Lecture Notes for

Machine Learning in Python

Professor Eric Larson

An Ongoing History of Convolutional Networks

Class logistics and Agenda

- Wide/Deep Lab due soon!
- Agenda:
  - CNN Demo
  - History of CNNs
    - with Modern CNN Architectures
- Next Time:
  - More Advanced CNN 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

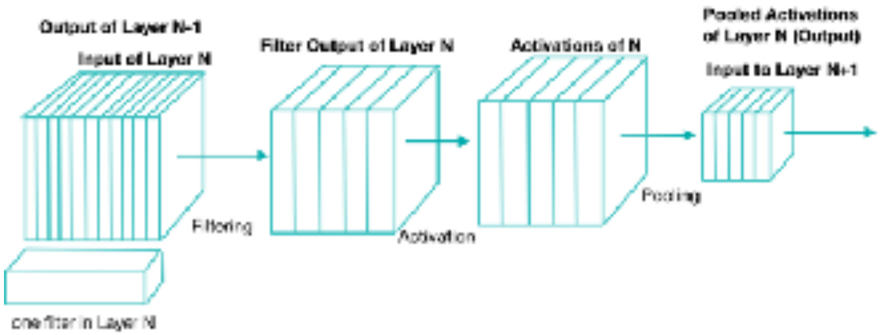
Wide and Deep Networks | **Convolutional Networks** | **Recurrent Networks**

Keras, Tensorflow
Intuition, Detailed implement.

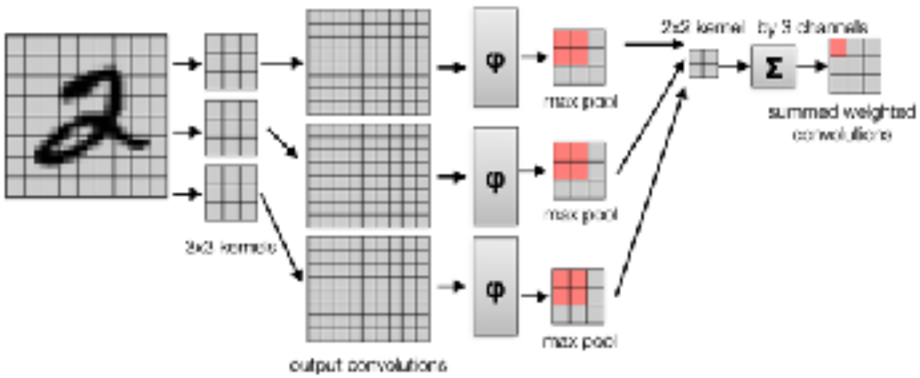
Ethics in Language Models

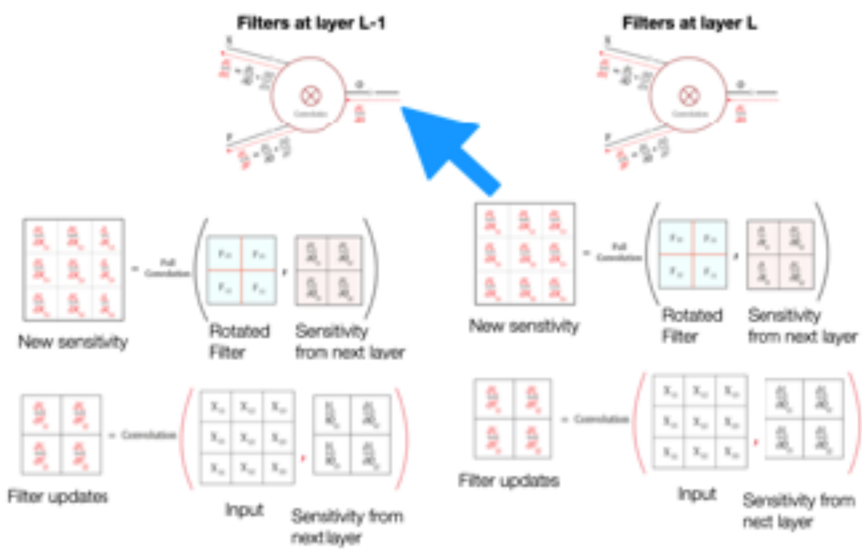
ConceptNet
Case studies

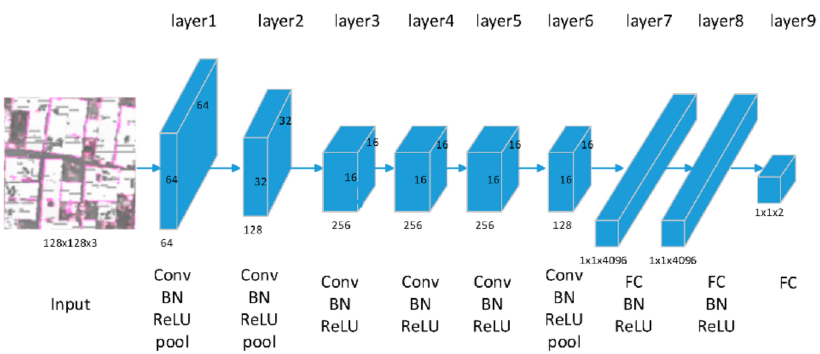
Last Time:



Structure of Each Tensor: Channels x Rows x Columns







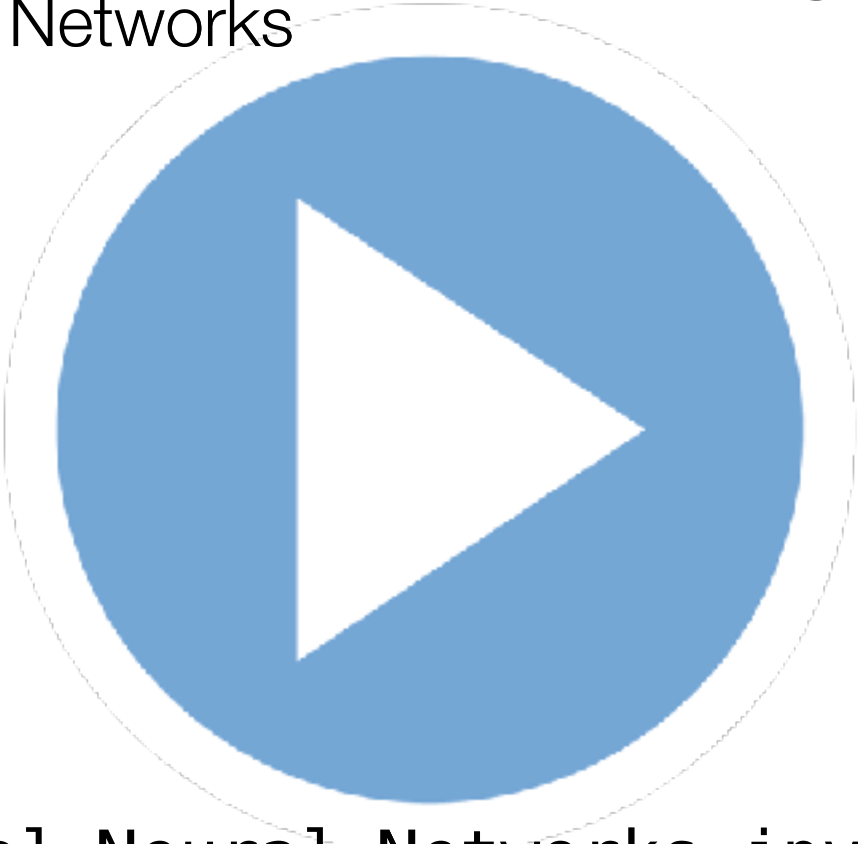
Last Time!

If needed:

Finish Demo

Convolutional Neural Networks
in TensorFlow
with Keras

with Sequential API!



