RECOD
reasoning for complex data

CNN Architectures

Machine Learning

(Largely based on slides from Fei-Fei Li & Justin Johnson & Serena Yeung)

Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

MC886, October 16, 2019

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - ResNet (2015)

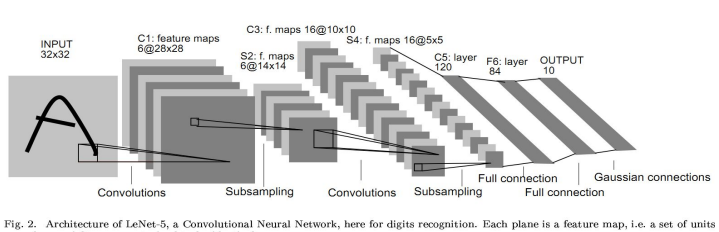
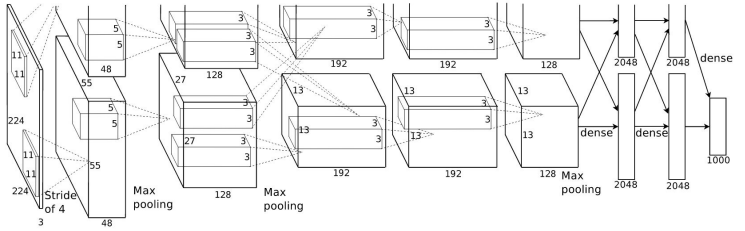
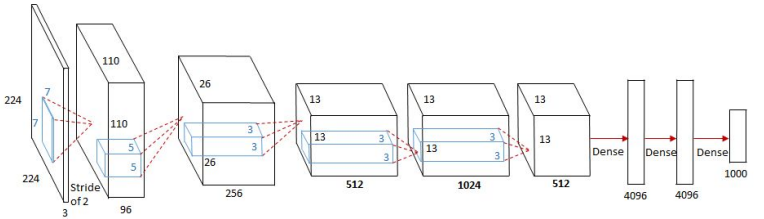
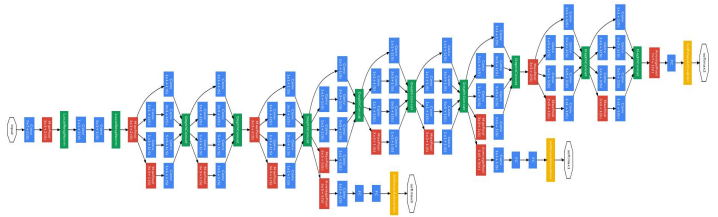
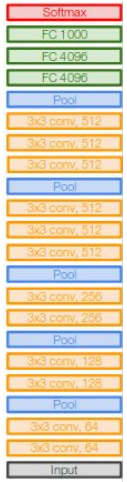


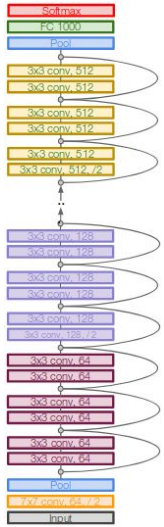
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.











Today's Agenda

- CNN Architectures
  - **LeNet (1998)**
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - ResNet (2015)

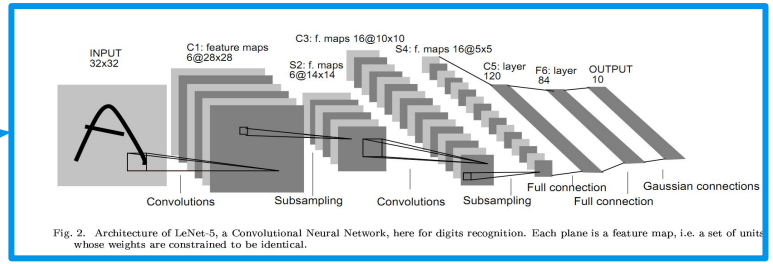
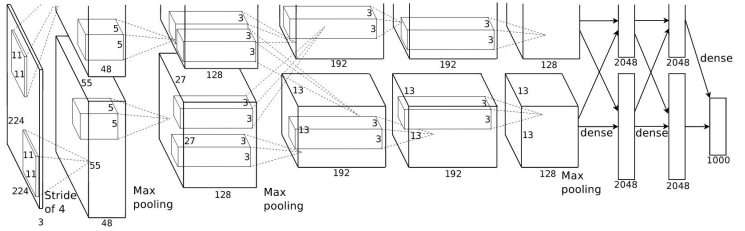
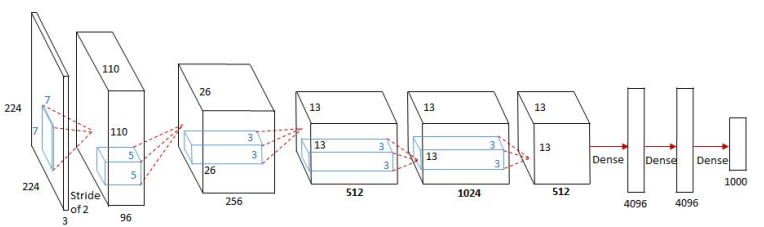
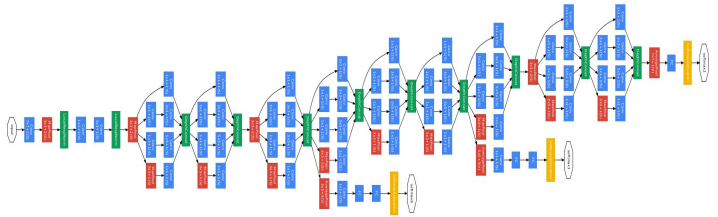
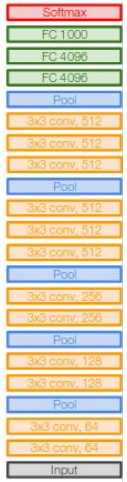


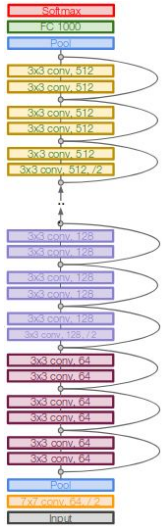
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.



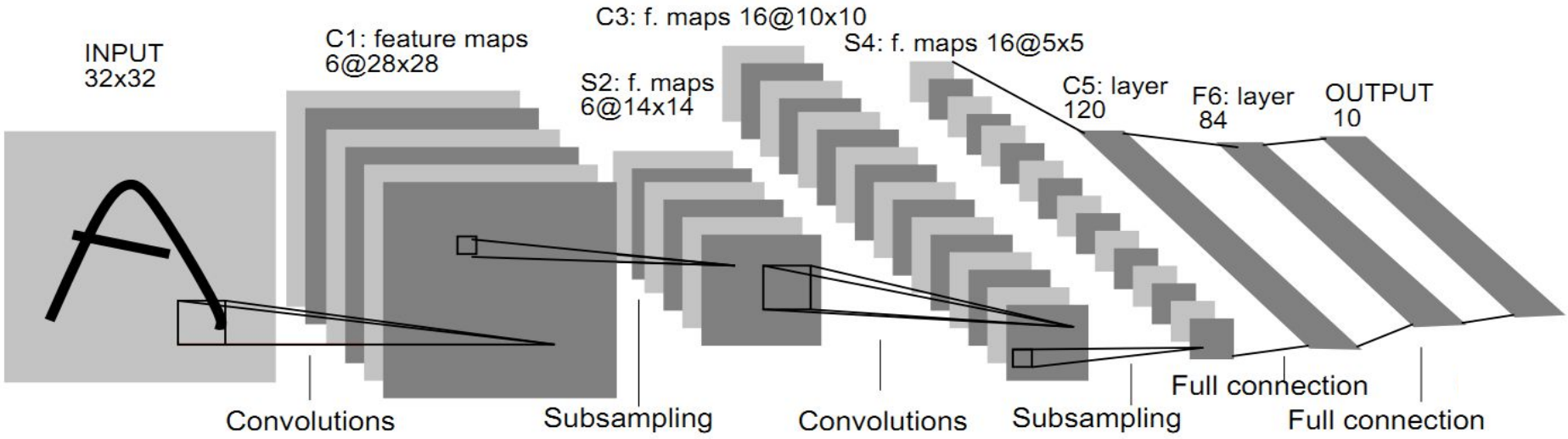








LeNet-5 [LeCun et al., 1998]



Convolution filters: 5x5 with stride 1

Subsampling (Pooling) layers: 2x2 with stride 2

[CONV-POOL-CONV-POOL-FC-FC]

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - **AlexNet (2012)**
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - ResNet (2015)

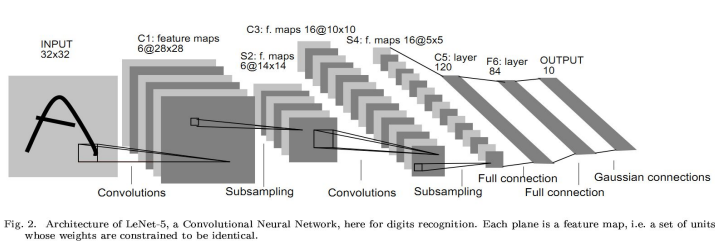
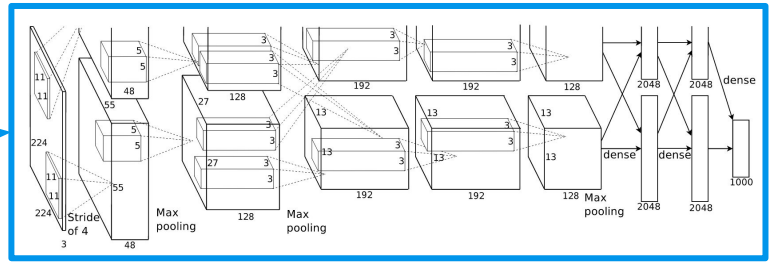
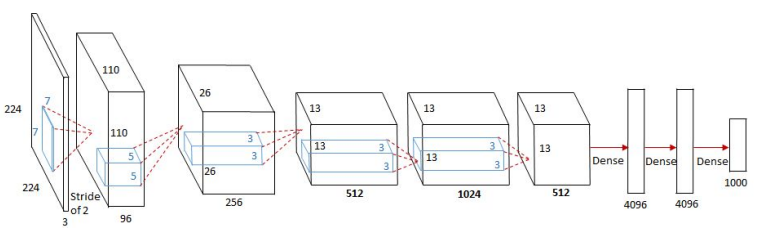
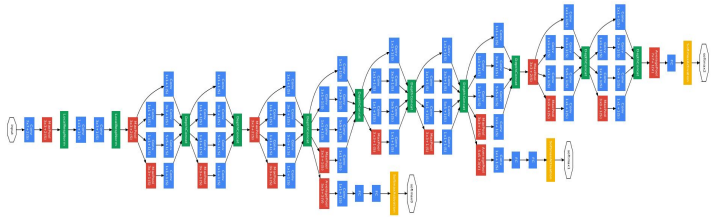
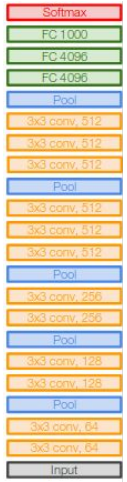


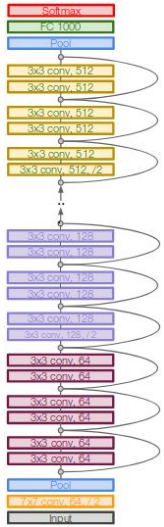
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

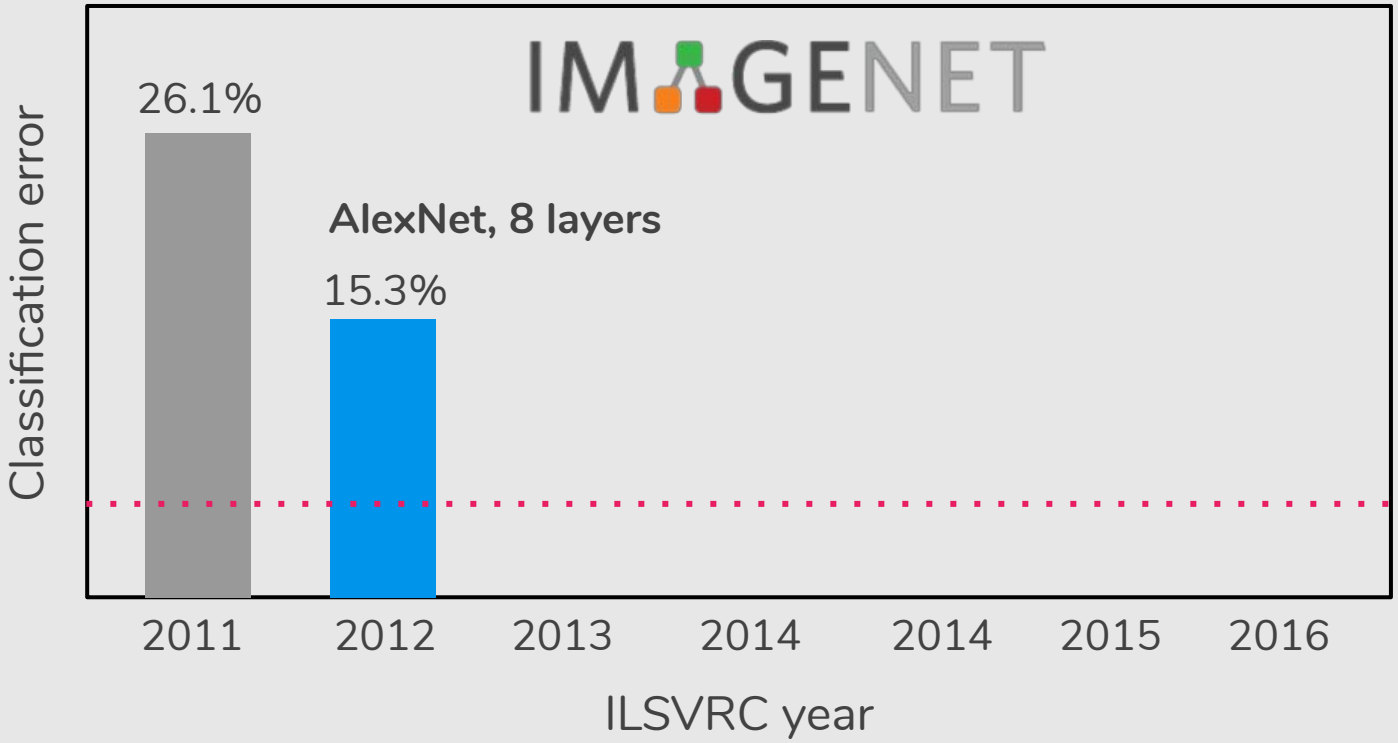






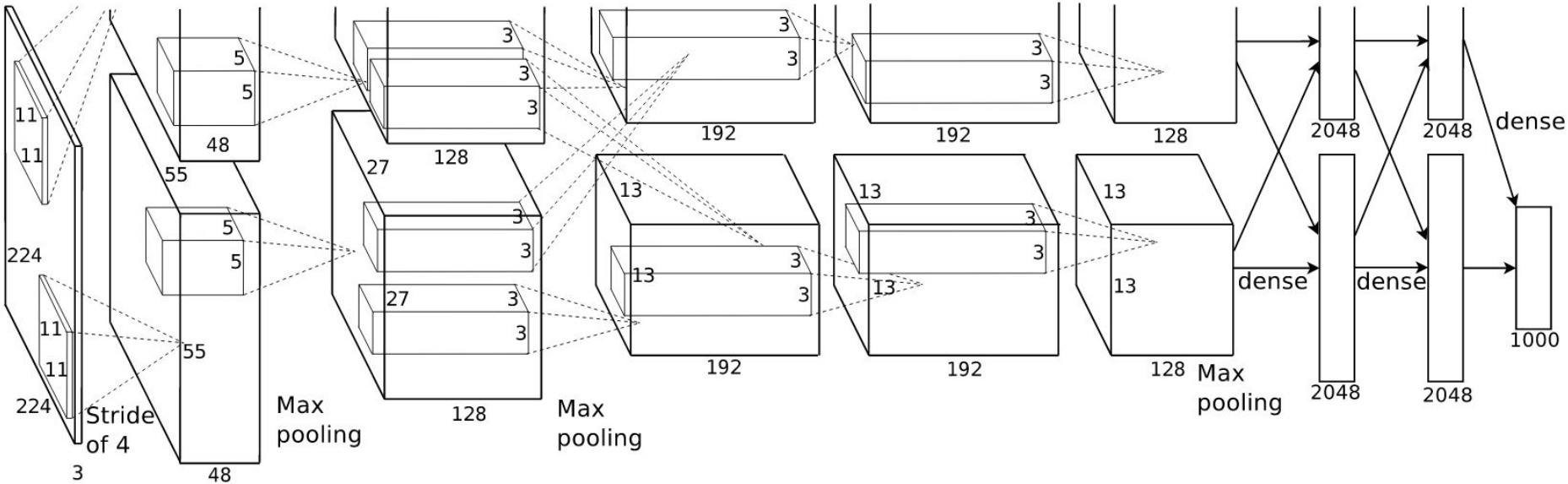






"ImageNet classification with deep convolutional neural networks". NIPS, 2012.

