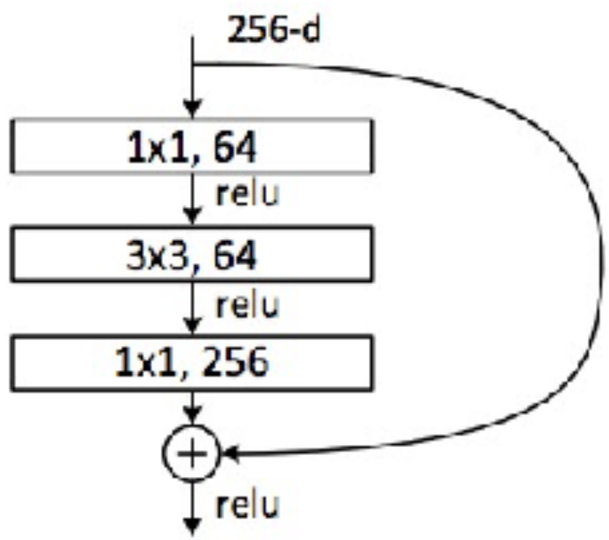
Microsoft Research

- Major Contributions:
  - ensembles, not strictly sequential
  - bio-plausible with feedback
- **ResNet**
  - PReLU: adaptive trained slope





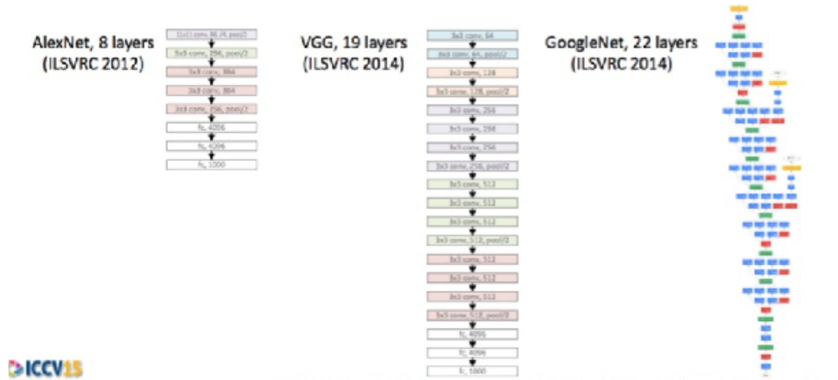
(A) ResNet with shared weights (B) ResNet in recurrent form

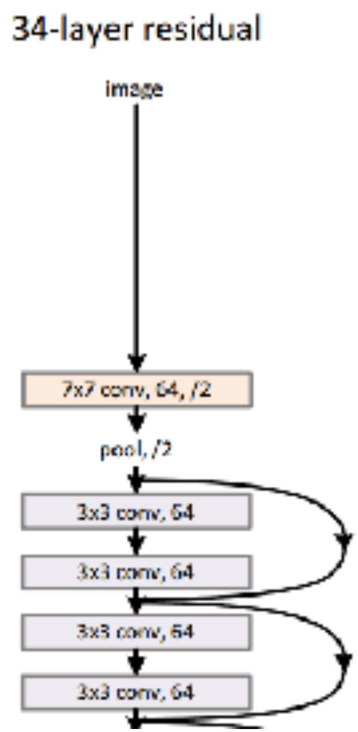


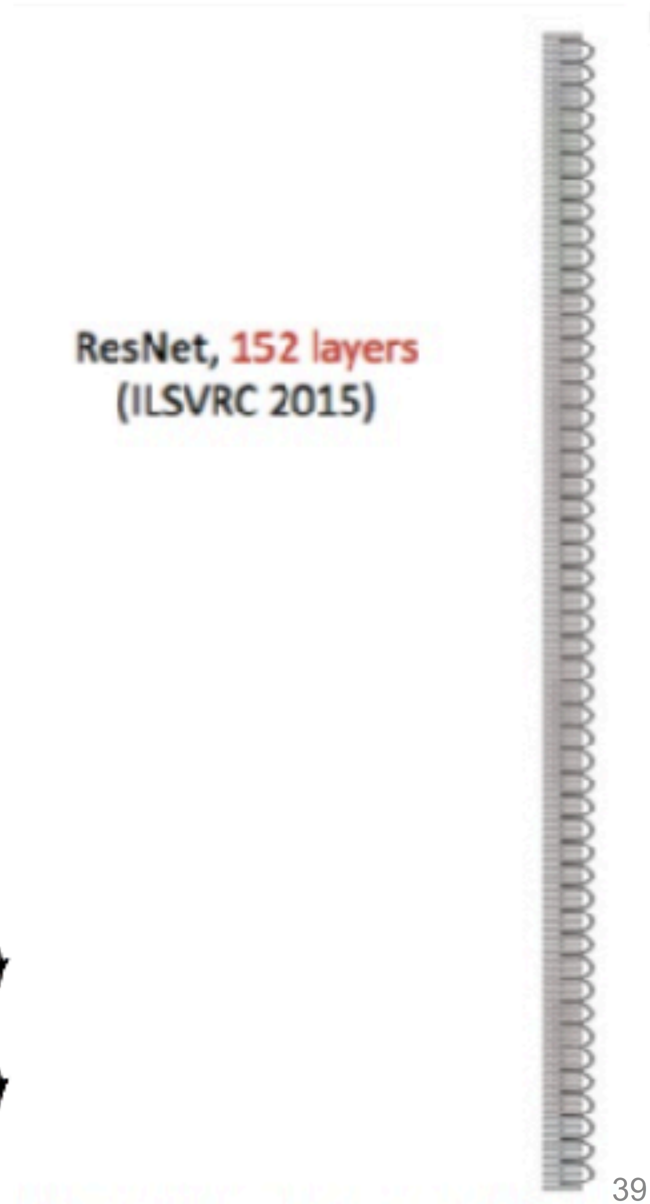
- NiN: double bypass layer
  - similar to bottelneck

How big are these networks?

How big are these networks?







Self Test

- We have seen a lot of different networks.
- The most important concept to understand in using convolutional neural networks is:
  - A. Use proper initialization of layers
  - B. Have plenty of data or use expansion
  - C. Set aside time for training
  - D. Use batch normalization

More Modern CNN Architectures

Even more Convolutional Neural Networks

…in TensorFlow

…with Keras

Next Time:

- Intro to Recurrent Neural Network Architectures
  - RNNs, GRUs, LSTMs
  - Training for characters