`11. Convolutional Neural Networks.ipynb`

History of Convolutional Neural Networks

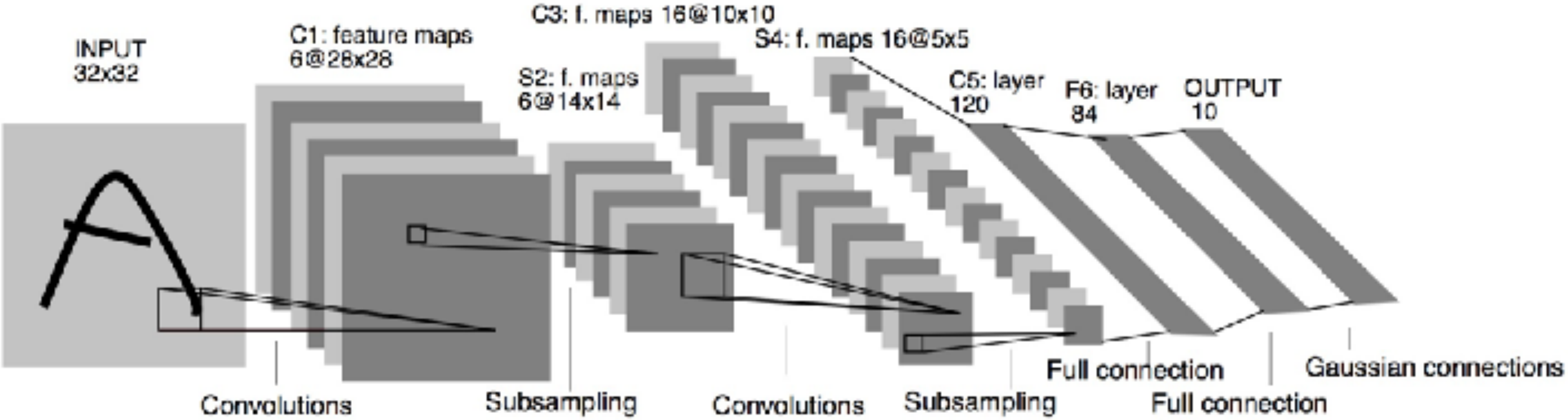
Thanks to machine-learning algorithms, the robot apocalypse was short-lived.

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**



CNN History

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

¯_(ツ)_/¯

- 2010

Types of CNN, 2010

AI Researcher
IDSA, Switzerland

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

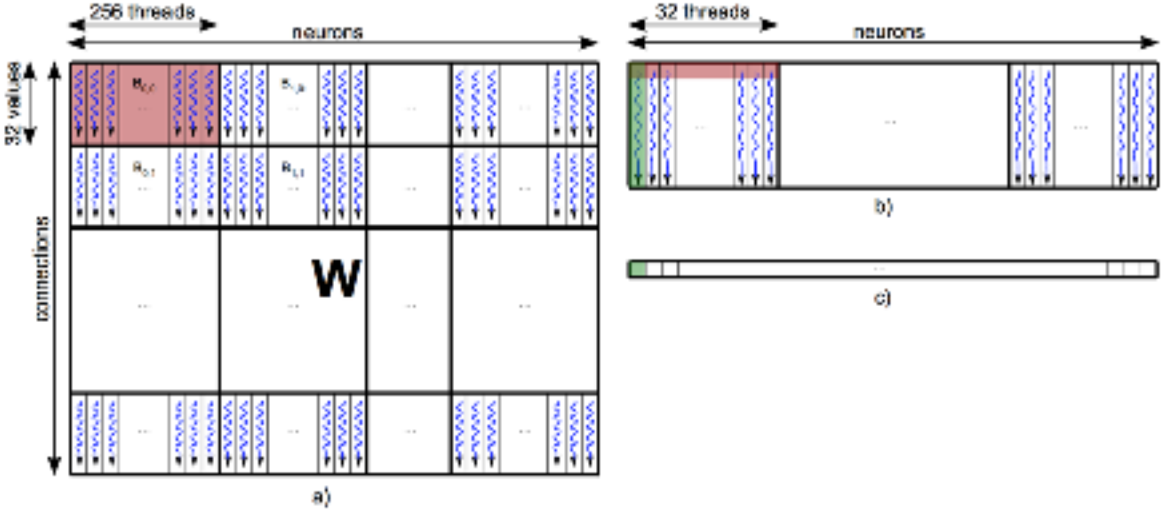
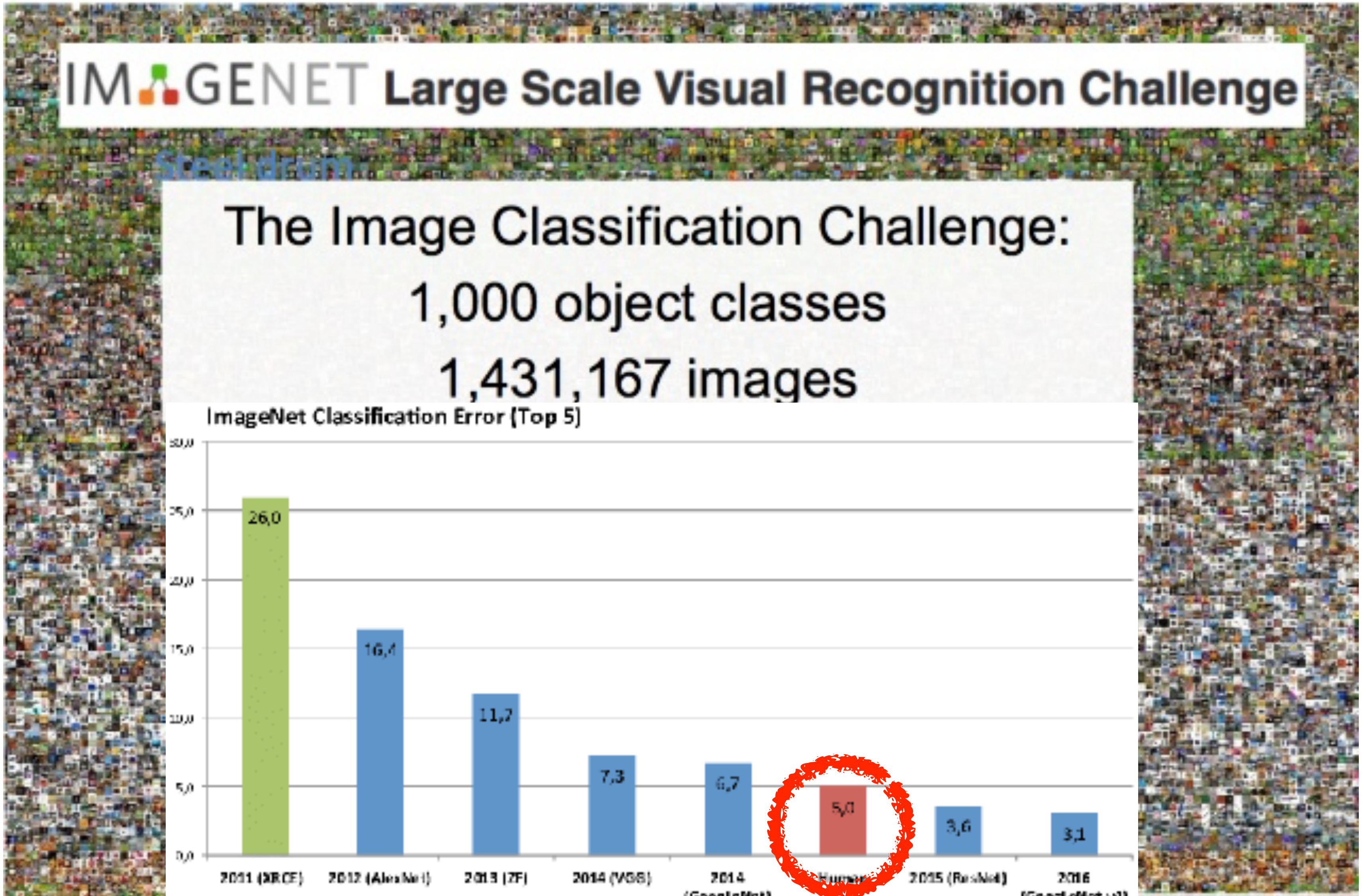


Figure 2: Forward propagation: a) mapping of kernel 1 grid onto the padded weight matrix; b) mapping the kernel 2 grid onto the partial dot products matrix; c) output of forward propagation.