AlexNet [Krizhevsky et al., 2012]

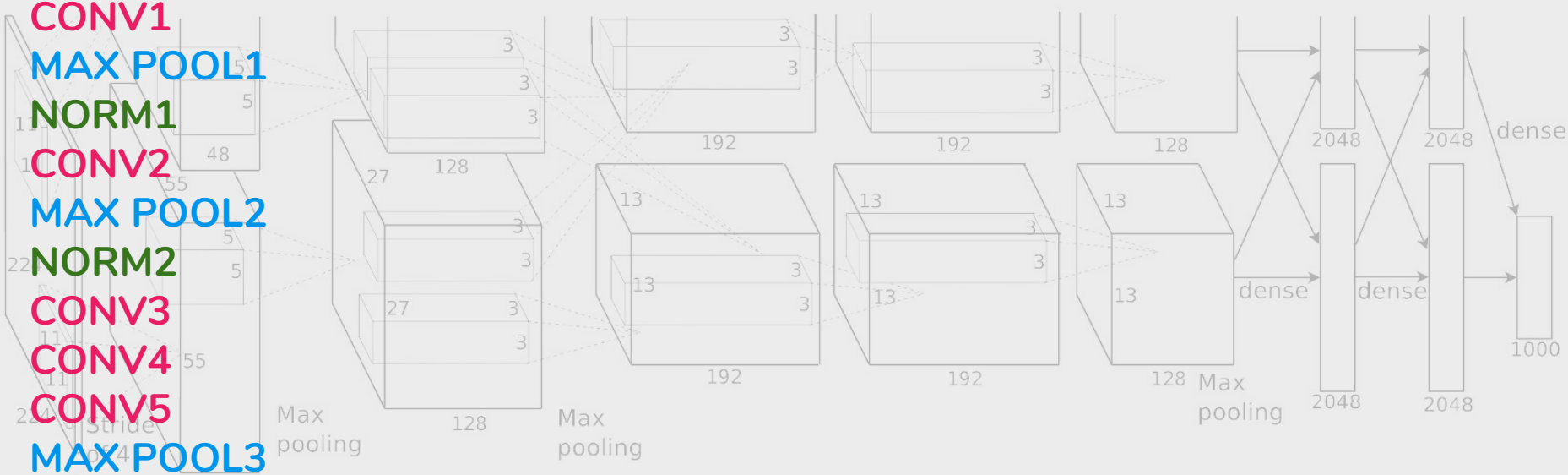


"ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.

AlexNet [Krizhevsky et al., 2012]

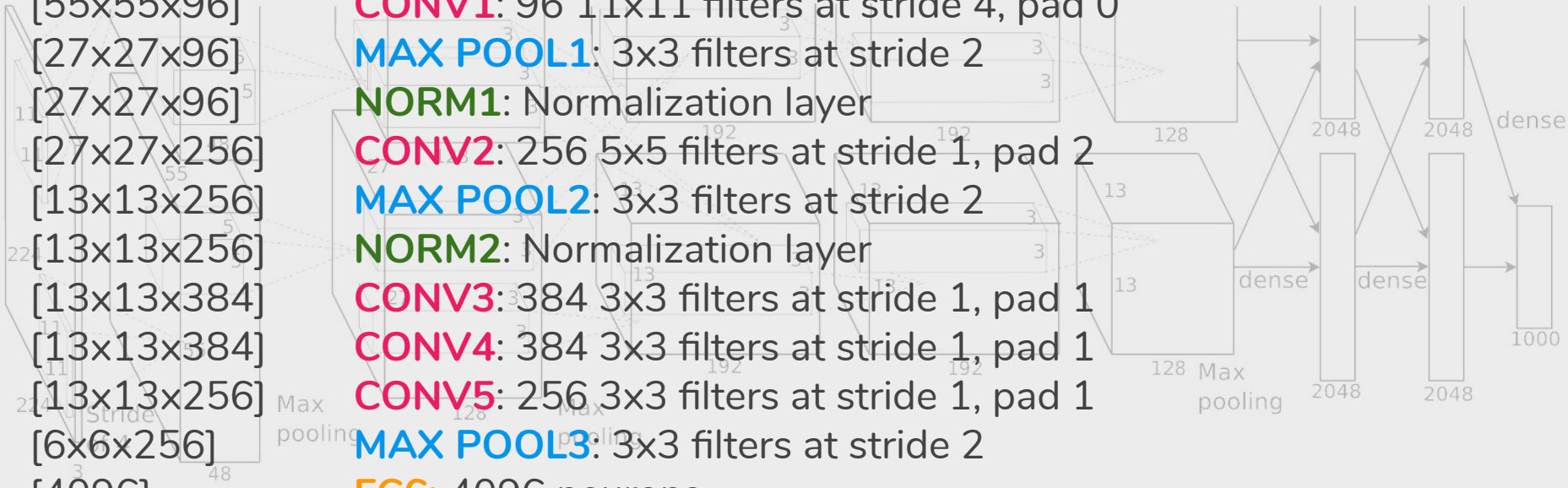
Architecture:

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
MAX POOL3
FC6
FC7
FC8

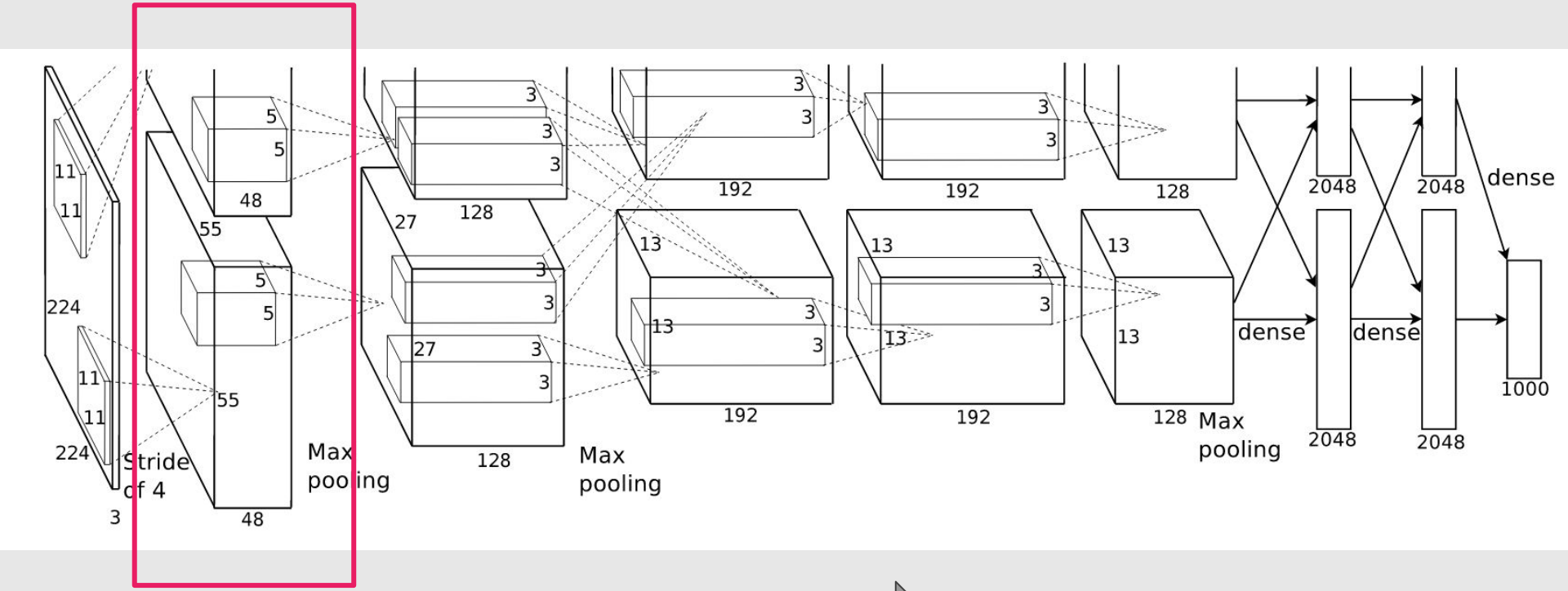


AlexNet [Krizhevsky et al., 2012]

[227x227x3]	INPUT
[55x55x96]	CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96]	MAX POOL1: 3x3 filters at stride 2
[27x27x96]	NORM1: Normalization layer
[27x27x256]	CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256]	MAX POOL2: 3x3 filters at stride 2
[13x13x256]	NORM2: Normalization layer
[13x13x384]	CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384]	CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256]	CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256]	MAX POOL3: 3x3 filters at stride 2
[4096]	FC6: 4096 neurons
[4096]	FC7: 4096 neurons
[1000]	FC8: 1000 neurons (class scores)

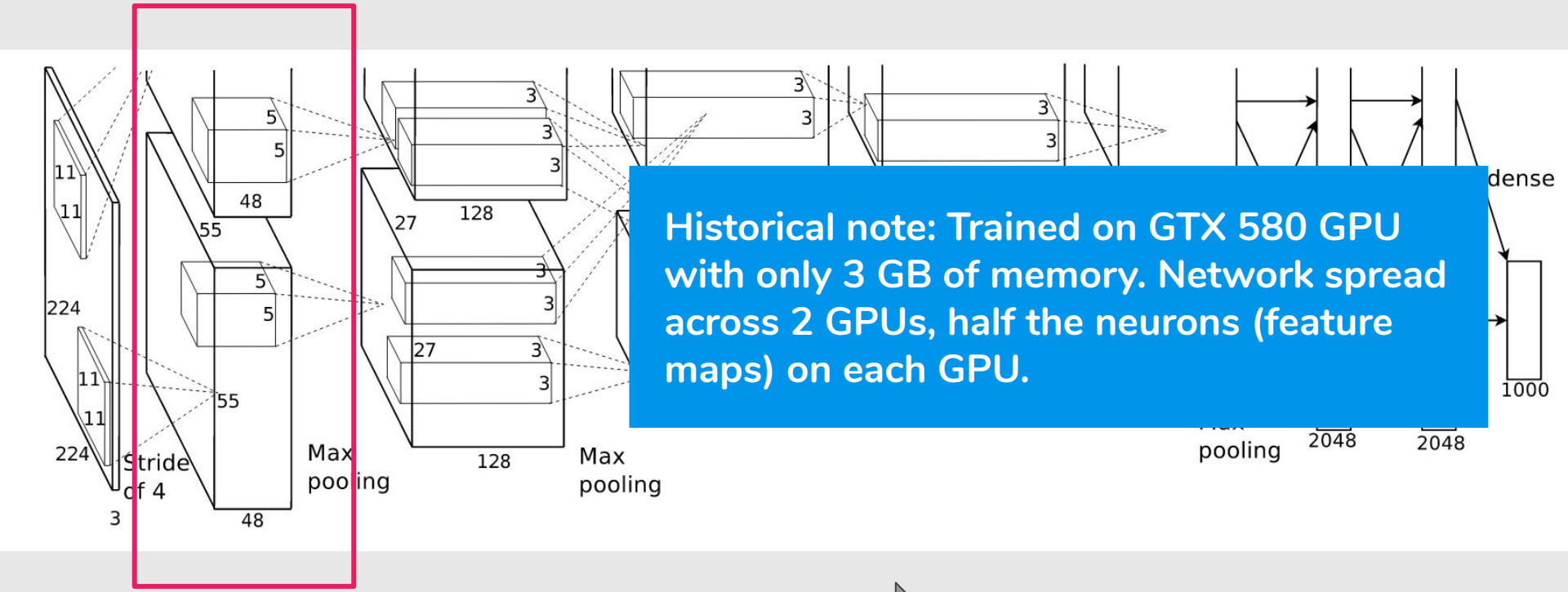


AlexNet [Krizhevsky et al., 2012]



[55x55x96] **CONV1** ➡ [55x55x48] x 2

AlexNet [Krizhevsky et al., 2012]



Historical note: Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

[55x55x96] **CONV1** ➡ [55x55x48] x 2

AlexNet [Krizhevsky et al., 2012]

Details:

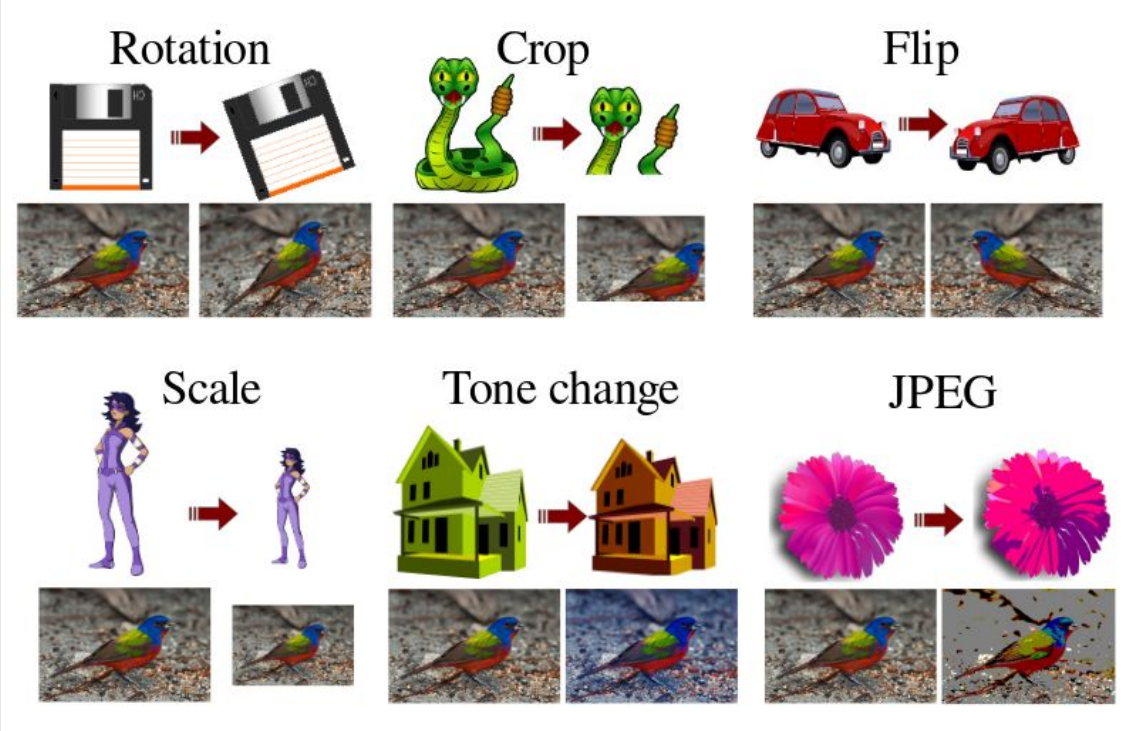
- 60 million learned parameters
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- 7 CNN ensemble: 18.2% -> 15.3%
- 5-6 days to train on 2 GTX 580 3GB GPUs

AlexNet [Krizhevsky et al., 2012]

Details:

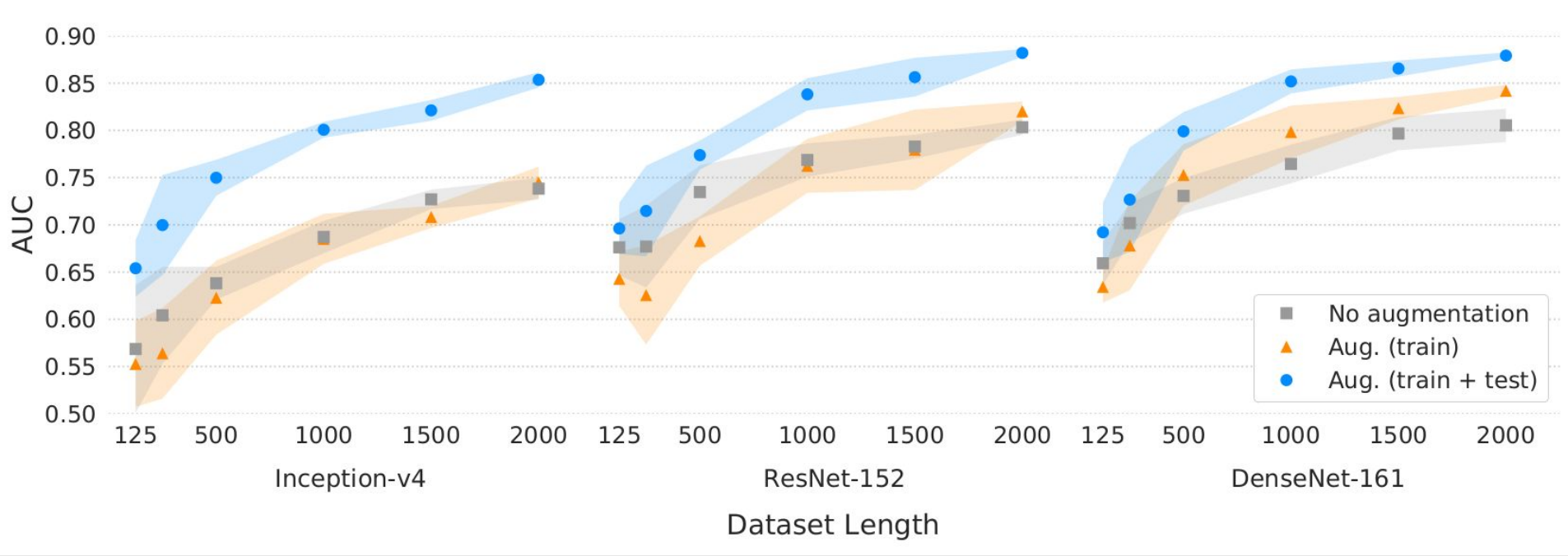
- 60 million learned parameters
- first use of ReLU
- used Norm layers (not common anymore)
- heavy **data augmentation**
- dropout 0.5
- batch size 128
- 7 CNN ensemble: 18.2% -> 15.3%
- 5-6 days to train on 2 GTX 580 3GB GPUs

Data Augmentation



"Transformation Pursuit for Image Classification", CVPR 2014.

