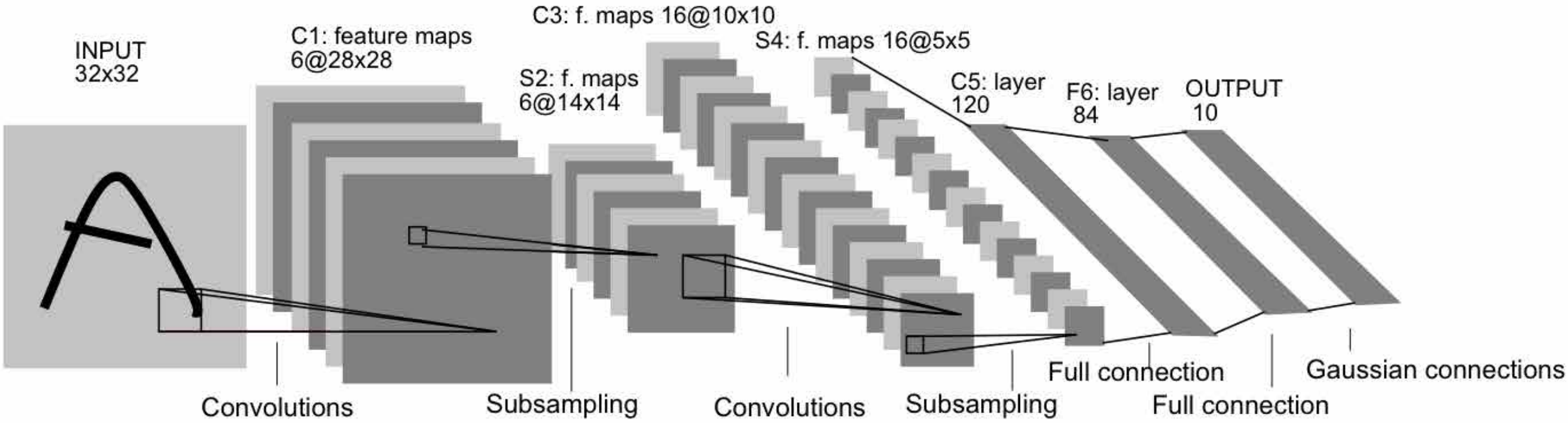
CS 466/566
Introduction to Deep Learning

Lecture 9 – Deep Learning Architectures and Details of Conv-Filters

Compiled from various resources

The thing that started all: LeNet-5 (1998)