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

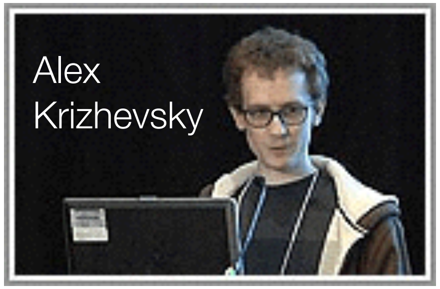
ImageNet Competition (2010-2016)



https://www.researchgate.net/figure/Winner-results-of-the-ImageNet-large-scale-visual-recognition-challenge-LSVRC-of-the_fig7_324476862

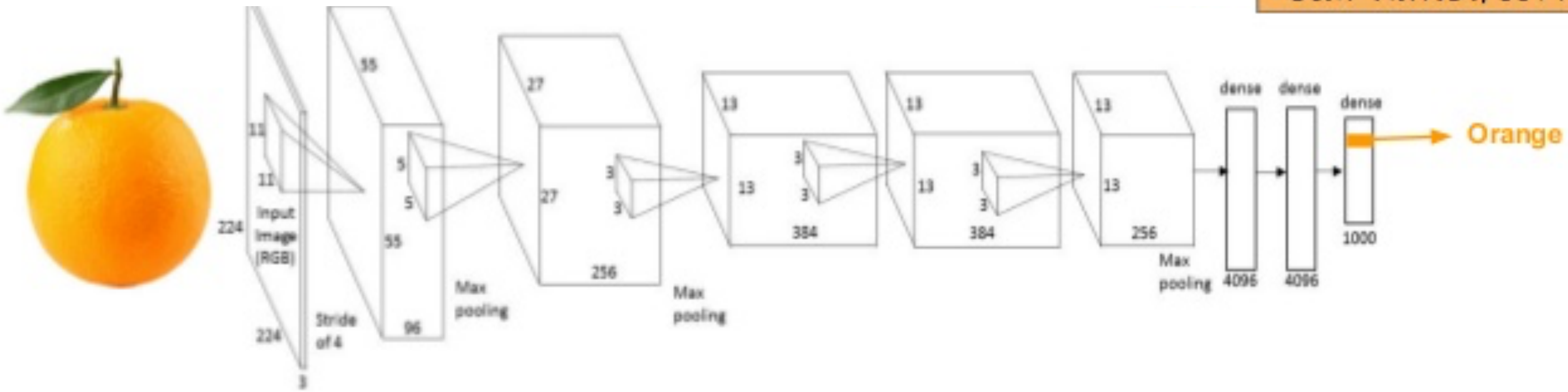
https://www.slideshare.net/nmhkahn/case-study-of-convolutional-neural-network-61556303

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***

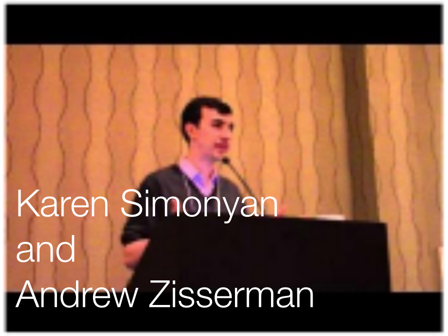


https://culurciello.github.io/tech/2016/06/04/nets.html

Warning


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

Types of CNN, 2013



- 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

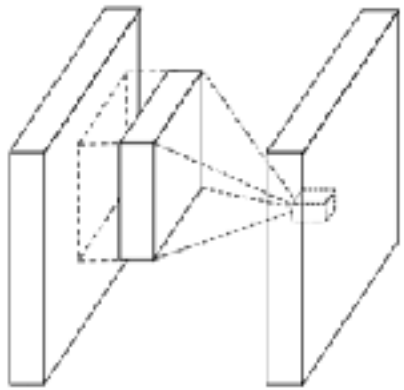
http://www.robots.ox.ac.uk/~vgg/practicals/cnn/

Types of CNN, 2014

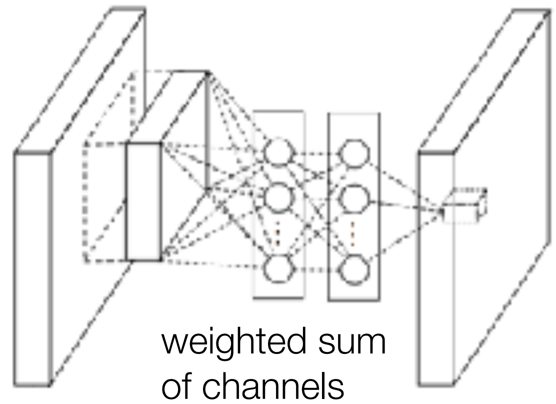
Network In Network

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

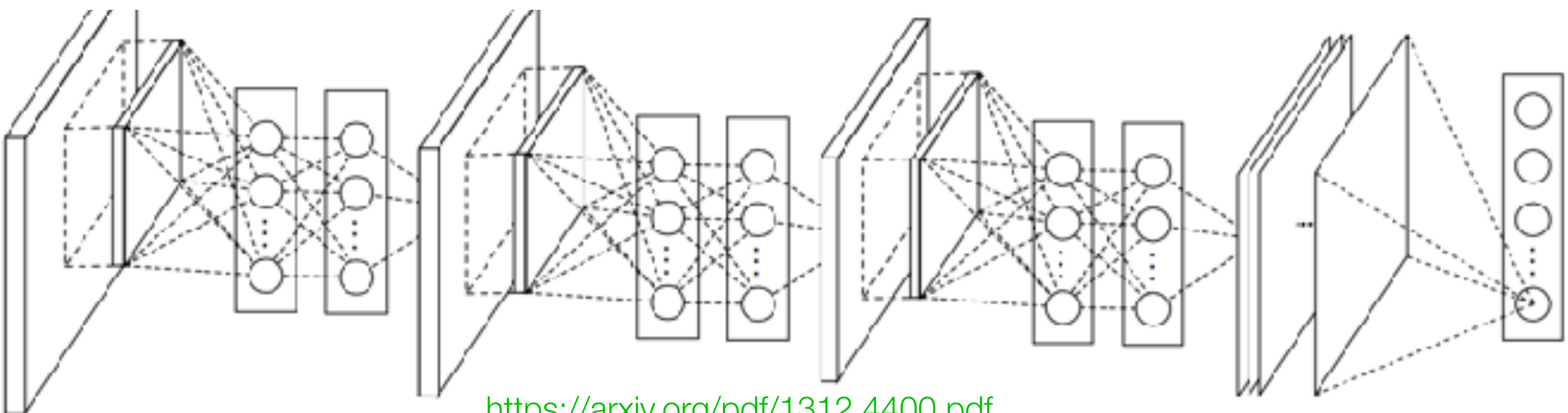
Min Lin[1,2], Qiang Chen[2], Shuicheng Yan[2]
[1]Graduate School for Integrative Sciences and Engineering
[2]Department of Electronic & Computer Engineering
National University of Singapore, Singapore
{linmin,chenqiang,eleyans}@nus.edu.sg