Data Augmentation (Train & Test)



"Data augmentation for skin lesion analysis", MICCAI 2018, https://arxiv.org/pdf/1809.01442

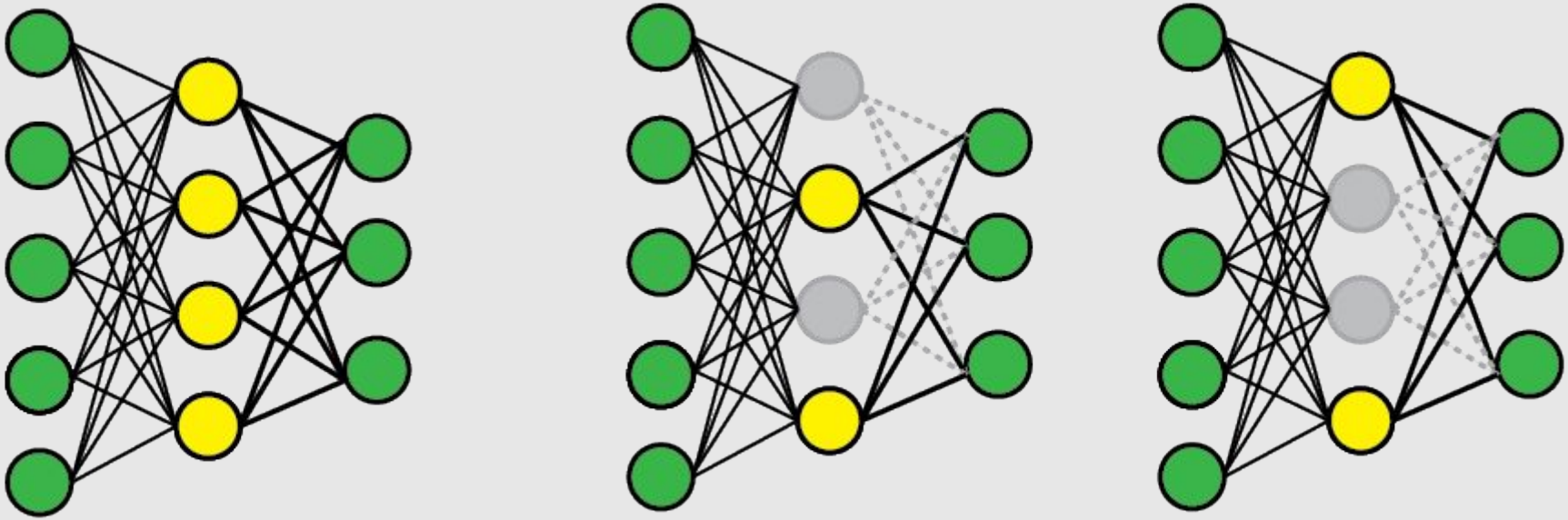
"Data, depth, and design: learning reliable models for melanoma screening", https://arxiv.org/pdf/1711.00441

AlexNet [Krizhevsky et al., 2012]

Details:

- 60 million learned parameters
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- **dropout** 0.5
- batch size 128
- 7 CNN ensemble: 18.2% -> 15.3%

Dropout [Hinton et al., 2012]



Standard Network

After applying dropout

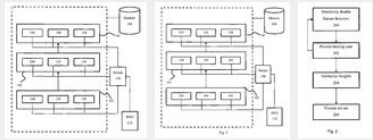
"Dropout: A Simple Way to Prevent Neural Networks from Overfitting", https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

System and method for addressing overfitting in a neural network

Abstract

A system for training a neural network. A switch is linked to feature detectors in at least some of the layers of the neural network. For each training case, the switch randomly selectively disables each of the feature detectors in accordance with a preconfigured probability. The weights from each training case are then normalized for applying the neural network to test data.

Images (3)



Classifications

G06N3/084 Back-propagation

View 4 more classifications

US9406017B2
United States

Download PDF | Find Prior Art | Similar

Inventor: Geoffrey E. Hinton, Alexander Krizhevsky, Ilya Sutskever, Nitish Srivastva

Current Assignee : Google LLC

Worldwide applications

2013 · US WO BR AU 2016 · US

Application US14/015,768 events

2012-12-24 • Priority to US201261745711P
2013-08-30 • Application filed by Google LLC
2014-06-26 • Publication of US20140180986A1
2016-08-02 • Publication of US9406017B2
2016-08-02 • Application granted
2019-10-10 • Application status is Active
2034-09-03 • Adjusted expiration

Show all events

Info: Non-patent citations (24), Cited by (13), Legal events, Similar documents, Priority and Related Applications

External links: USPTO, USPTO Assignment, Espacenet, Global Dossier, Discuss

Description

CROSS-REFERENCE TO RELATED APPLICATION

Claims (24)

Hide Dependent

What is claimed is:

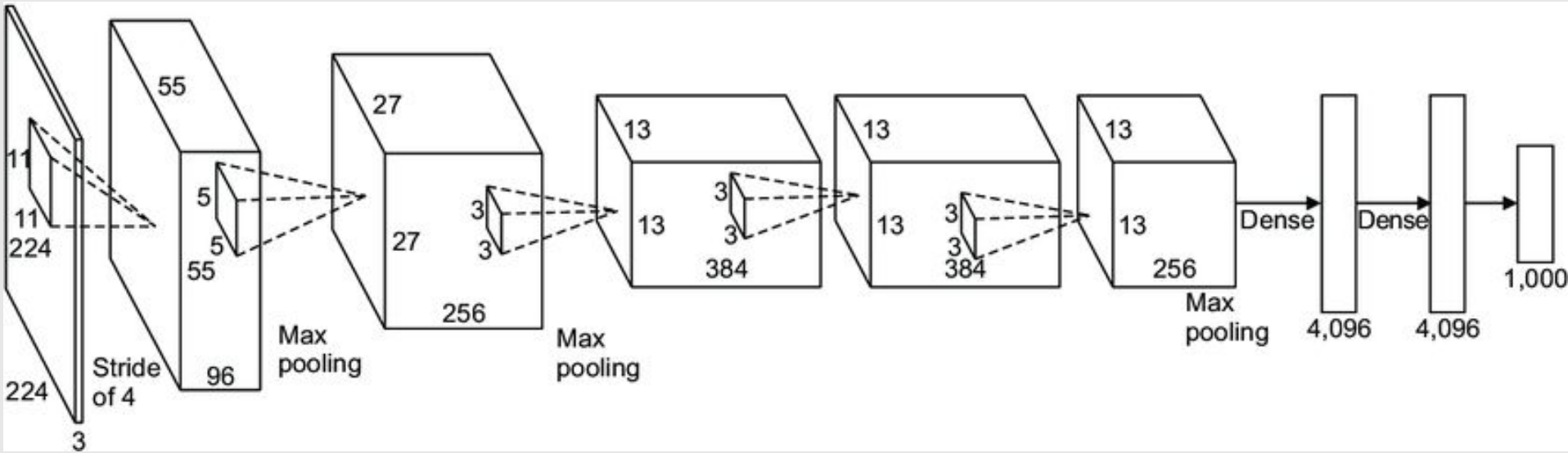
AlexNet [Krizhevsky et al., 2012]

Details:

- 60 million learned parameters
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- 7 CNN **ensemble**: 18.2% -> 15.3%

AlexNet [Krizhevsky et al., 2012]

CaffeNet = versão 1-GPU da AlexNet



Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - **ZFNet (2013)**
  - VGGNet (2014)
  - GoogLeNet (2014)
  - ResNet (2015)

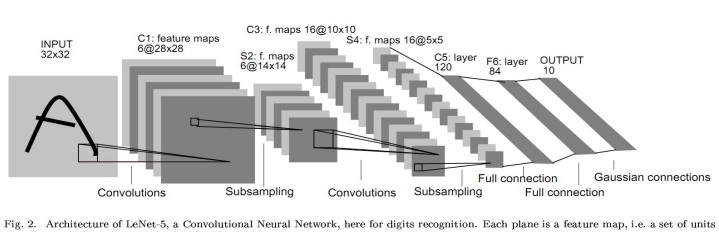
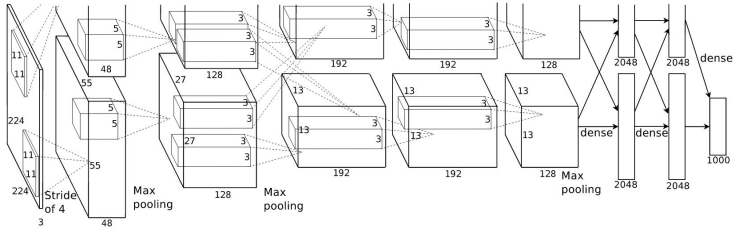
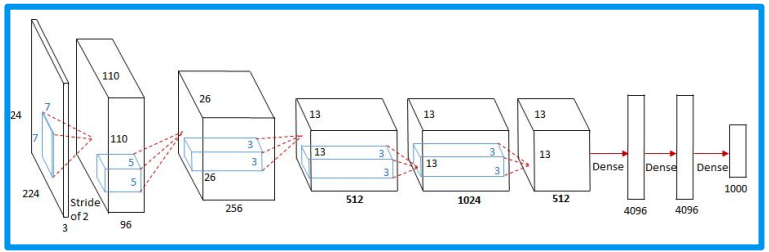
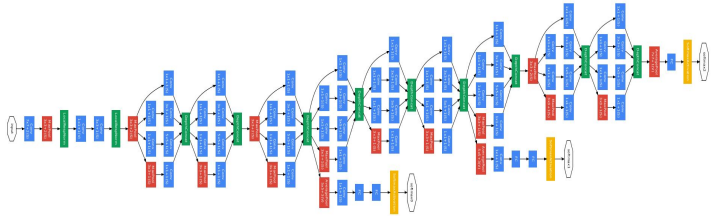
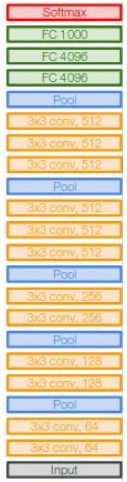


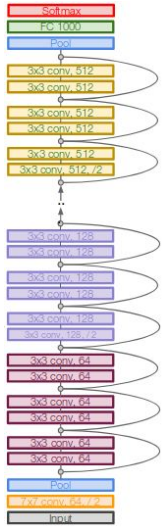
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

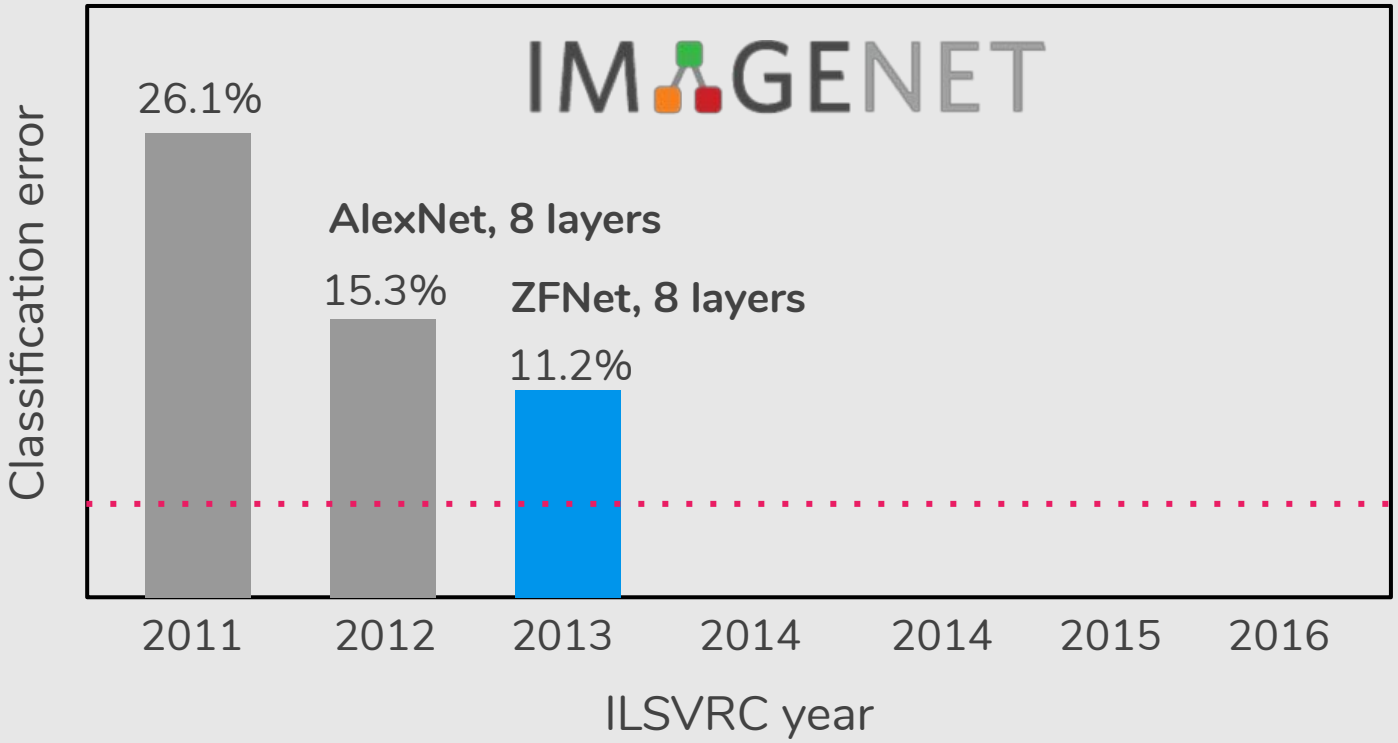






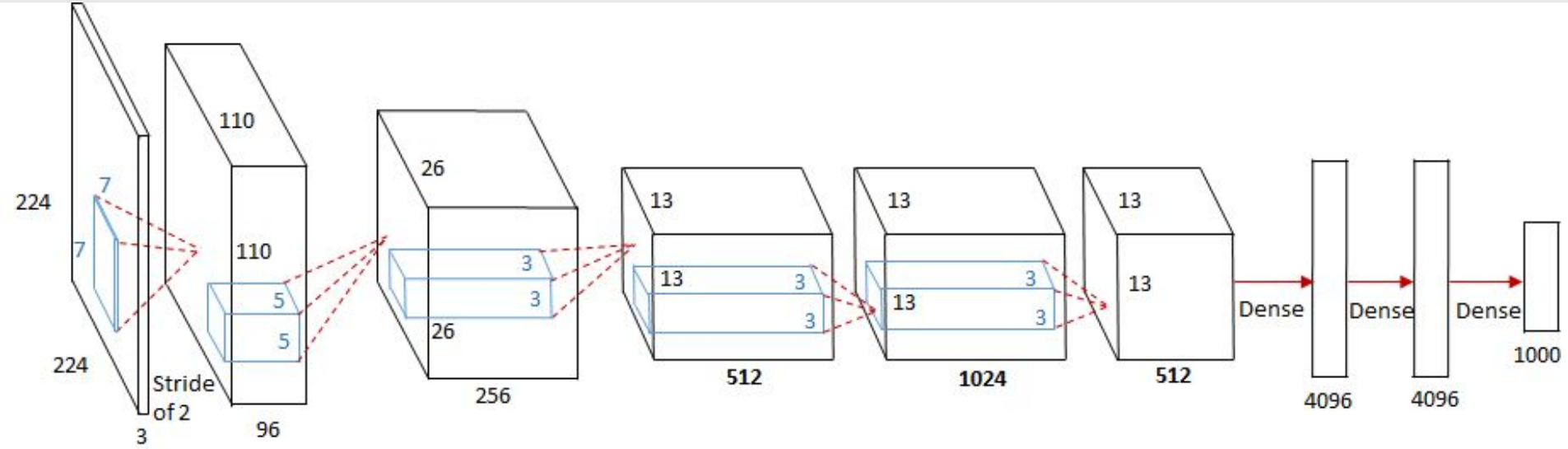






"Visualizing and Understanding Convolutional Networks", ECCV 2014, https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf

ZFNet [Zeiler & Fergus, 2013]

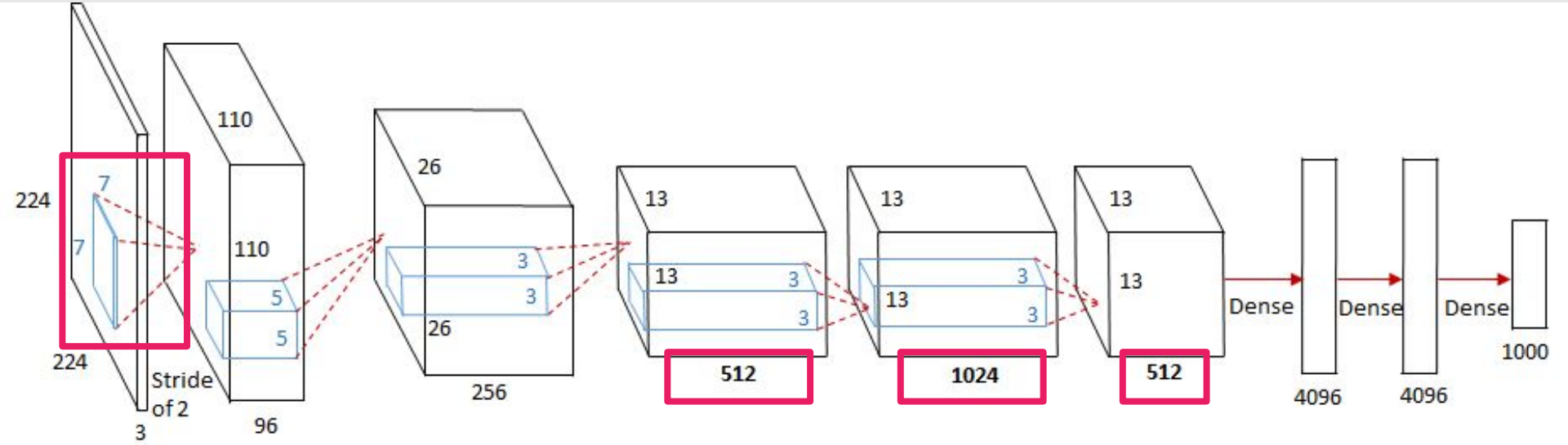


AlexNet but:

CONV1: change from (11x11 stride 4) to (7x7 stride 2)

CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

ZFNet [Zeiler & Fergus, 2013]



AlexNet but:

CONV1: change from (11x11 stride 4) to (7x7 stride 2)

CONV3,4,5: instead of 384, 384, 256 filters use 512, 1024, 512

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - **VGGNet (2014)**
  - GoogLeNet (2014)
  - ResNet (2015)

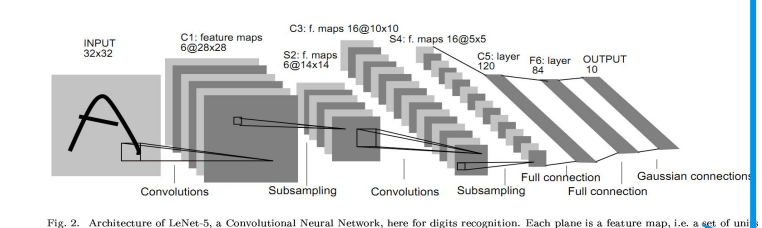
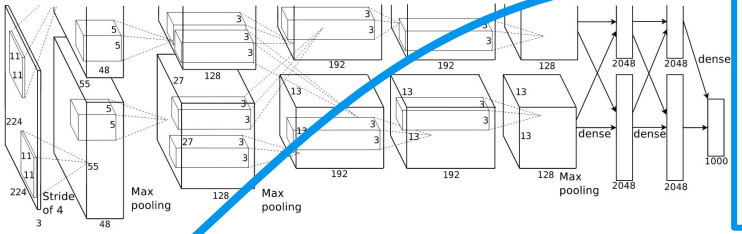
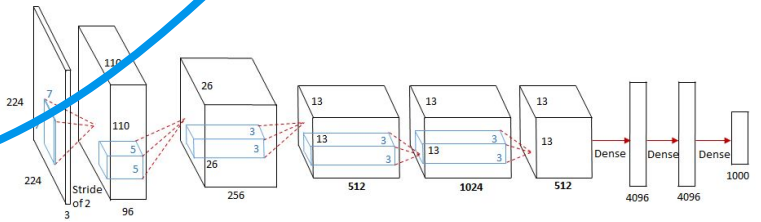
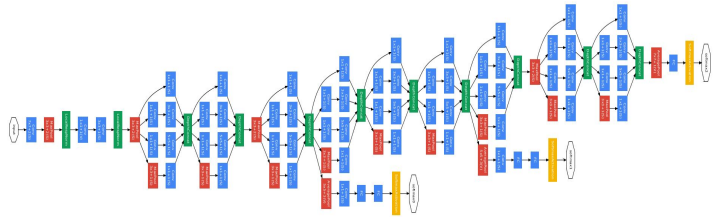
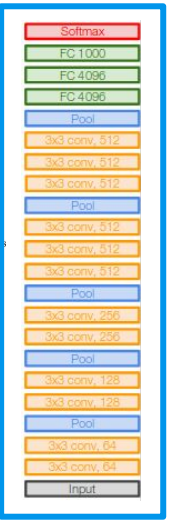


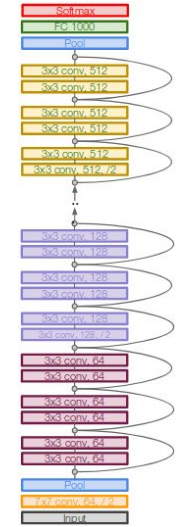
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

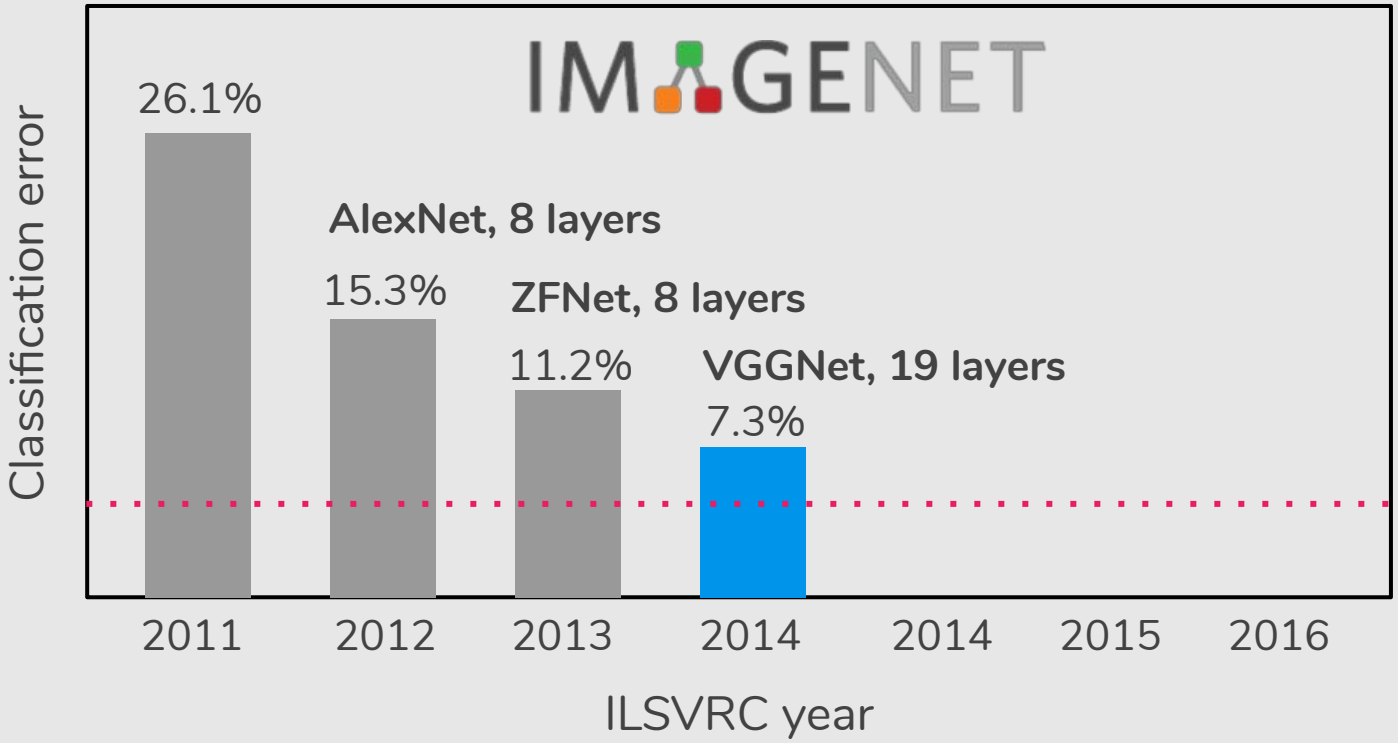












"Very Deep Convolutional Networks for Large-Scale Image Recognition", https://arxiv.org/pdf/1409.1556

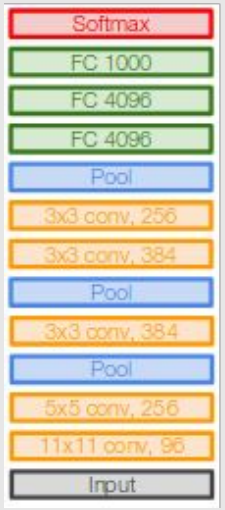
VGGNet [Simonyan & Zisserman, 2014]

Small filters, Deeper networks

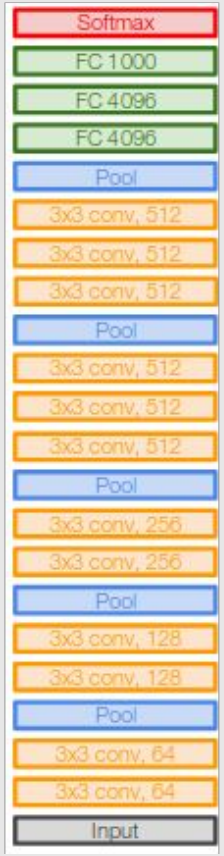
8 layers (AlexNet)
16-19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

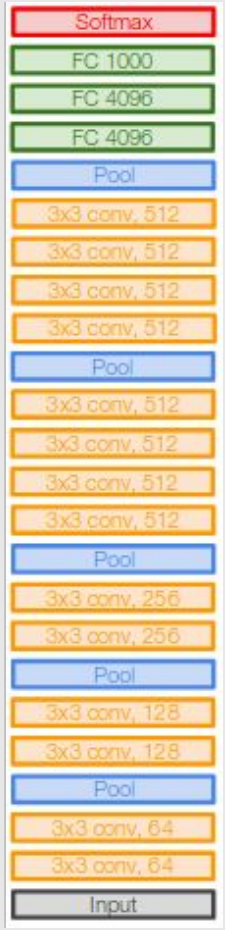
11.2% in ILSVRC'13 (ZFNet)
7.3% in ILSVRC'14



AlexNet



VGG16

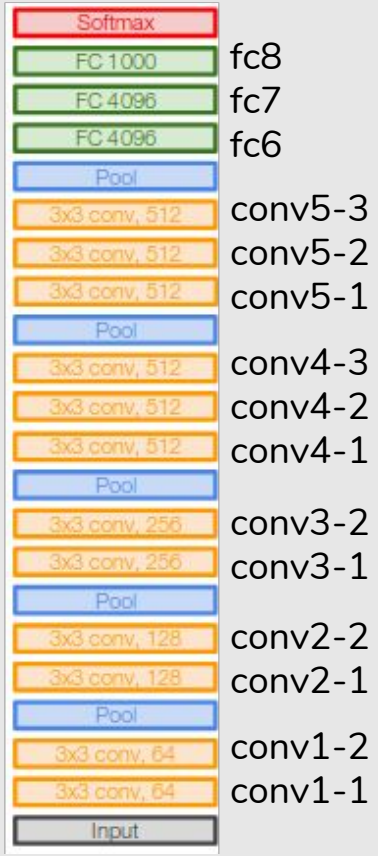


VGG19

VGGNet [Simonyan & Zisserman, 2014]

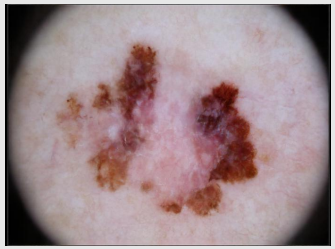
Details:

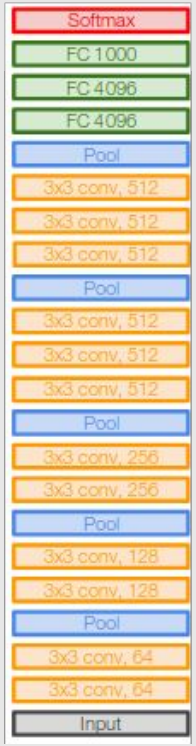
- 138M parameters
- 2nd in classification, 1st in localization
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks



VGG16

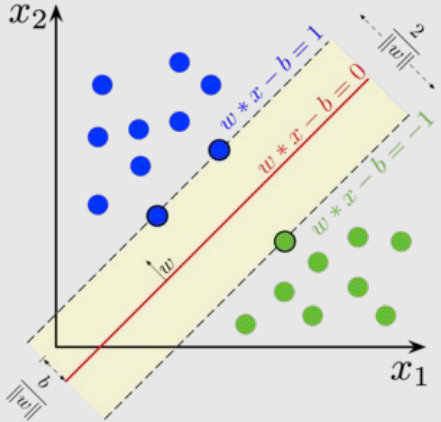
VGGNet [Simonyan & Zisserman, 2014]





[0.01 0.8 1 0.5 … 0.3 0.07 0 0.4 0.6 0 0]
4096-d

Train a classifier (e.g., SVM)



VGG as Feature Extractor

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - **GoogLeNet (2014)**
  - ResNet (2015)

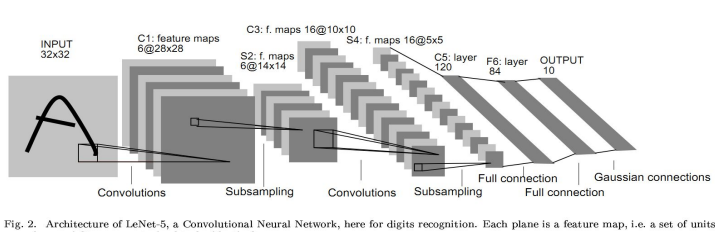
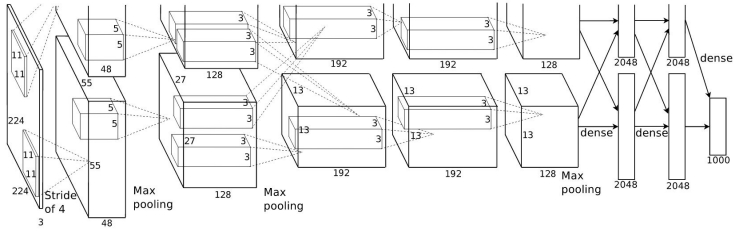
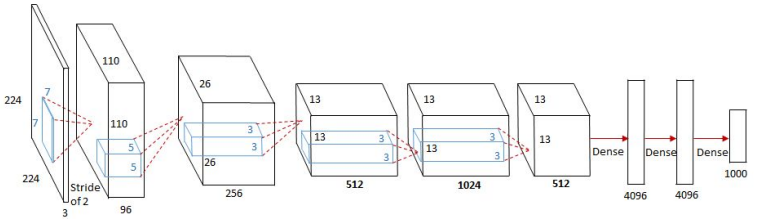
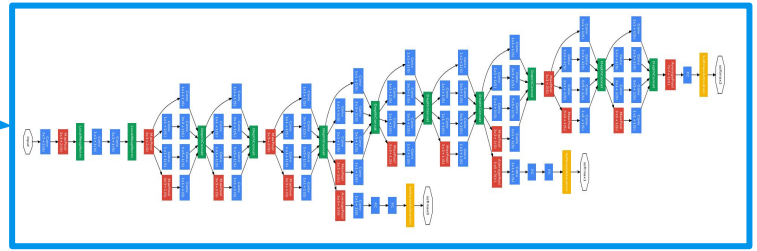
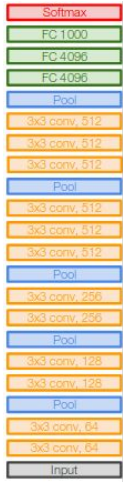


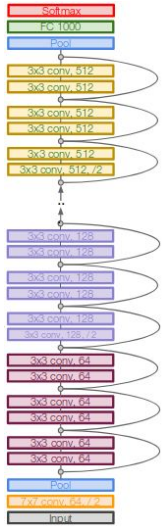
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

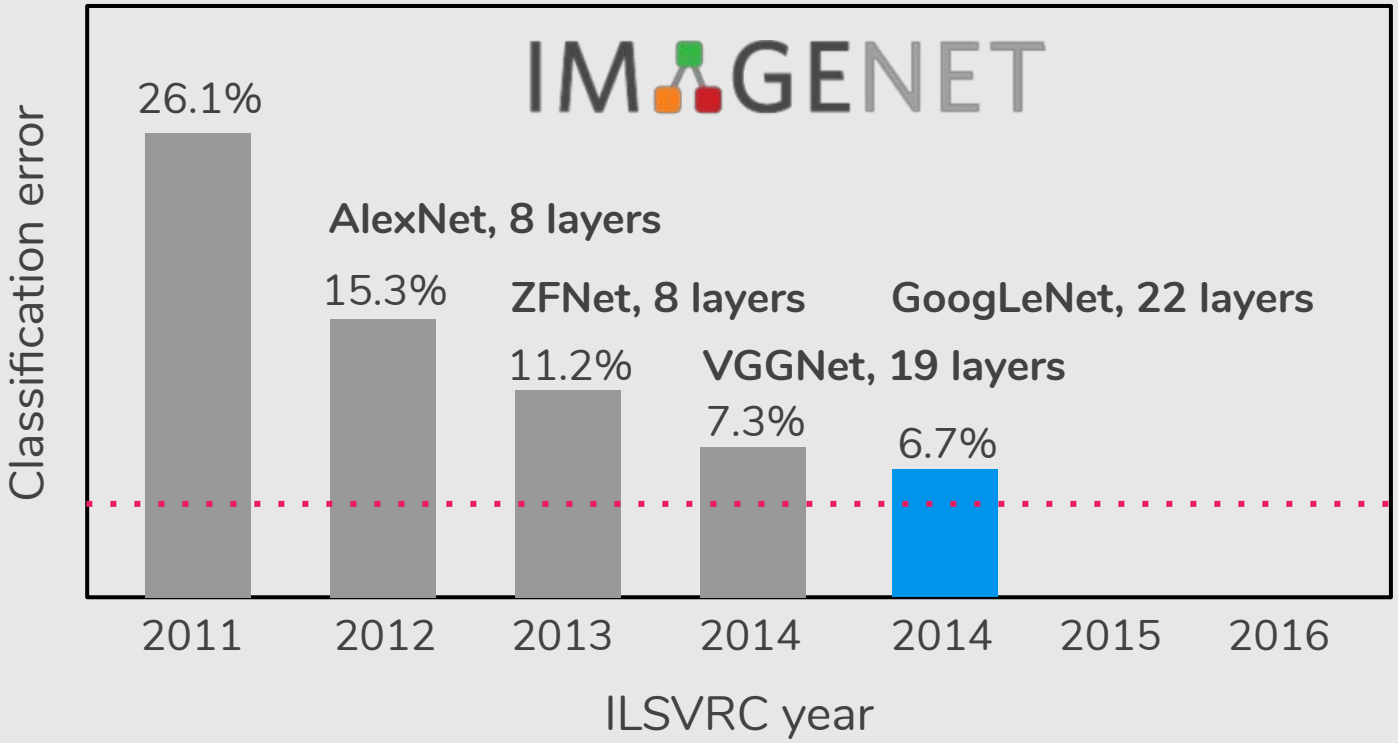












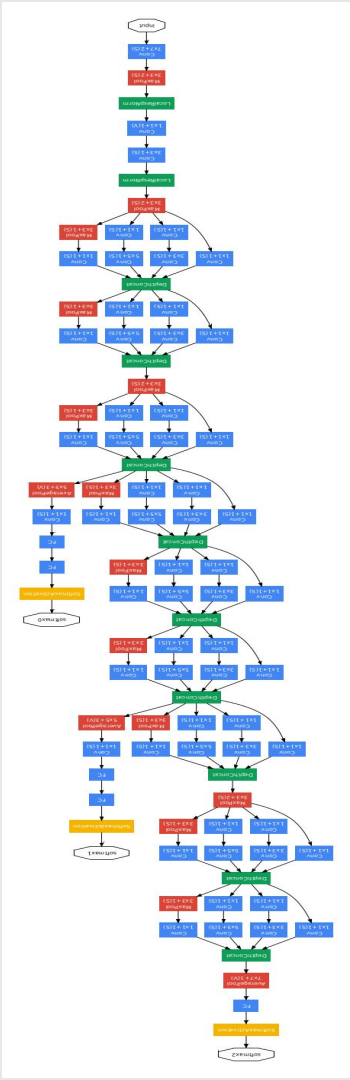
"Very Deep Convolutional Networks for Large-Scale Image Recognition", https://arxiv.org/pdf/1409.1556