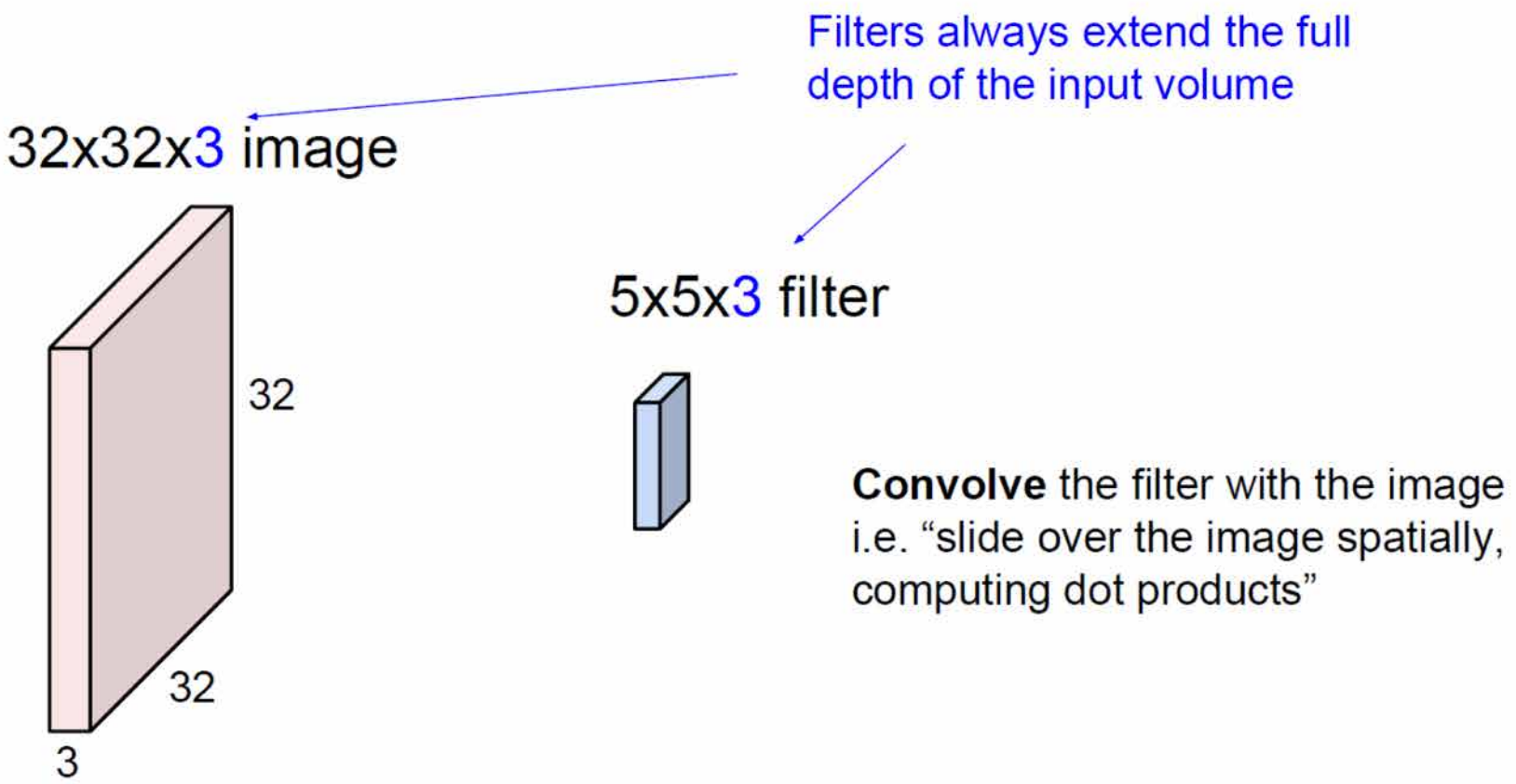


- Architecture of LeNet-5 (LeCun, 1998), a Convolutional Neural Network for digits recognition.
- Each plane is a feature map, a set of weights of a unique neuron applied over a local receptive field of previous layer's output.