(a) Linear convolution layer



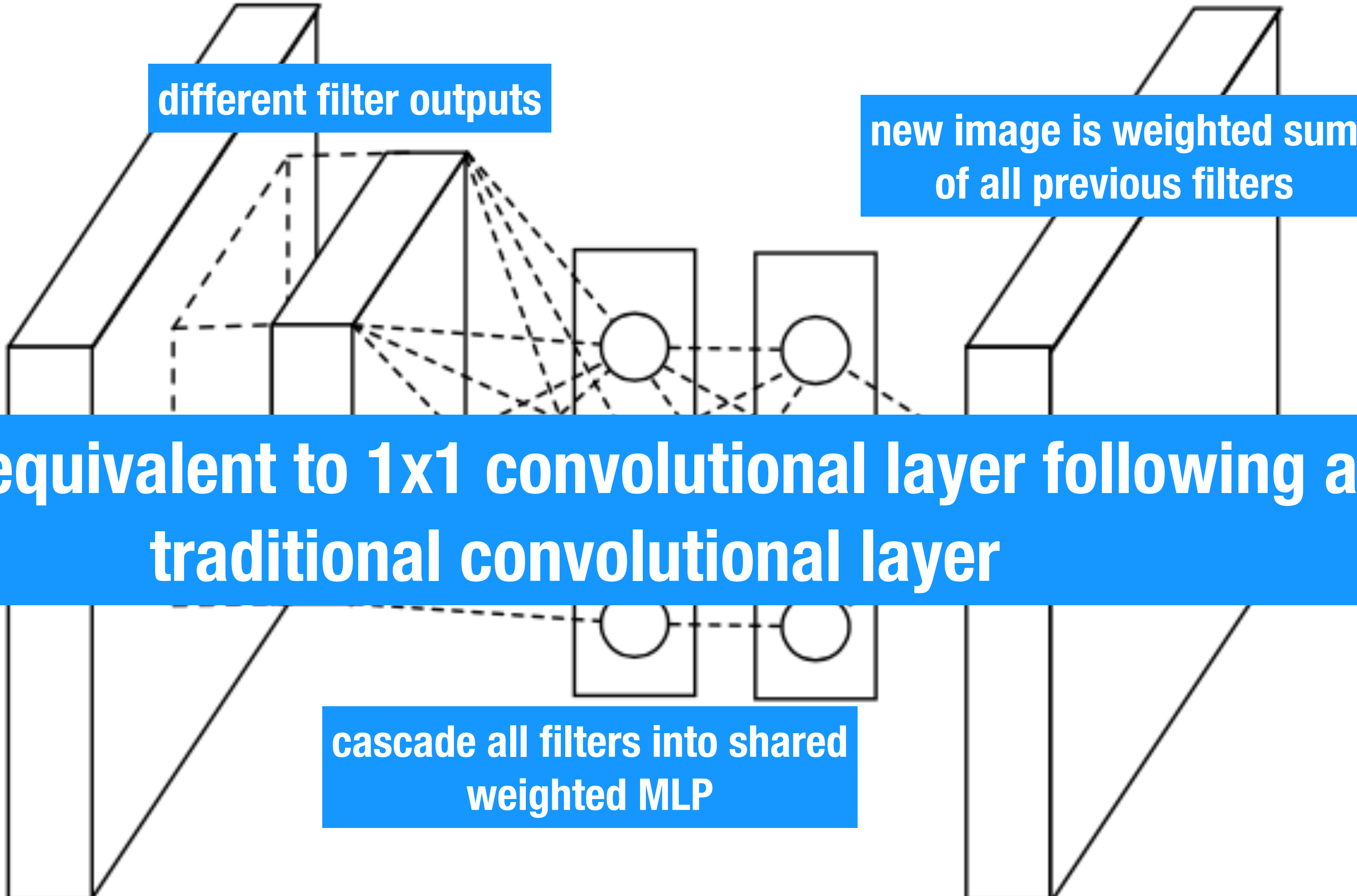
(b) Mlpconv layer



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

Types of CNN, 2014

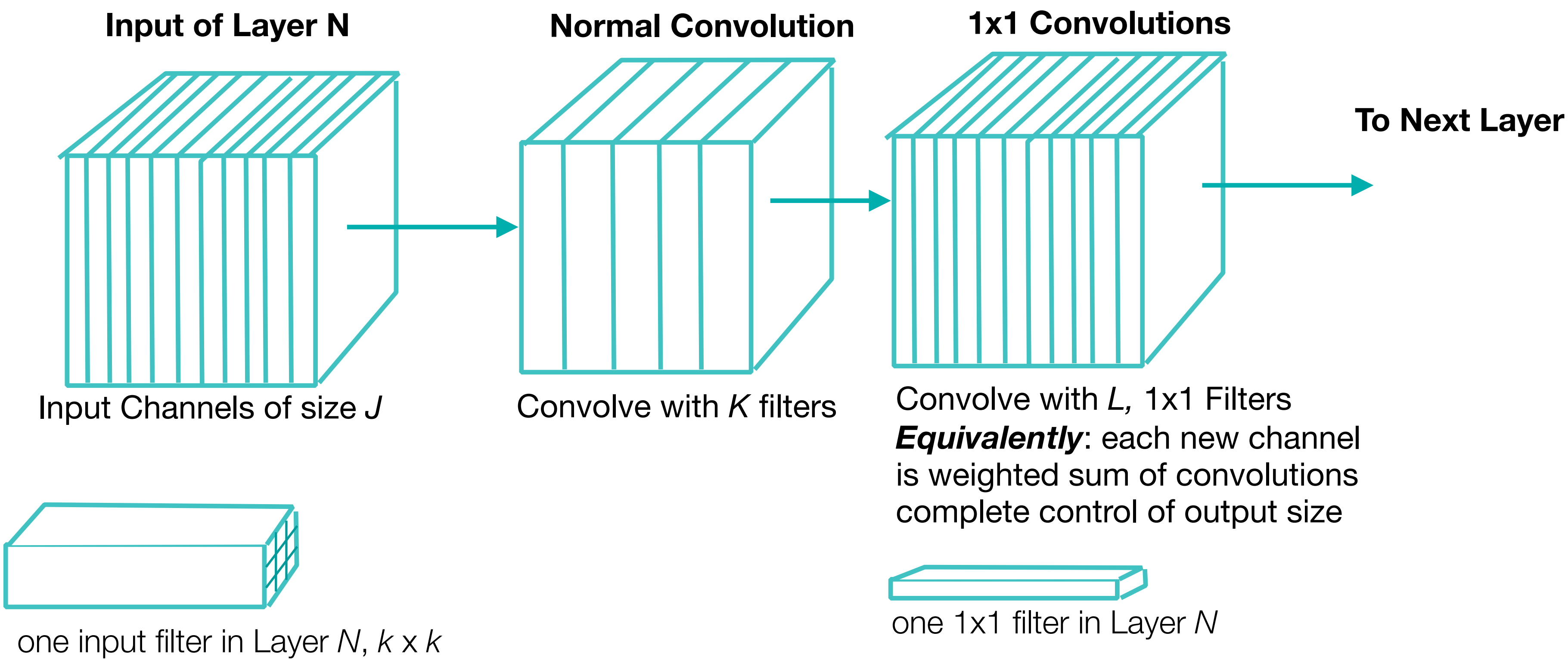
- Network in Network



this is equivalent to 1x1 convolutional layer following a traditional convolutional layer

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

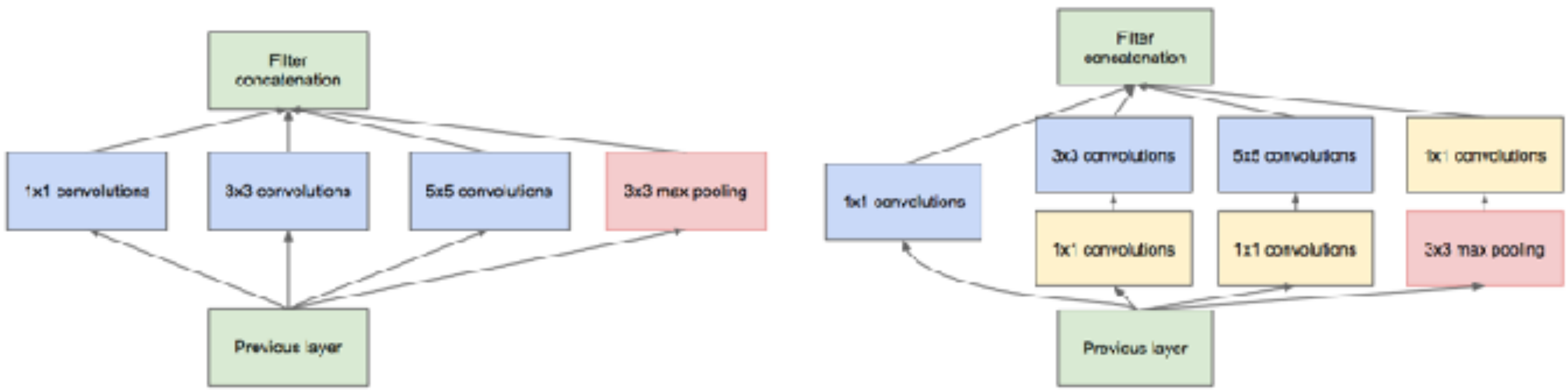
NiN, expanded view



Structure of Each Tensor: Channels x Rows x Columns

Types of CNN, 2014

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



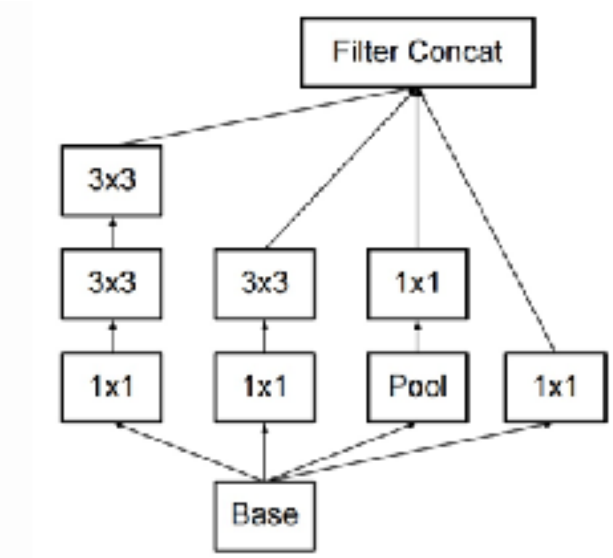
(a) Inception module, naïve version (b) Inception module with dimension reductions