GoogLeNet [Szegedy et al., 2014]

Deeper networks, with computational efficiency

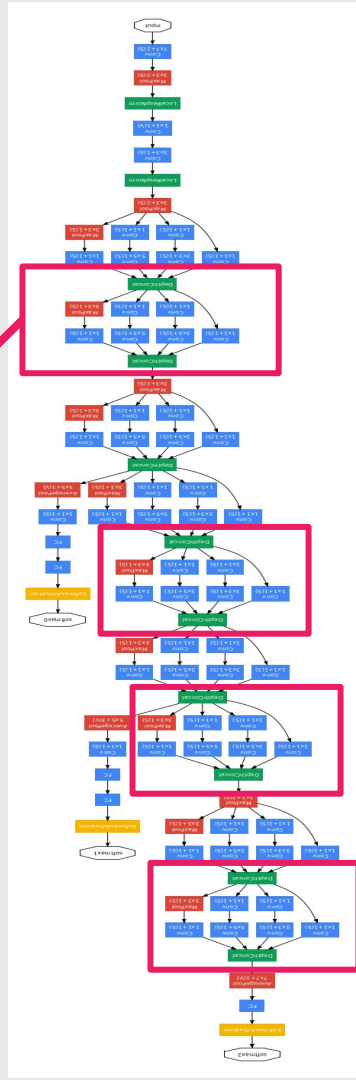
- 22 layers
- Inception module
- Only 5 million parameters!

  12x less than AlexNet

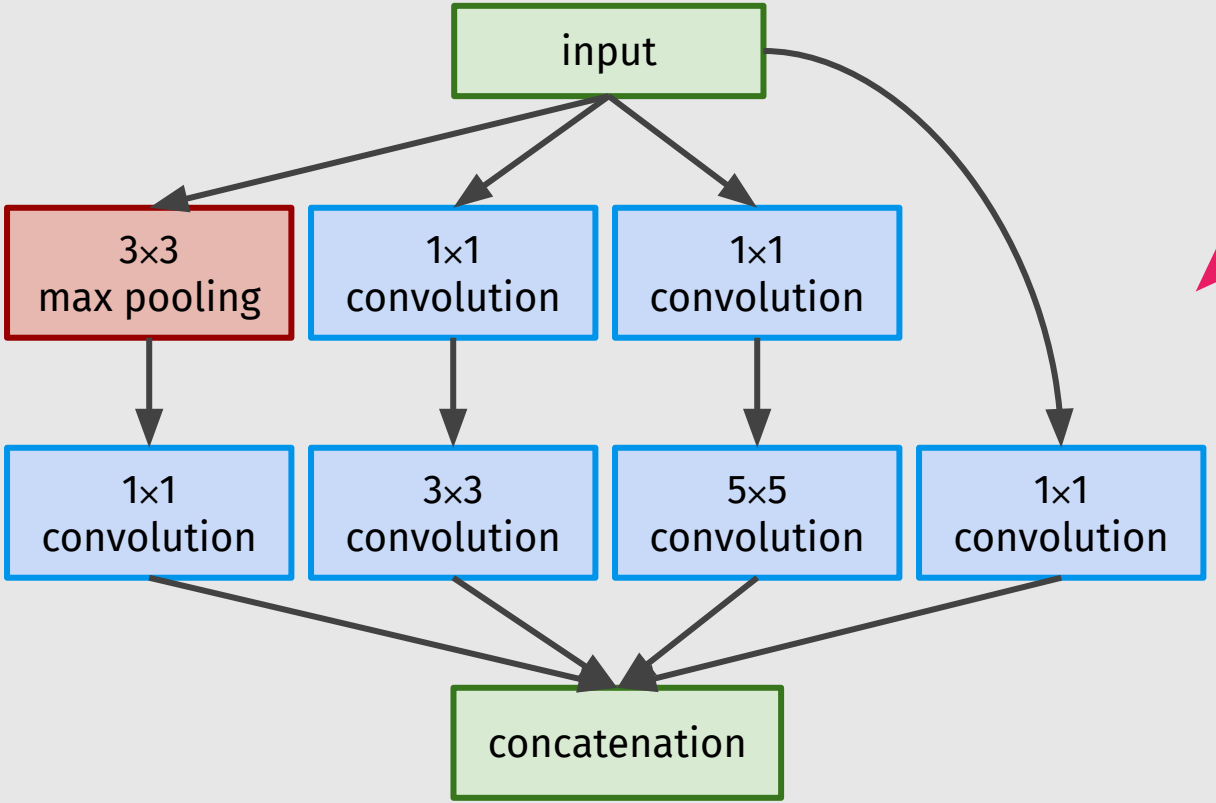


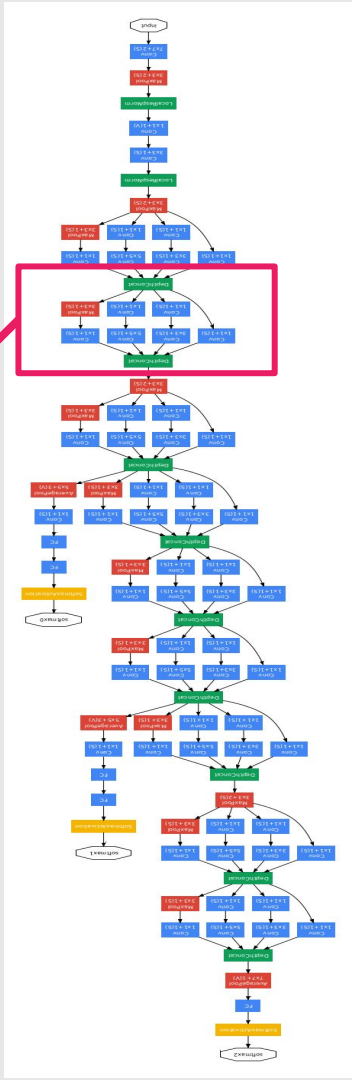
GoogLeNet [Szegedy et al., 2014]

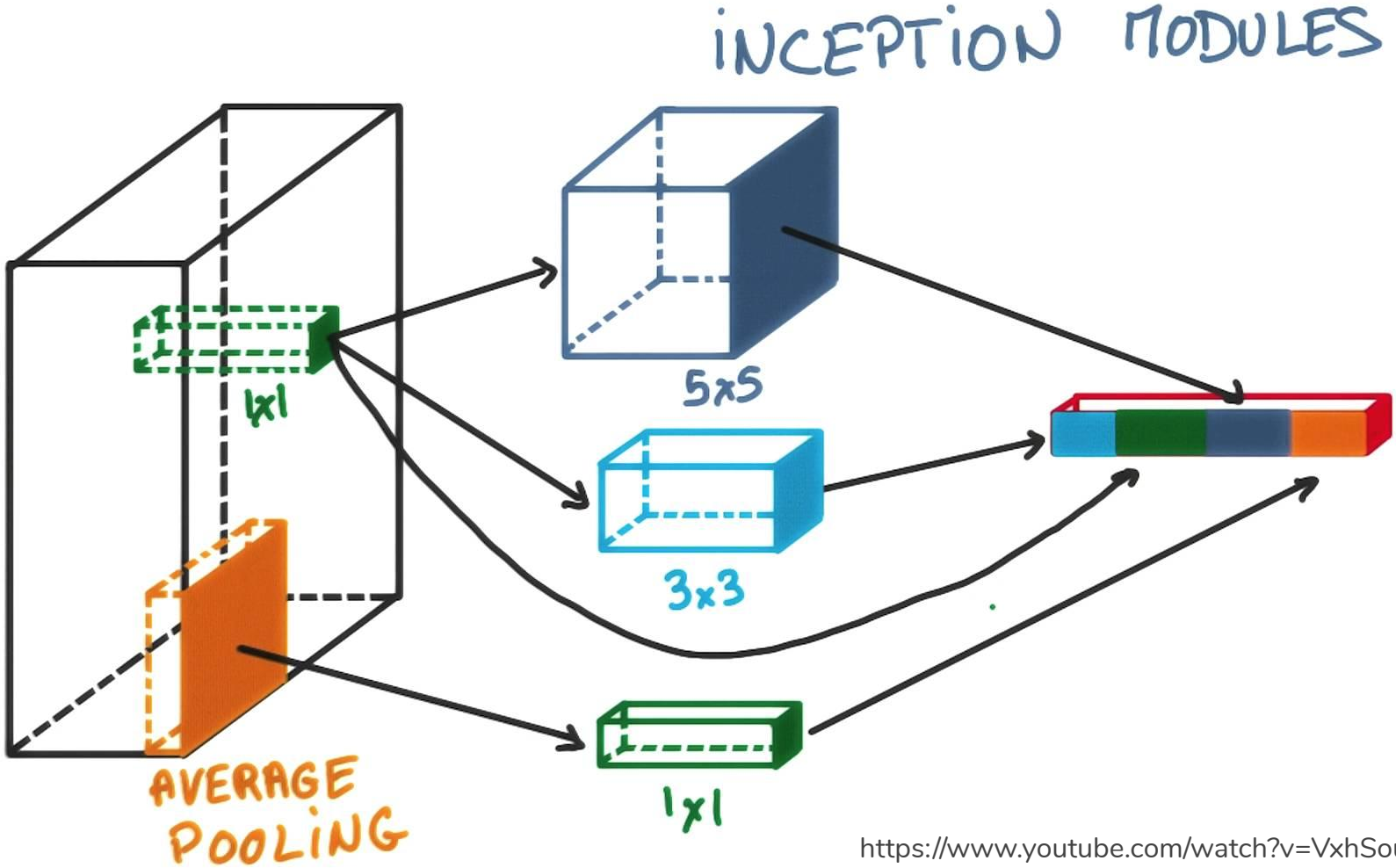
Inception module: design a good local network topology (network within a network) and then stack these modules on top of each other.



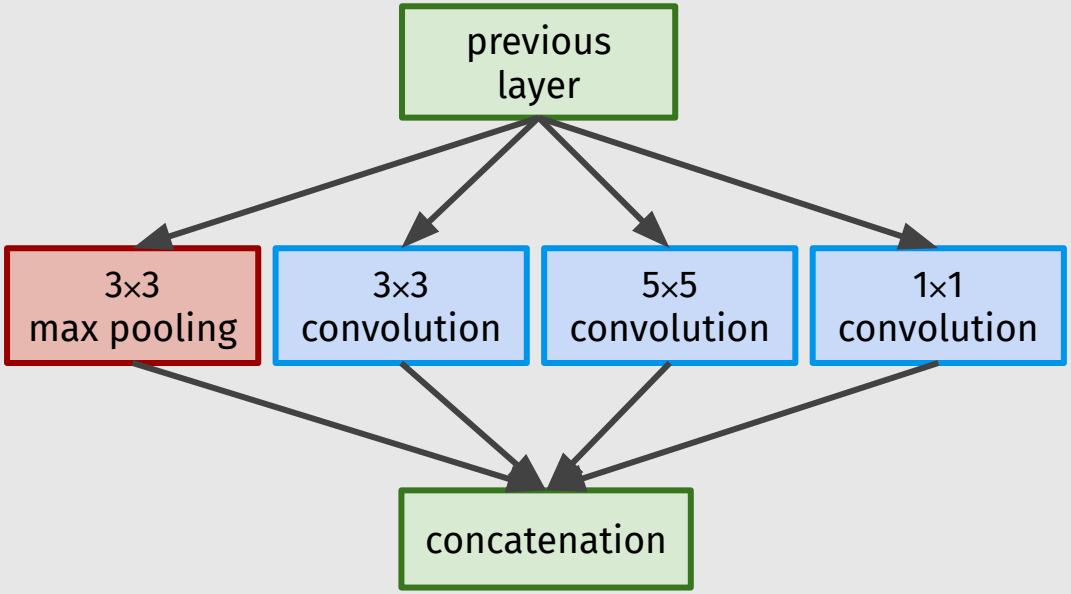
GoogLeNet [Szegedy et al., 2014]






INCEPTION MODULES
1x1
5x5
3x3
1x1
AVERAGE POOLING
https://www.youtube.com/watch?v=VxhSouuSZDY

GoogLeNet [Szegedy et al., 2014]



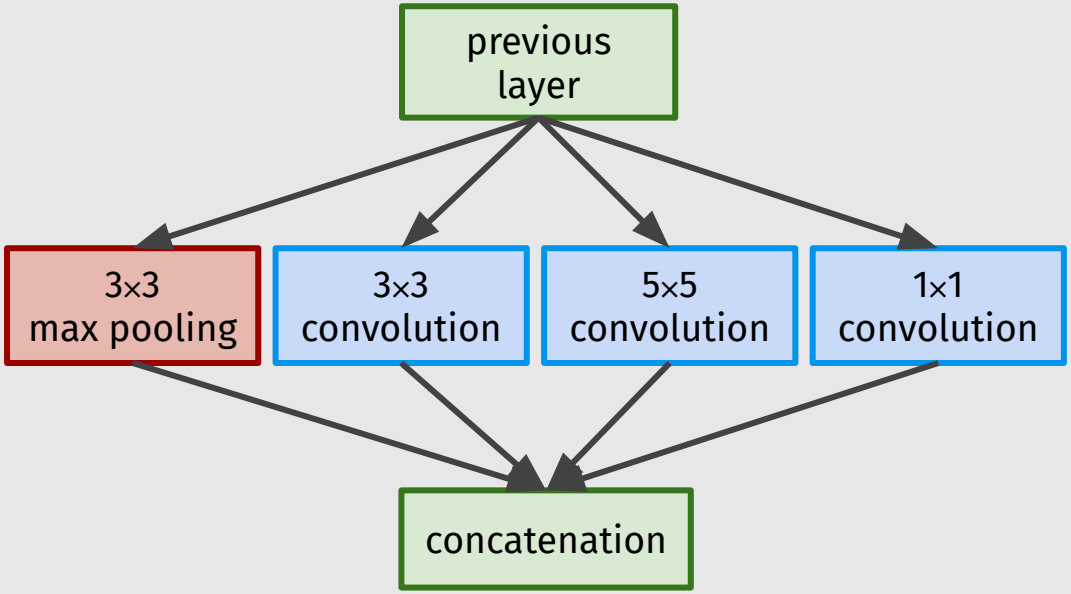
Naive Inception Module

Apply parallel filters on the input from previous layer:

- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

GoogLeNet [Szegedy et al., 2014]



Naive Inception Module

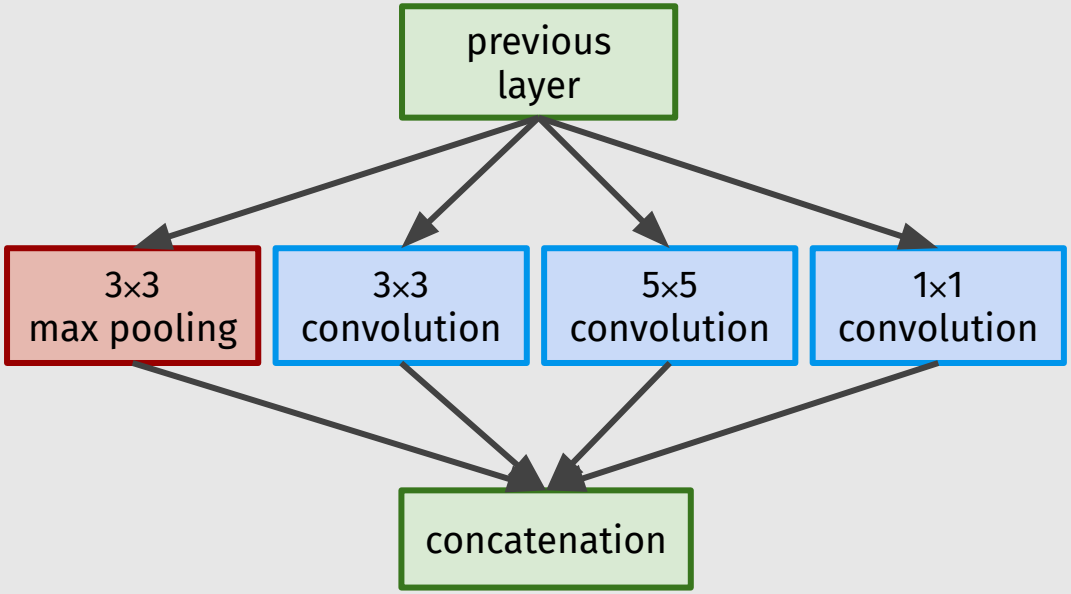
Q: What is the problem with this?