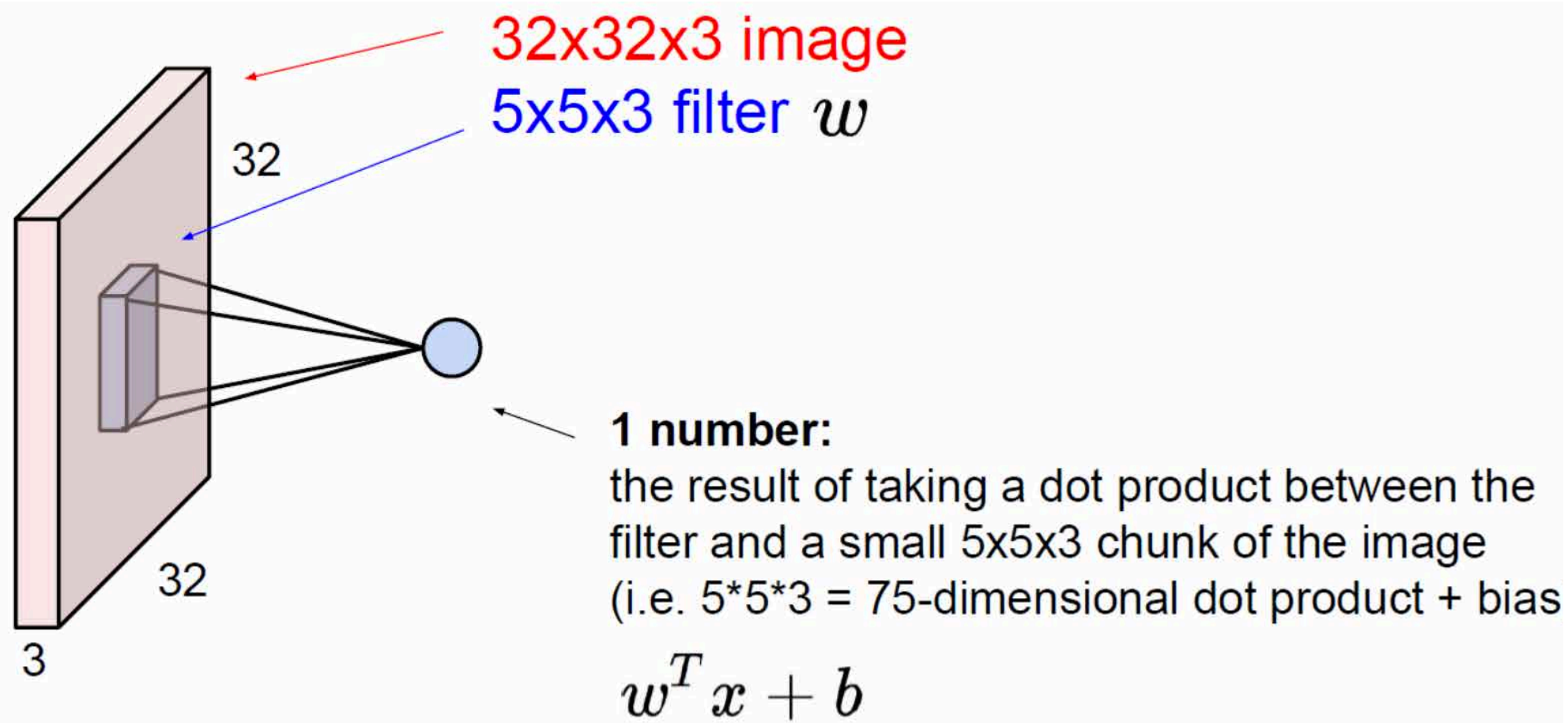
Convolutional Layers Revisited

- This layer consists of a set of learnable filters that we slide over the image spatially, computing dot products between the entries of the filter and the input image.
- **The filters should extend to the full depth of the input image.**
- For example, if we want to apply a filter of size 5x5 to a colored image of size 32x32, then the filter should have depth 3 (5x5x3) to cover all 3 color channels (Red, Green, Blue) of the image.