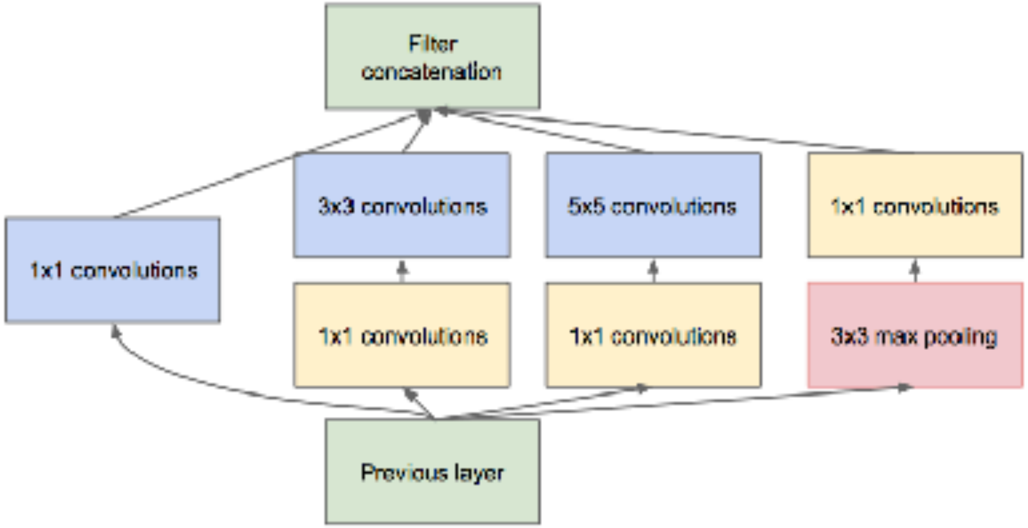
Figure 2: Inception module

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



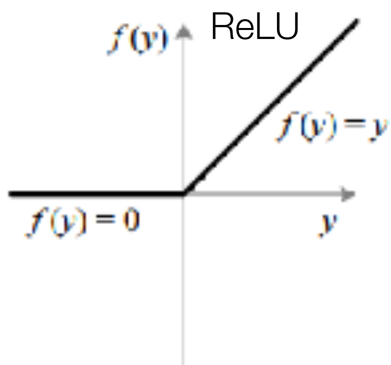


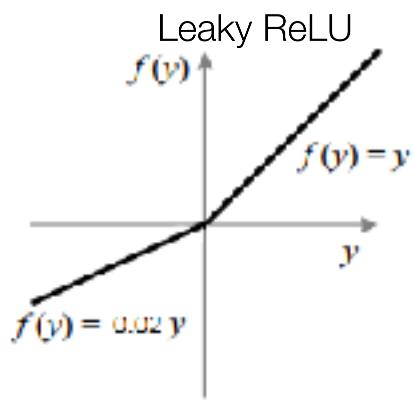
https://arxiv.org/pdf/1512.00567.pdf

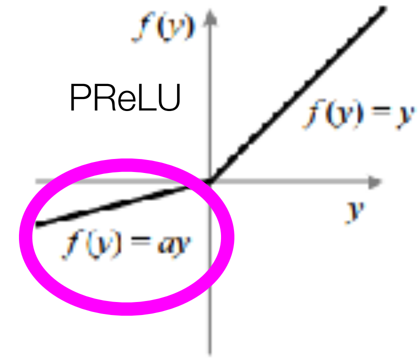
Types of CNN, 2015 December

Microsoft Research

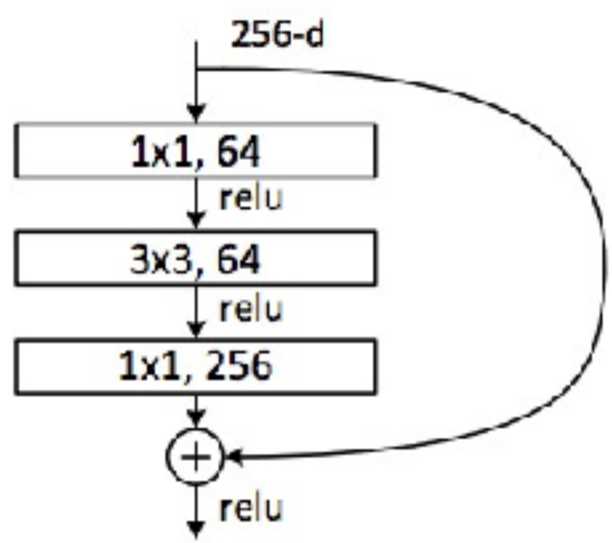
- Major Contributions:
  - "ensembles" not strictly sequential
  - "bio-plausible" with feedback
- **ResNet**
  - Parametric ReLU
    PReLU: adaptive trained slope