Apply parallel filters on the input from previous layer:

- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

GoogLeNet [Szegedy et al., 2014]



Naive Inception Module

Apply parallel filters on the input from previous layer:

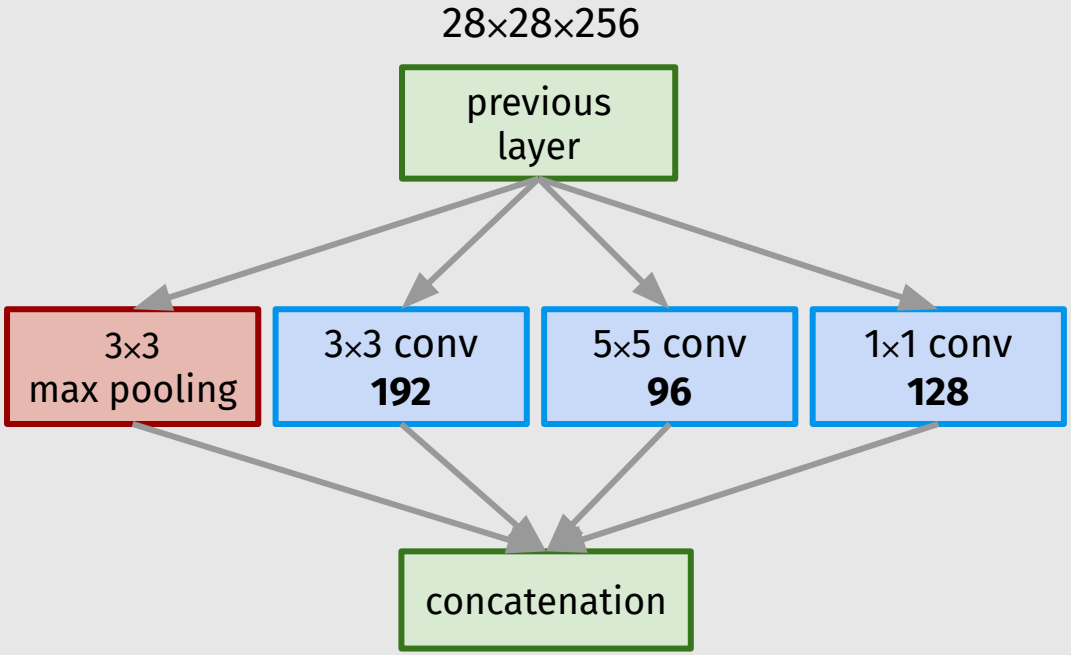
- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

Q: What is the problem with this? Computational complexity!

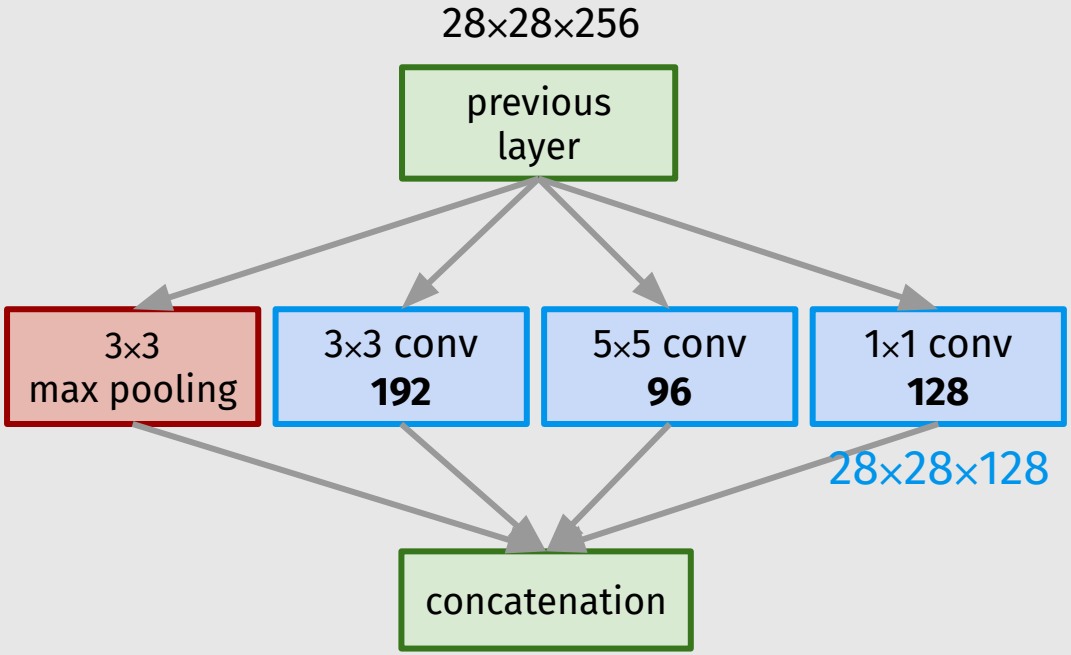
GoogLeNet [Szegedy et al., 2014]

Example: What is the output size of the **1x1 conv, with 128 filters**?



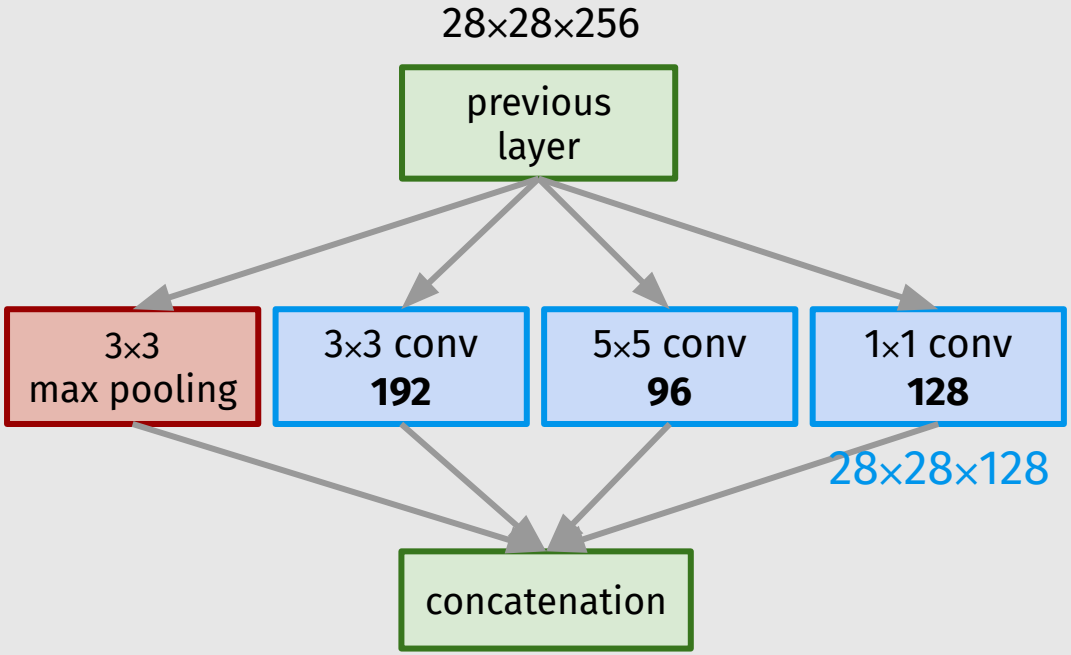
GoogLeNet [Szegedy et al., 2014]

Example: What is the output size of the **1x1 conv, with 128 filters**?



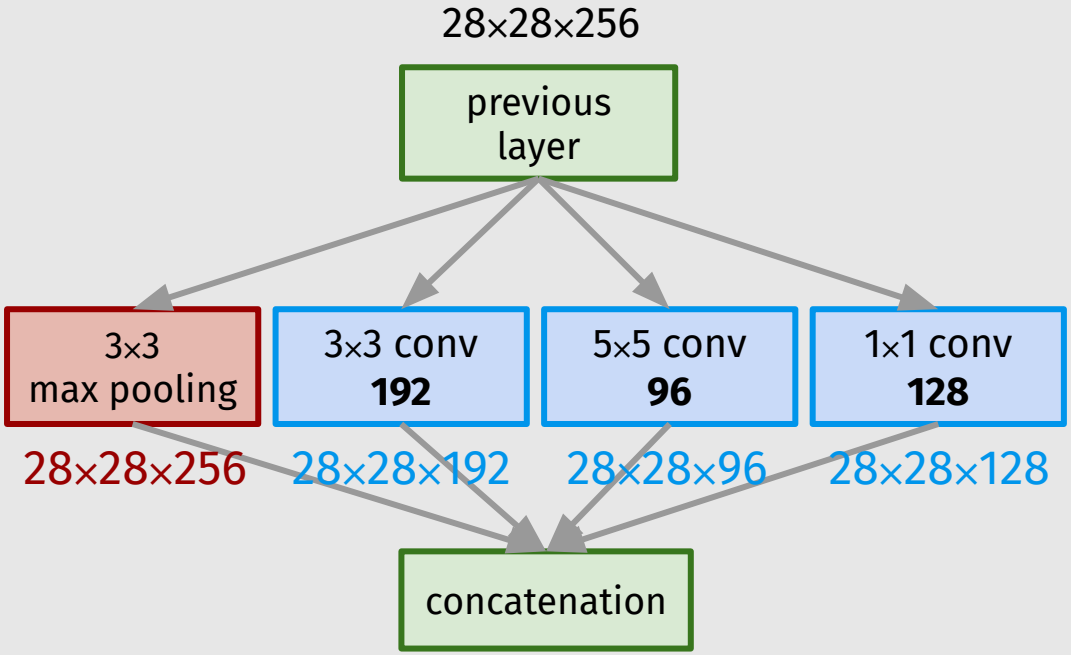
GoogLeNet [Szegedy et al., 2014]

Example: What are the output sizes of all different filter operations?



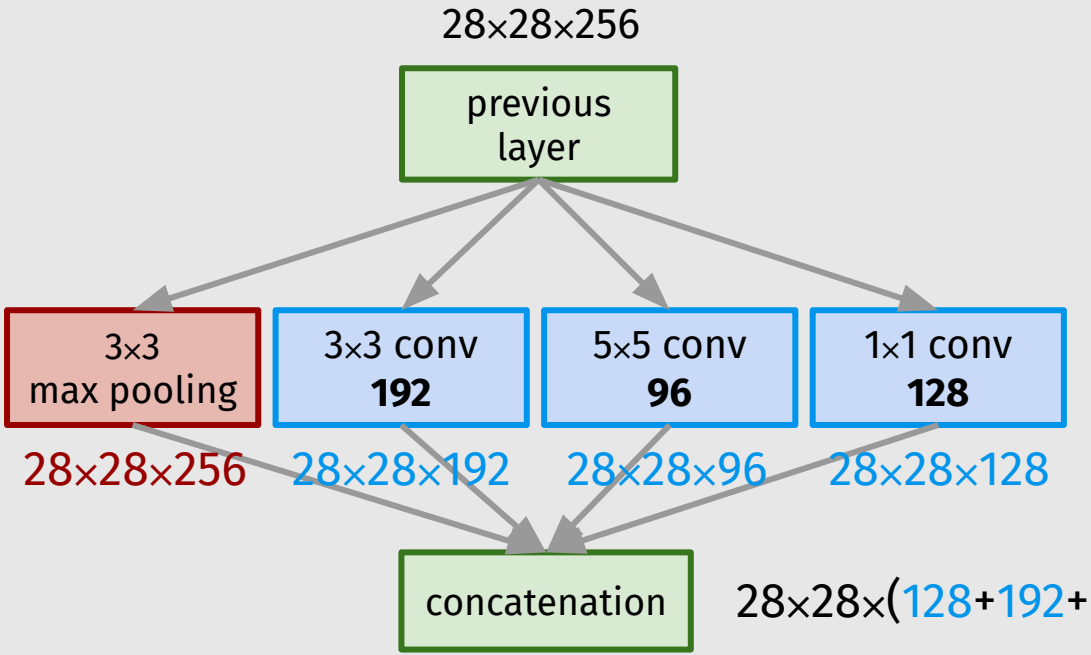
GoogLeNet [Szegedy et al., 2014]

Example: What are the output sizes of all different filter operations?



GoogLeNet [Szegedy et al., 2014]

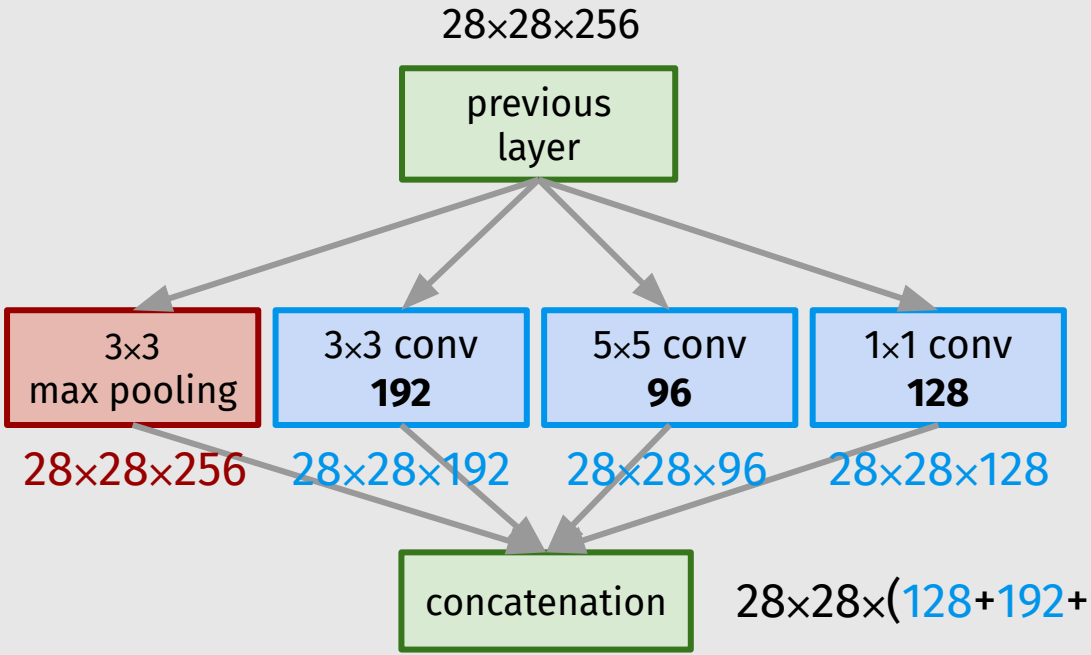
Example: What is output size after filter concatenation?



28×28×(128+192+96+256) = 28×28×672

GoogLeNet [Szegedy et al., 2014]

Example: What is output size after filter concatenation?

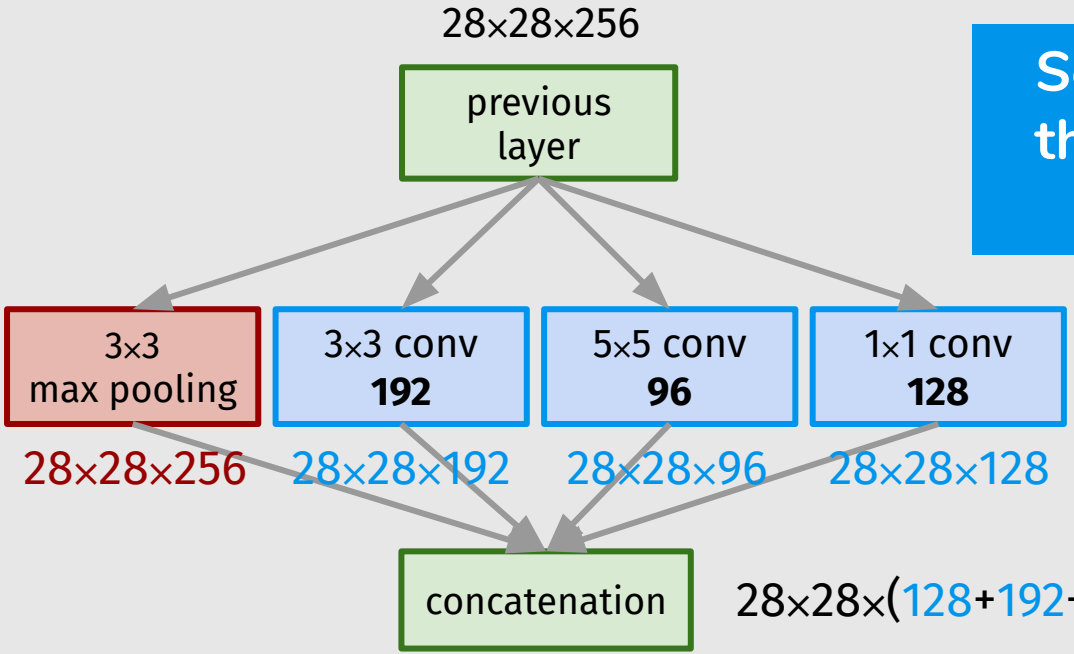


Total: 854M ops!!!