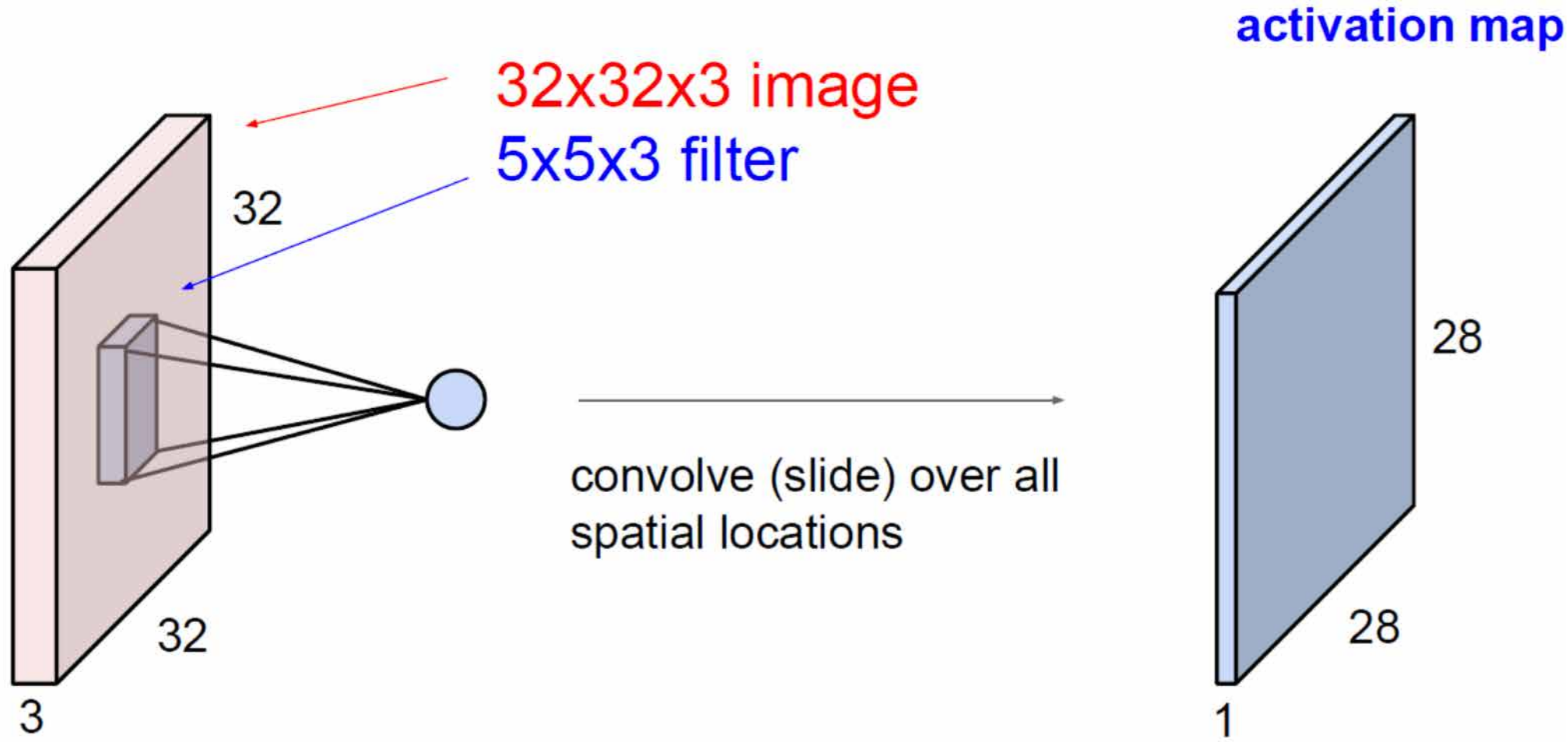
Convolutional Layers Revisited:
Multiple Input Channels (Feature maps)