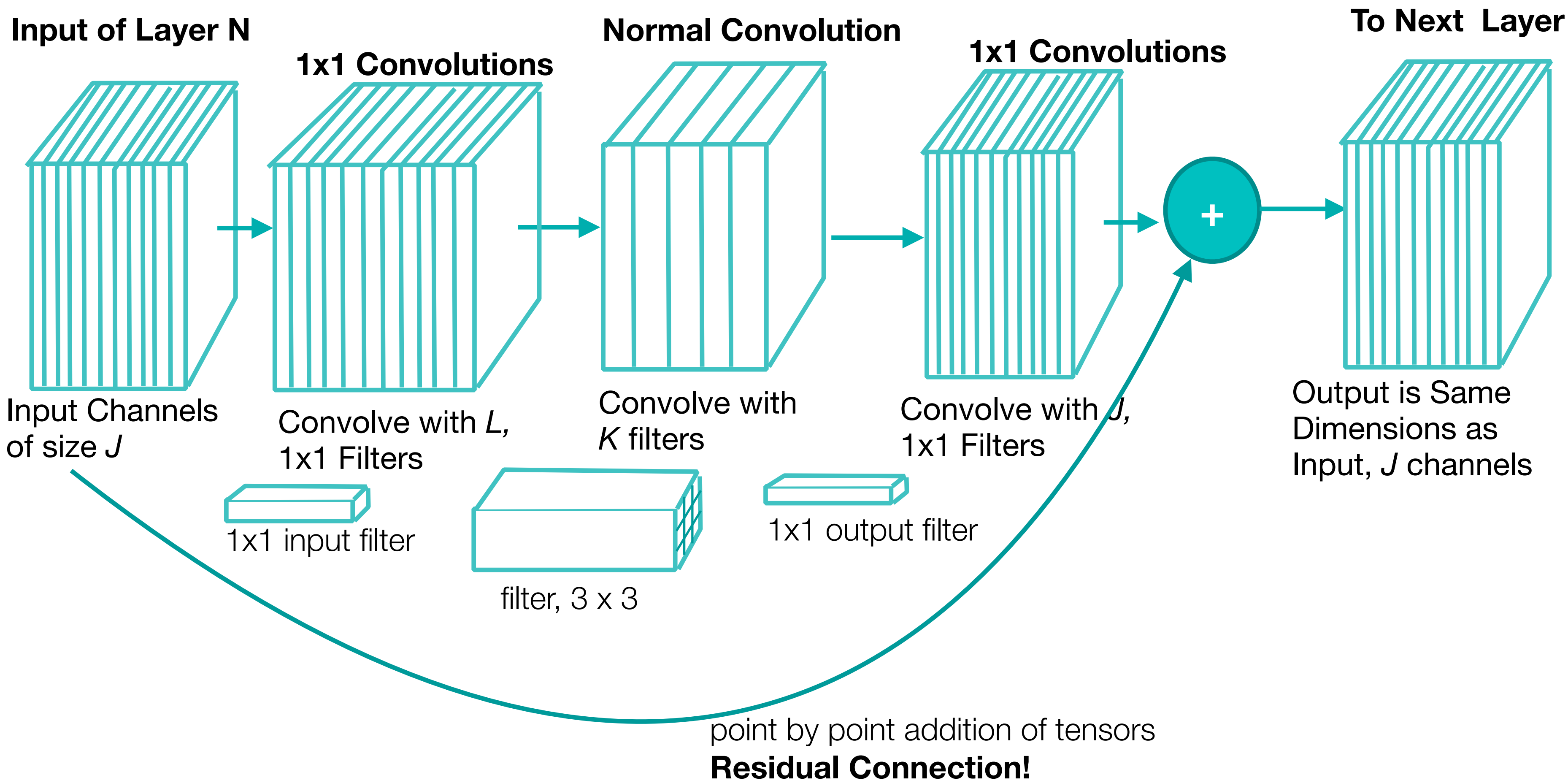




- NiN: triple bypass layer
  - similar to bottelneck



Residual Connection, expanded view



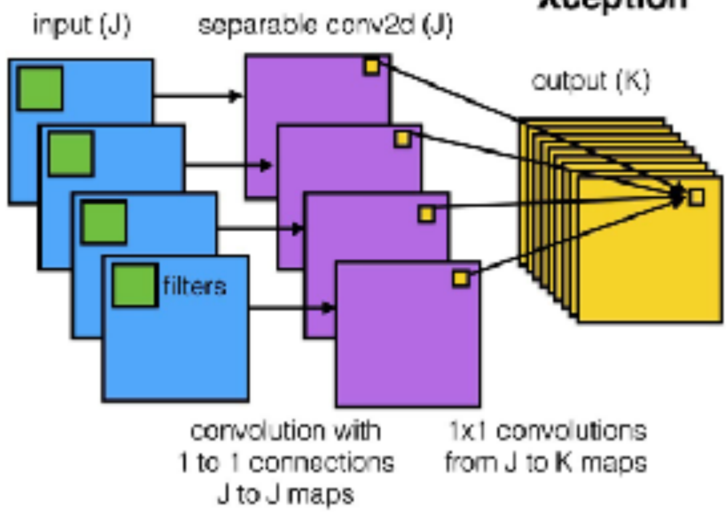
Back Propagation: Two paths, including one without ANY operations that cause the gradient to vanish…

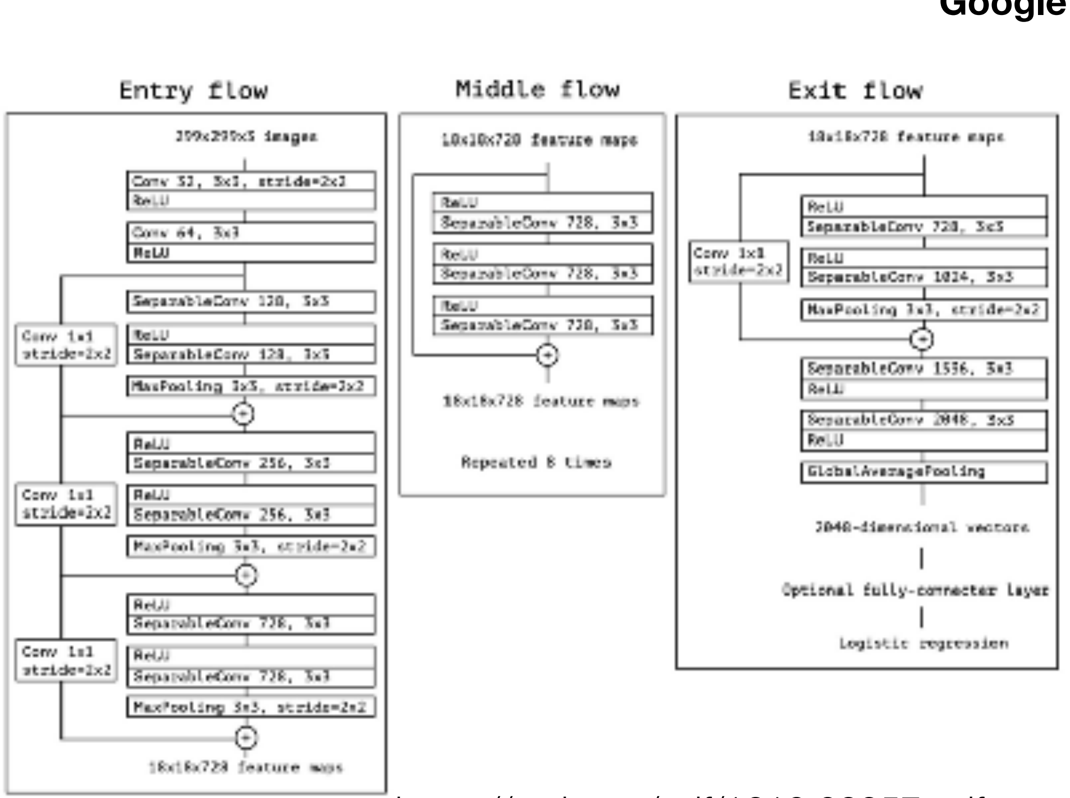
Types of CNN, 2017

Francois Chollet
Google

Xception

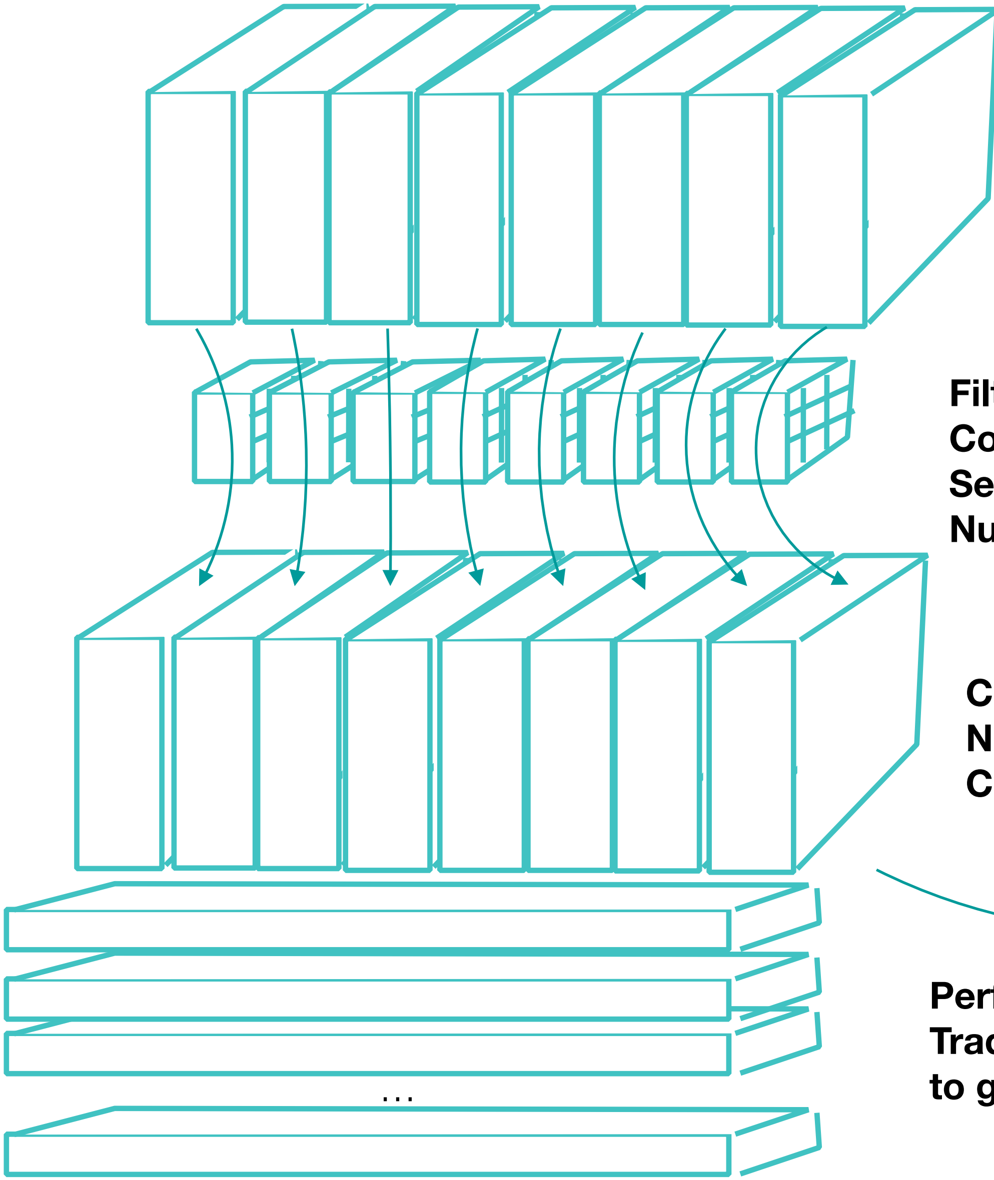
- Major Contributions:
    - combining branching / residual blocks
    - separable convolutions