28×28×(128+192+96+256) = 28×28×672

GoogLeNet [Szegedy et al., 2014]

Example: What is output size after filter concatenation?



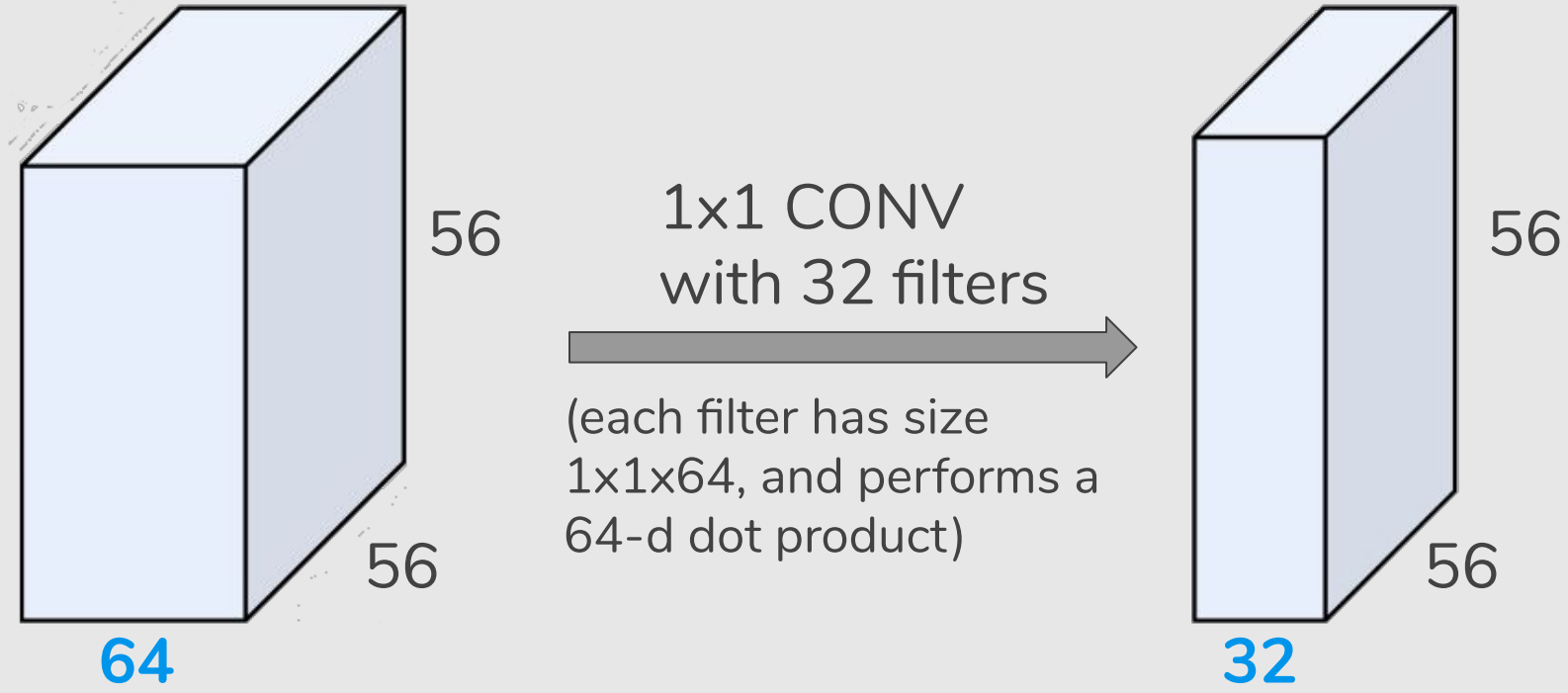
Solution: "bottleneck" layers that use 1x1 convolutions to reduce feature depth

Total: 854M ops!!!

28×28×(128+192+96+256) = 28×28×672

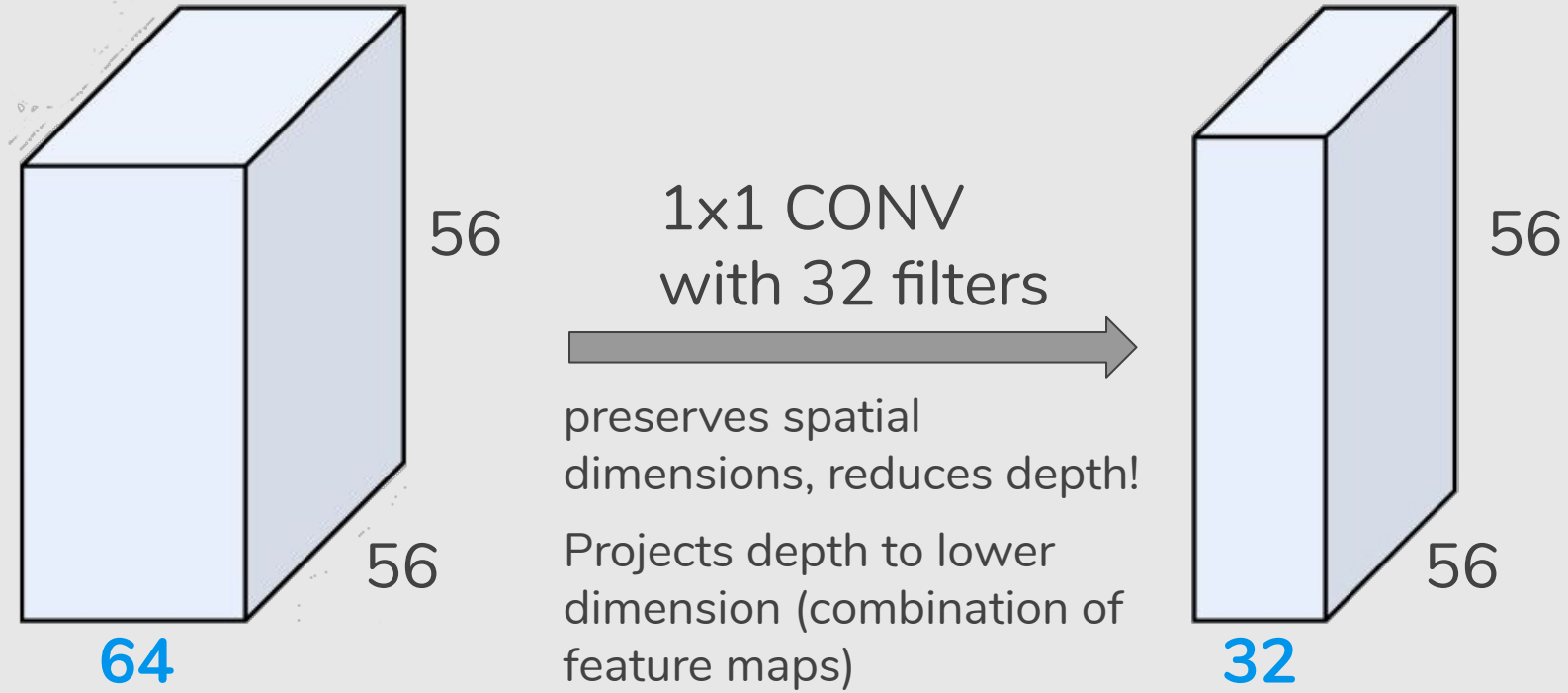
GoogLeNet [Szegedy et al., 2014]

Reminder: 1x1 convolutions

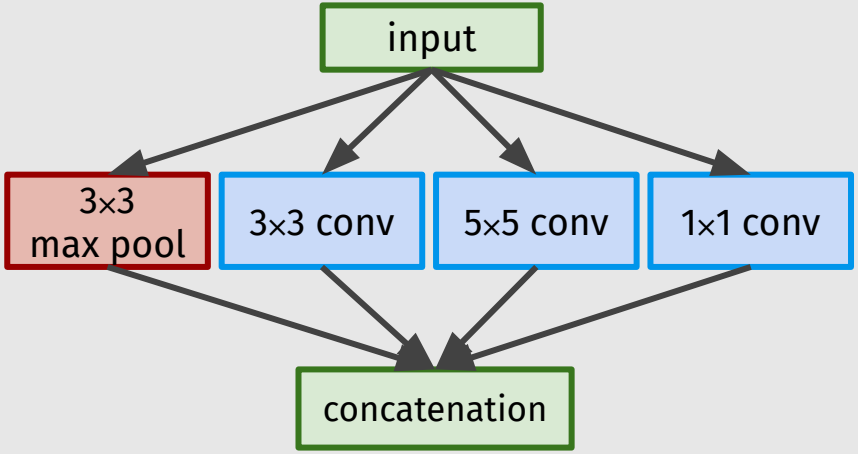


GoogLeNet [Szegedy et al., 2014]

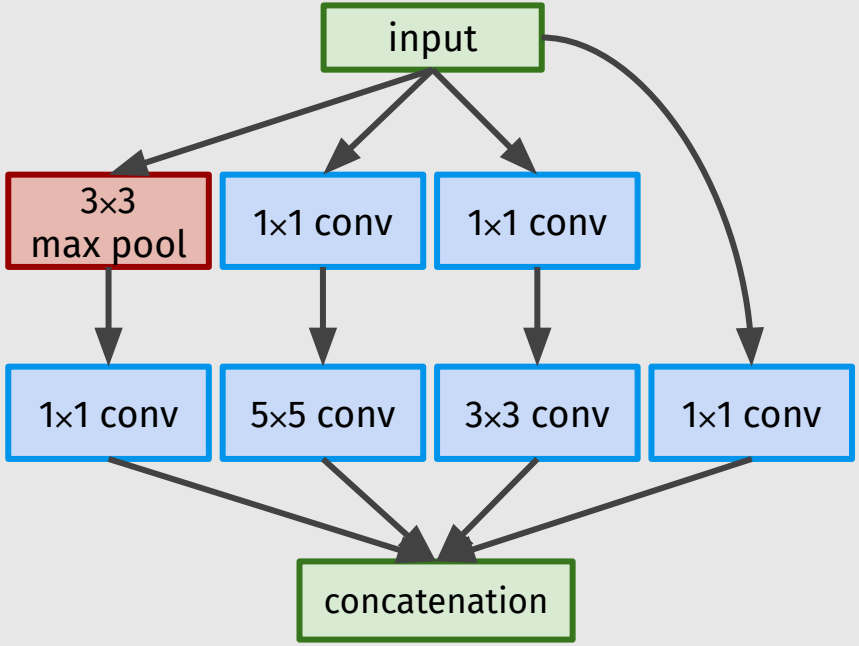
Reminder: 1x1 convolutions



GoogLeNet [Szegedy et al., 2014]



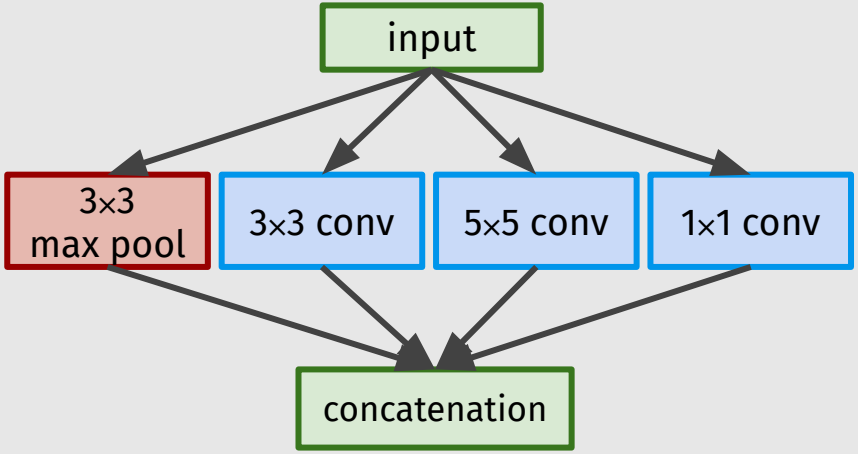
Naive Inception Module



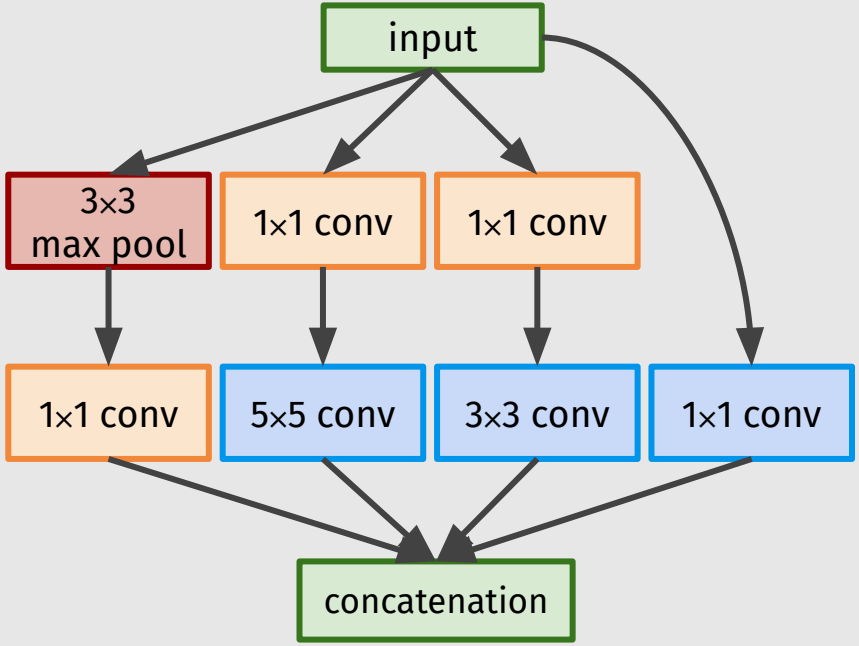
Inception Module

GoogLeNet [Szegedy et al., 2014]