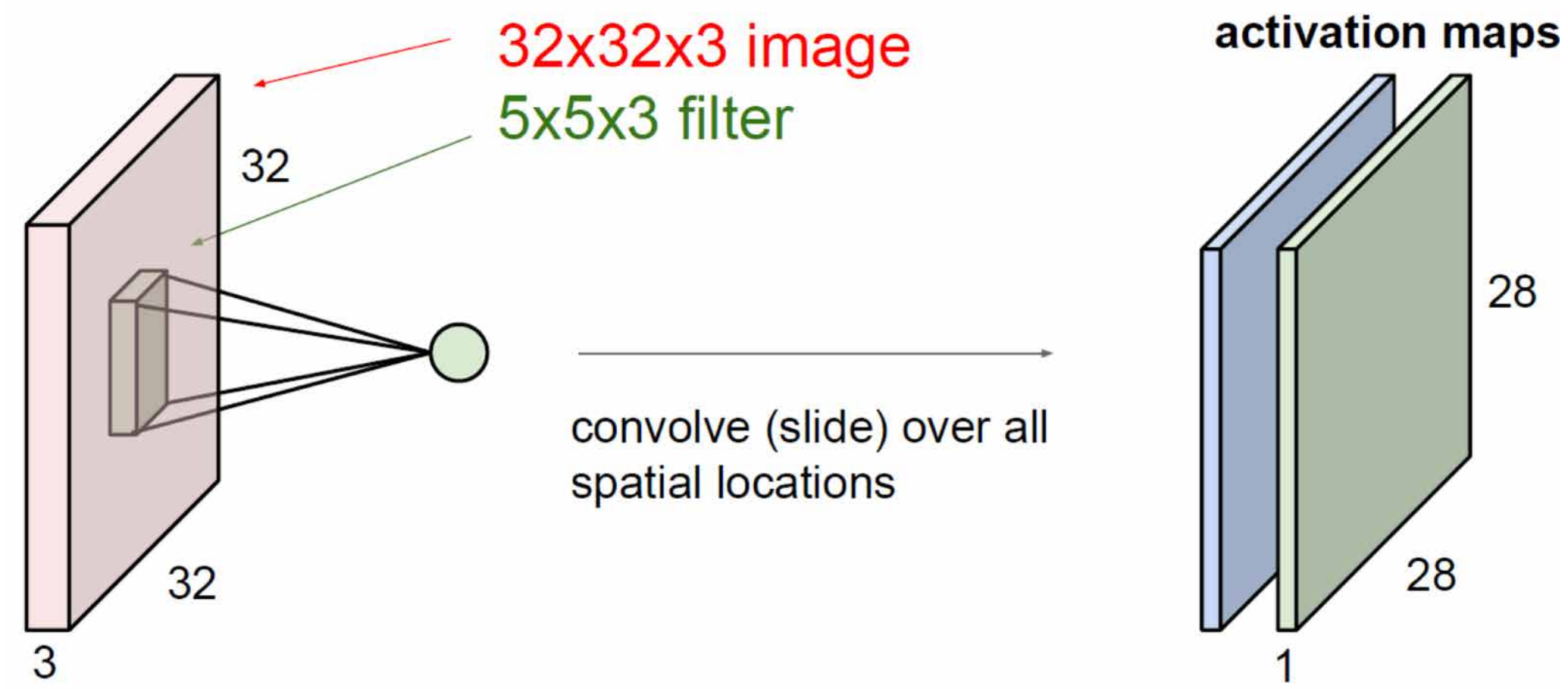
Convolutional Layers Revisited:
Multiple Input Channels (Feature maps)