https://arxiv.org/pdf/1610.02357.pdf

Separable Convolution Primer

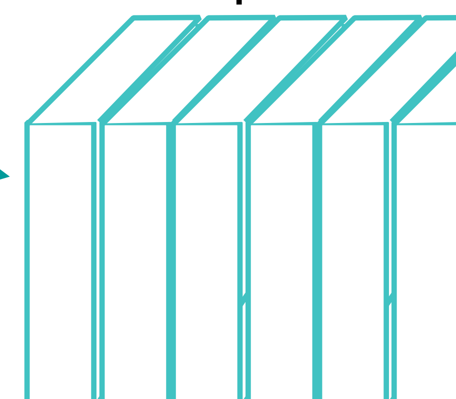


Inputs, From Layer N-1
Num Channels = J

Filters, Layer N
Convolve Each Channel Separately
Num filters = J

Concat Channel Outputs
Num Intermediate Channels = J

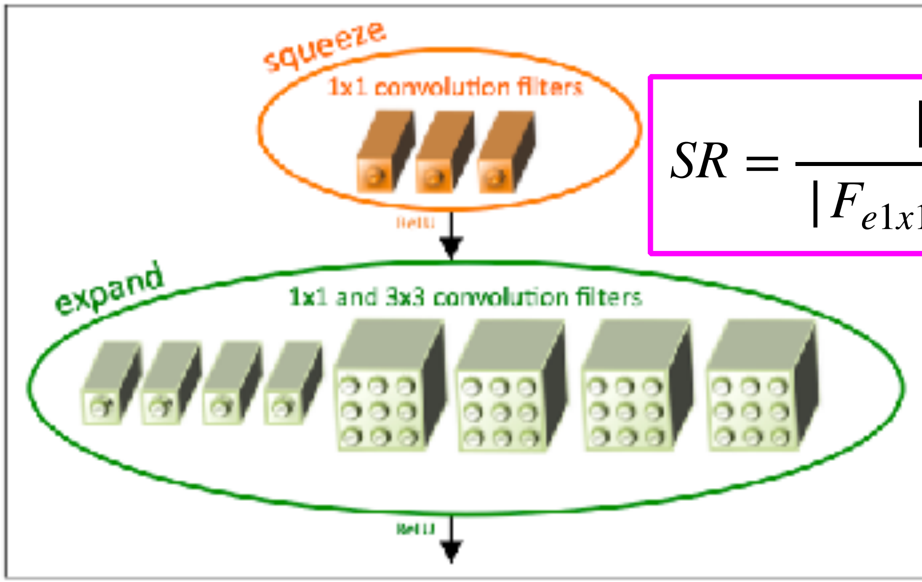
K Outputs



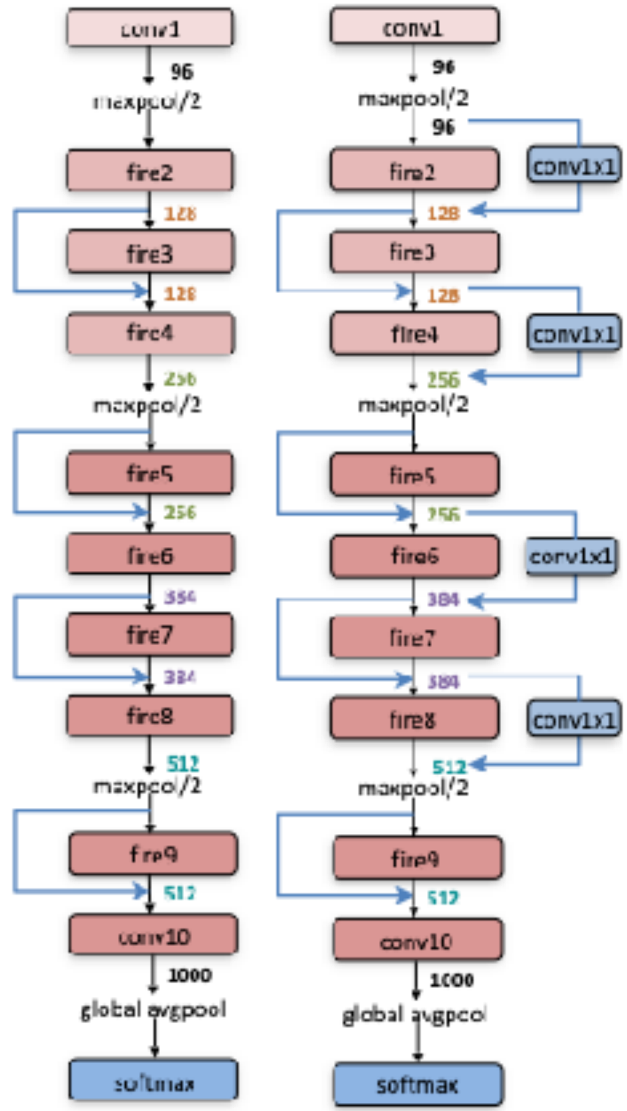
Perform K 1x1 Traditional Convolutions to get K Outputs

SqueezeNet (2018)

- Idea: squeeze and expand in each layer
  - Use mostly 1x1 filters,
  - downsample later in network,
  - reduce channels before 3x3 filters



$$SR = \frac{|F_{s1x1}|}{|F_{e1x1}| + |F_{e3x3}|}$$



SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE

Forrest N. Iandola[1], Song Han[2], Matthew W. Moskewicz[1], Khalid Ashraf[1], William J. Dally[2], Kurt Keutzer[1]
[1]DeepScale* & UC Berkeley [2]Stanford University
{forresti, moskewcz, kashraf, keutzer}@eecs.berkeley.edu
{songhan, dally}@stanford.edu