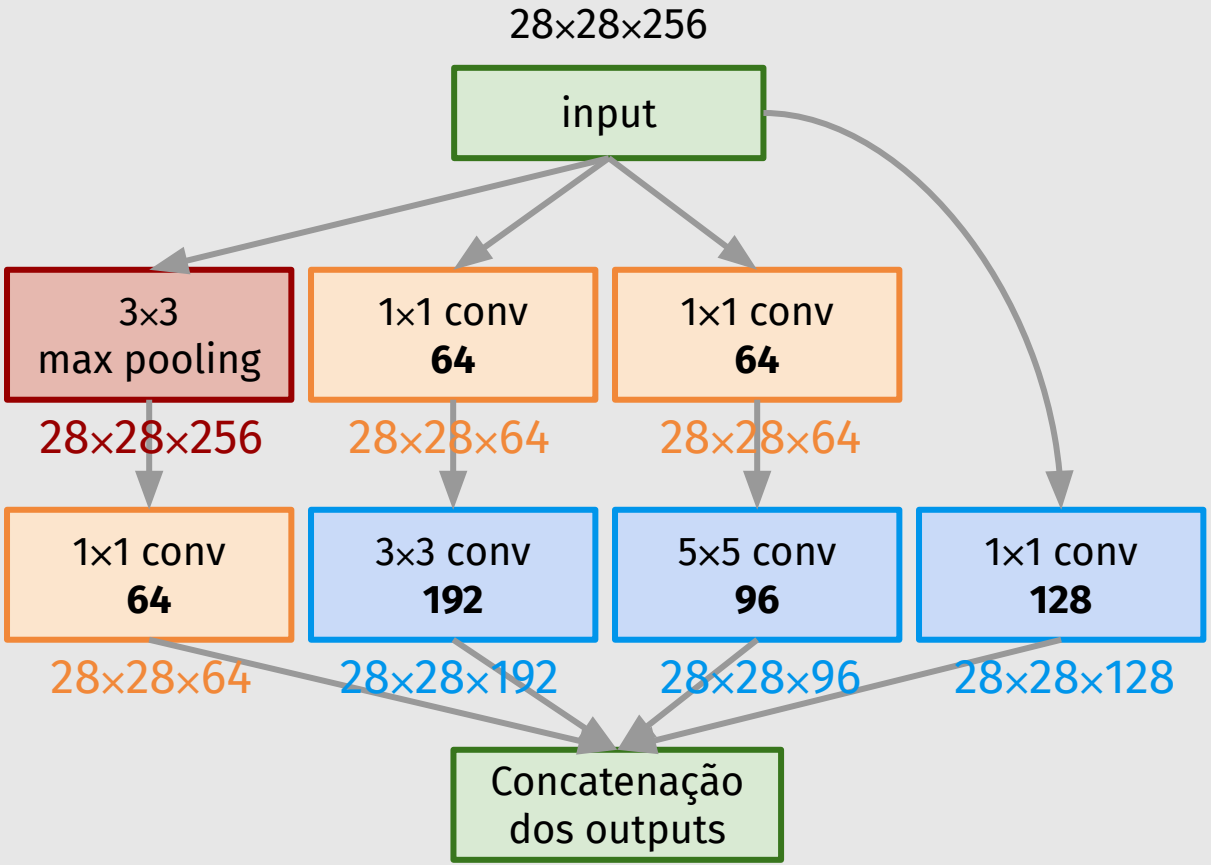
1x1 conv
"bottleneck" layers



Naive Inception Module



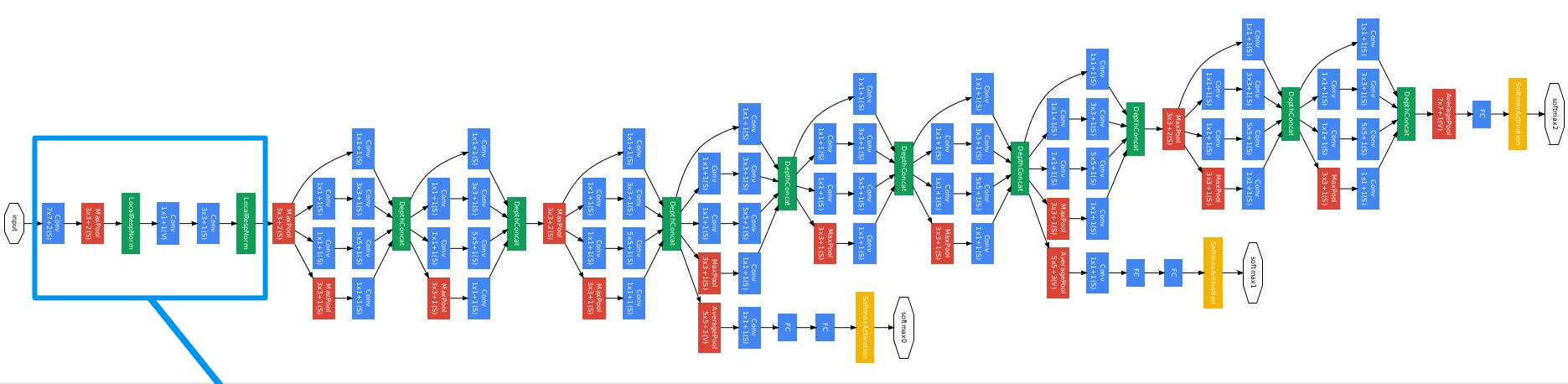
Inception Module



28×28×(128+192+96+64) = 28×28×480

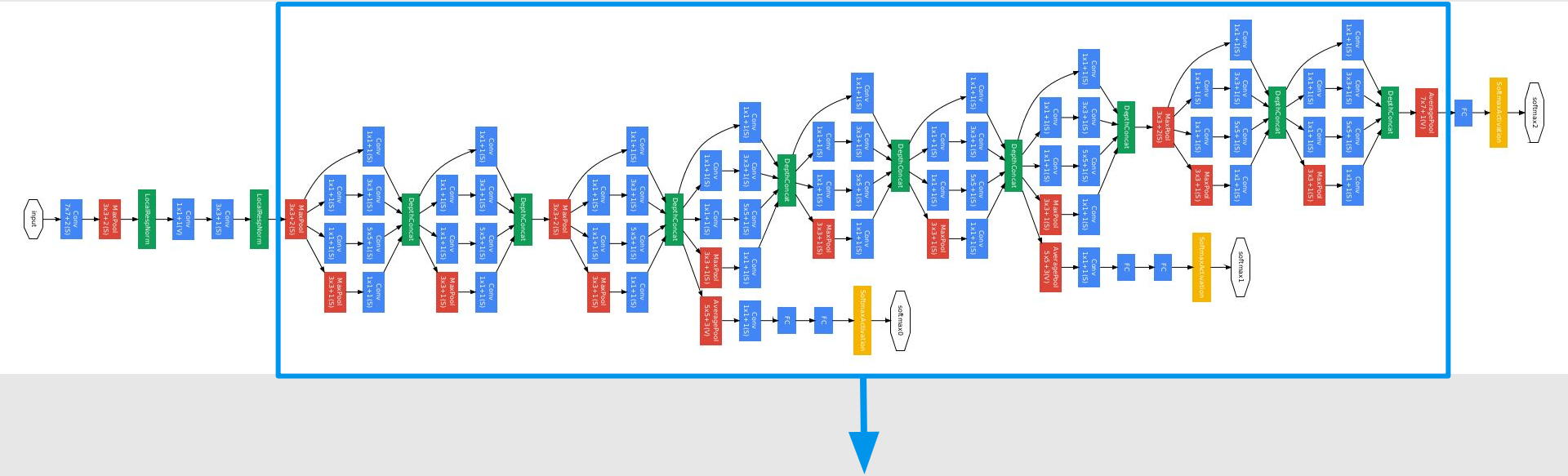
358M ops!!!
(vs. 854M)

GoogLeNet [Szegedy et al., 2014]



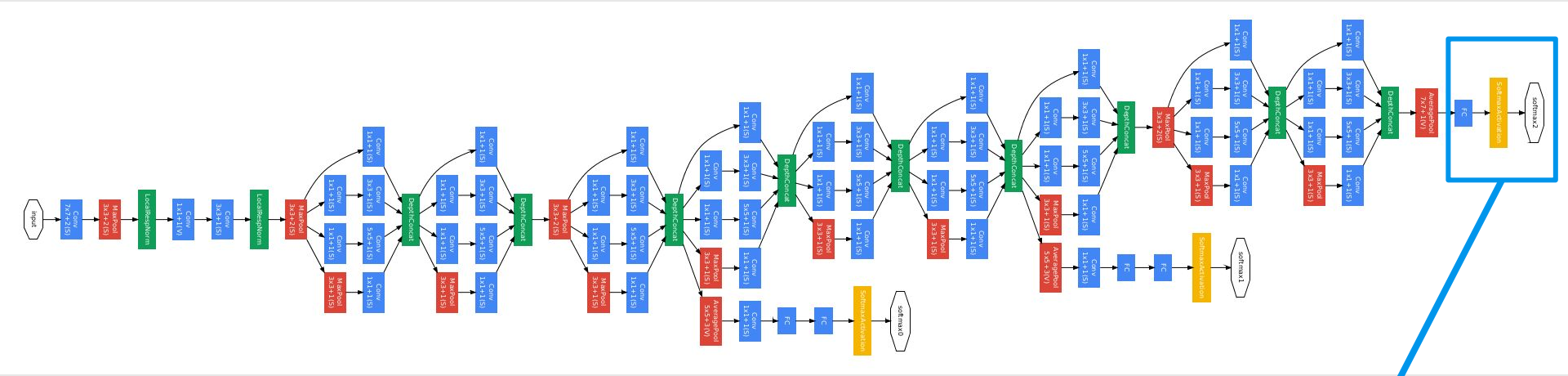
Conv-Pool 2x Conv-Pool

GoogLeNet [Szegedy et al., 2014]



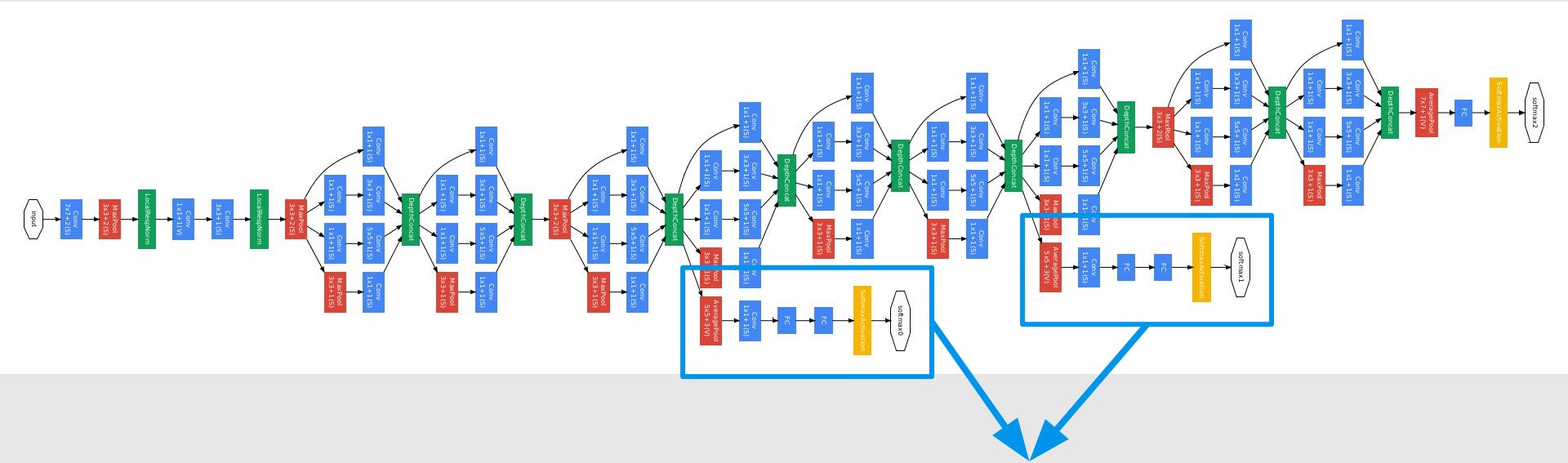
Stacked Inception Modules

GoogLeNet [Szegedy et al., 2014]

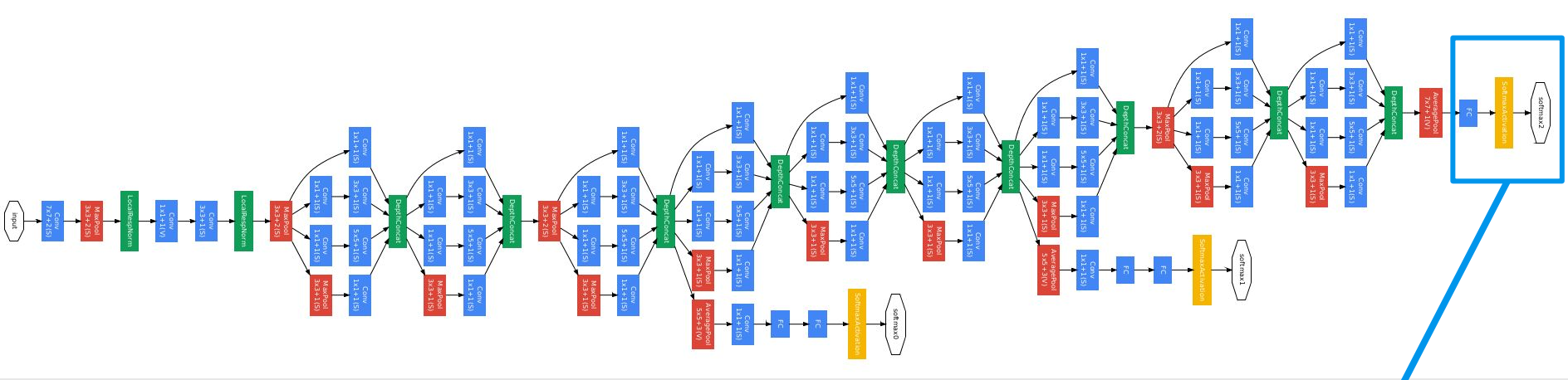


Classifier Output

GoogLeNet [Szegedy et al., 2014]



GoogLeNet [Szegedy et al., 2014]



The total loss function is a weighted sum of the auxiliary loss and the real loss.

```
total_loss = real_loss + 0.3*aux_loss_1 + 0.3*aux_loss_2
```

GoogLeNet [Szegedy et al., 2014]

- GoogLeNet has **9 inception modules** stacked linearly.
- It is **22 layers deep** (27, including the pooling layers).
- It uses **global average pooling** at the end of the last inception module.

GoogLeNet [Szegedy et al., 2014]

- GoogLeNet has **9 inception modules** stacked linearly.
- It is **22 layers deep** (27, including the pooling layers).
- It uses **global average pooling** at the end of the last inception module.
- GoogLeNet = Inception v1
- Inception v2, v3, v4, Inception-ResNet v1, v2: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - **ResNet (2015)**

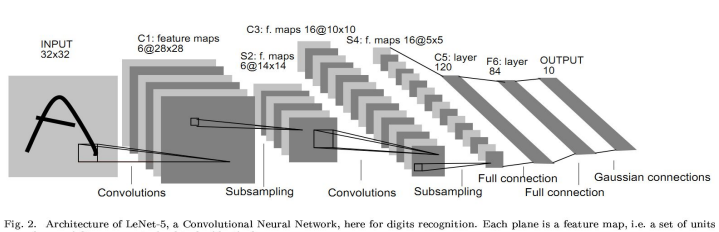
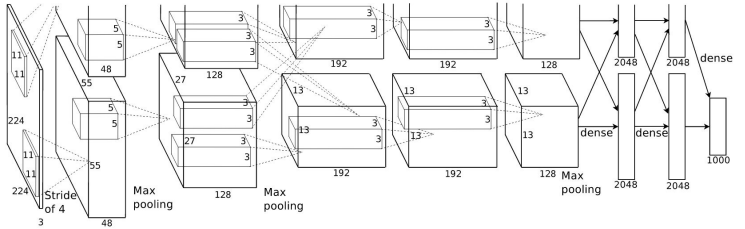
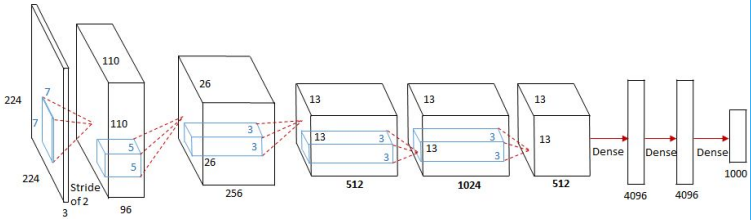
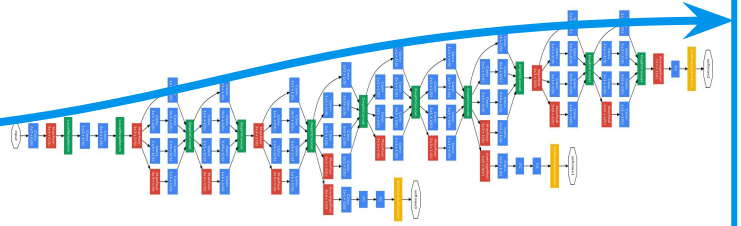
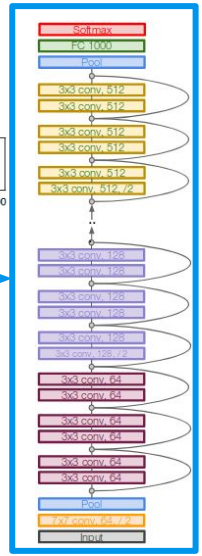


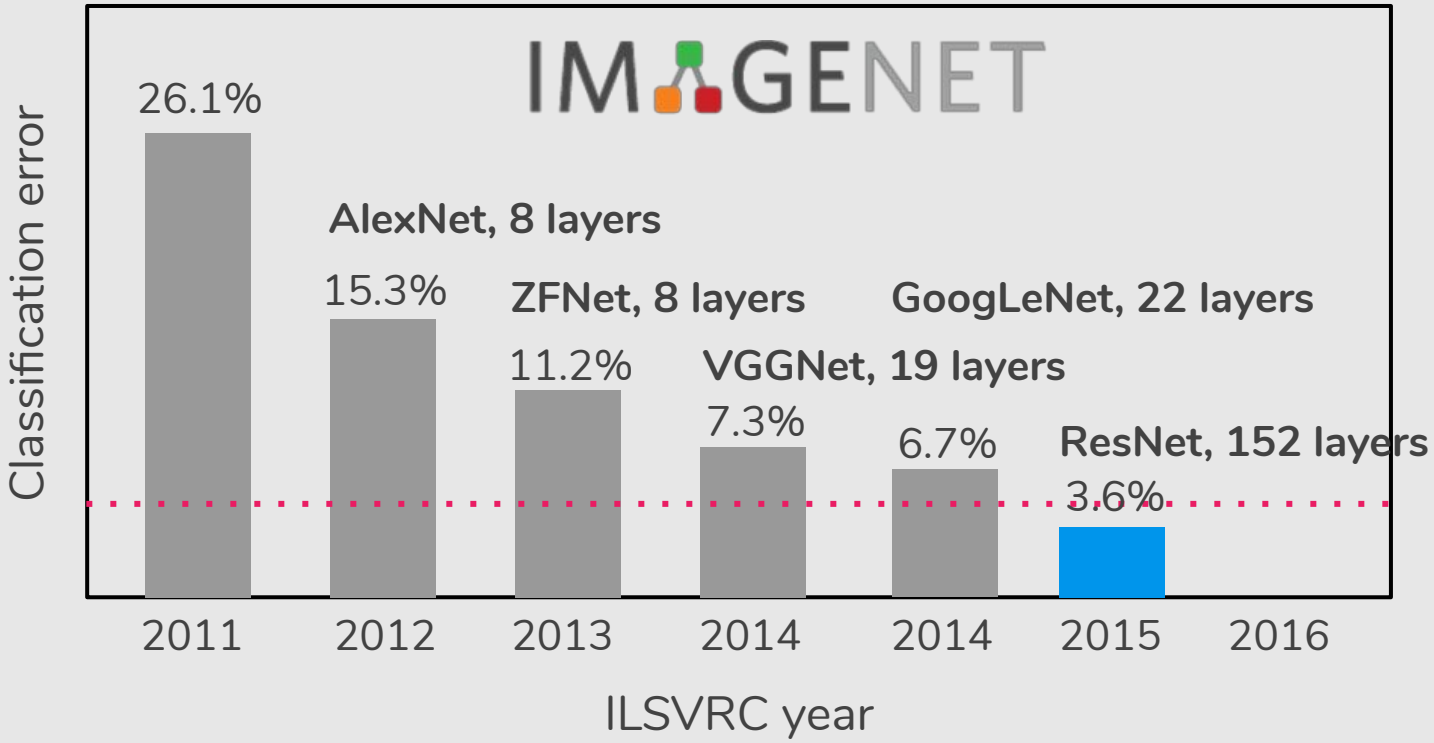
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.











"Deep Residual Learning for Image Recognition", CVPR 2016, https://arxiv.org/pdf/1512.03385

To be continued ...