Efficient Net (2019)

depth: $d = \alpha^{\phi}$

- Define
scaling
betwe

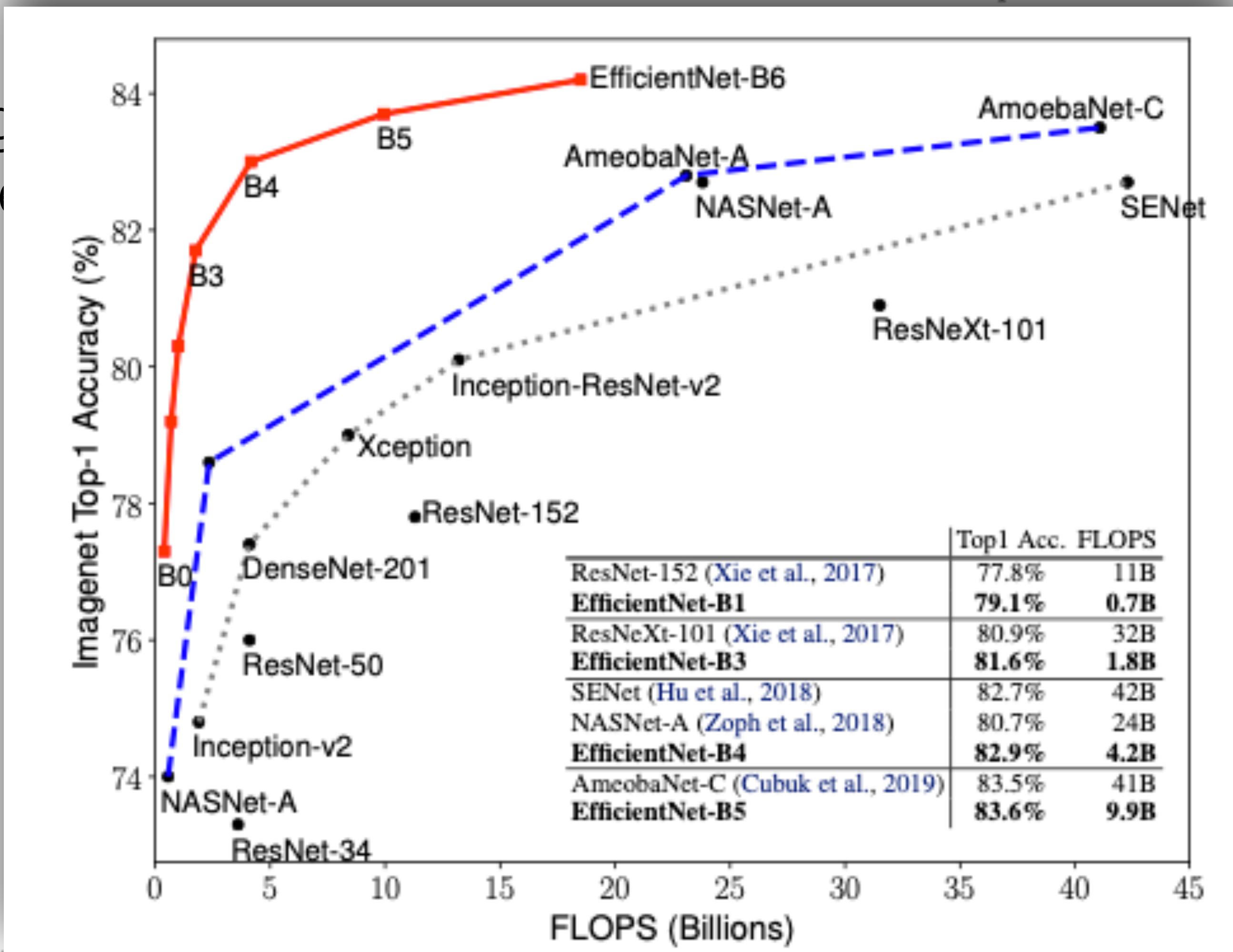
2

$\gamma \geq 1$

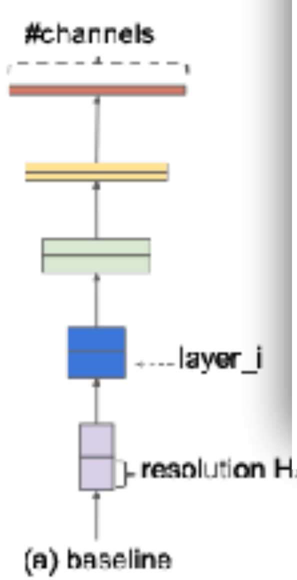
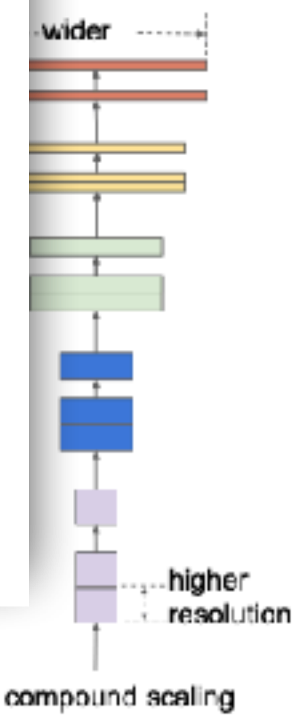
cular, we find

$\alpha = 1.2, \beta =$

$^2 \cdot \gamma^2 \approx 2.$



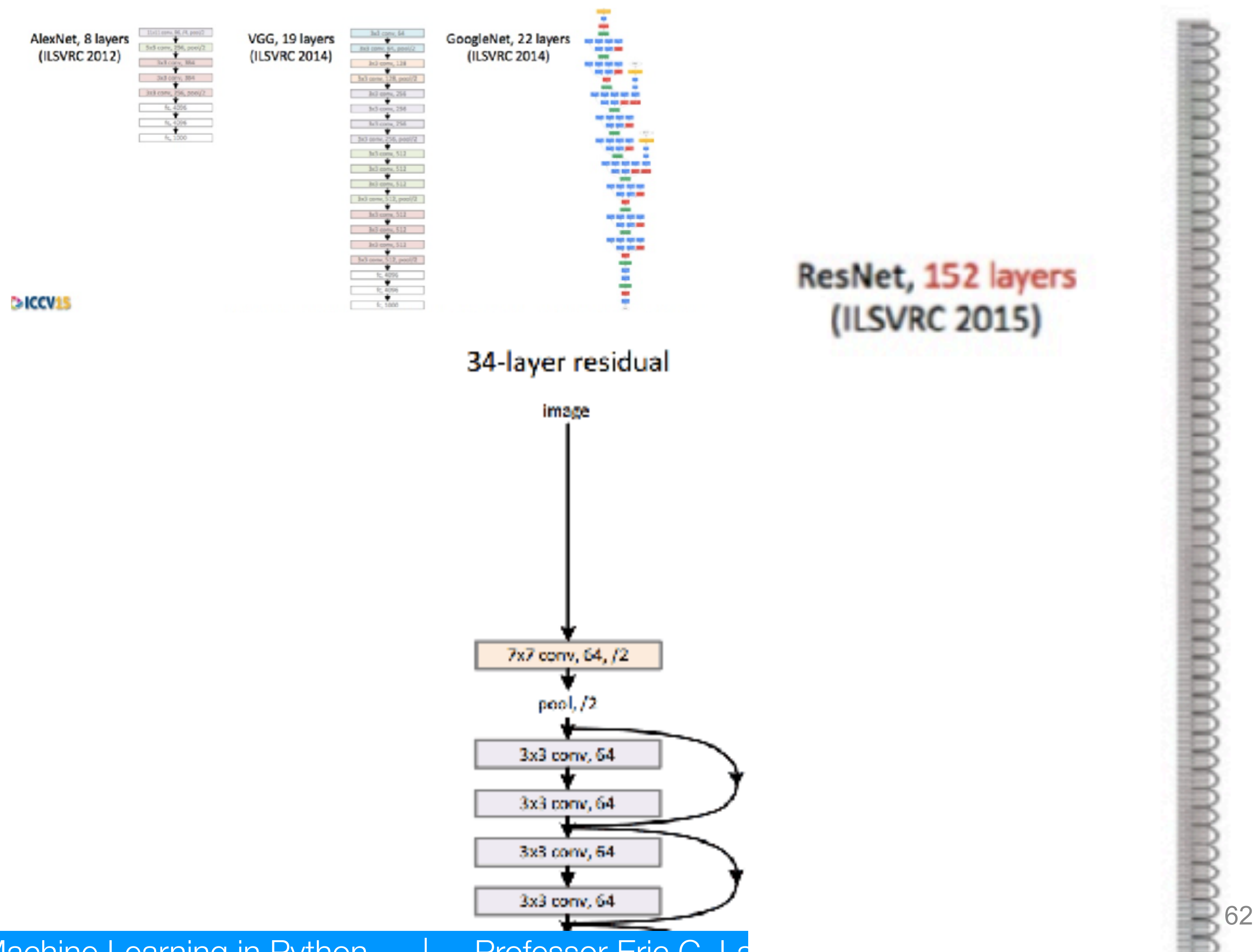
	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.1%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.6%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.9%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.6%**	**9.9B**

channels layers pooling

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