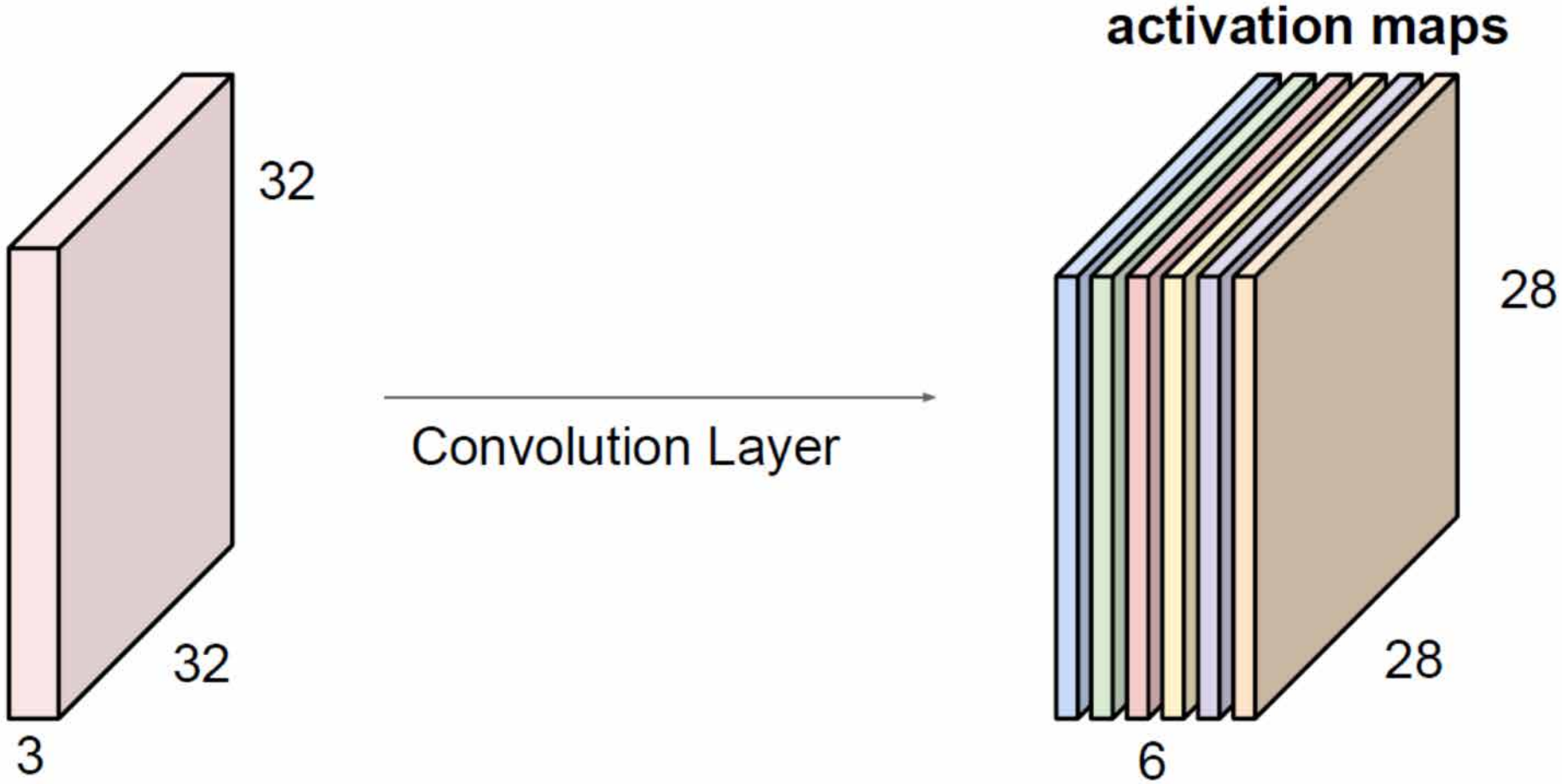
Convolutional Layers Revisited:
Multiple Input Channels (Feature maps)



Convolutional Layers Revisited:
Multiple Input Channels (Feature maps)



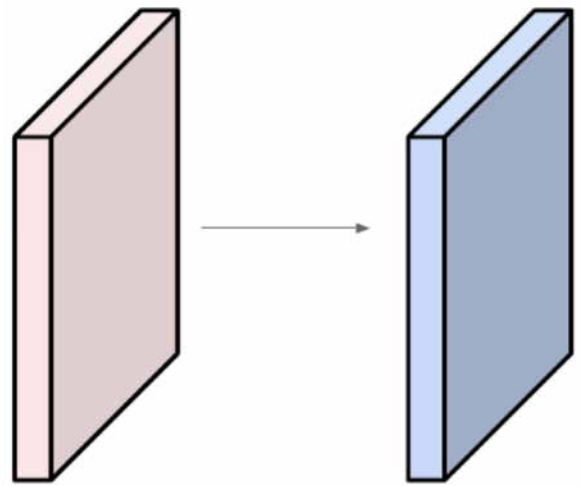
Convolutional Layers Revisited:
Multiple Input Channels (Feature maps)



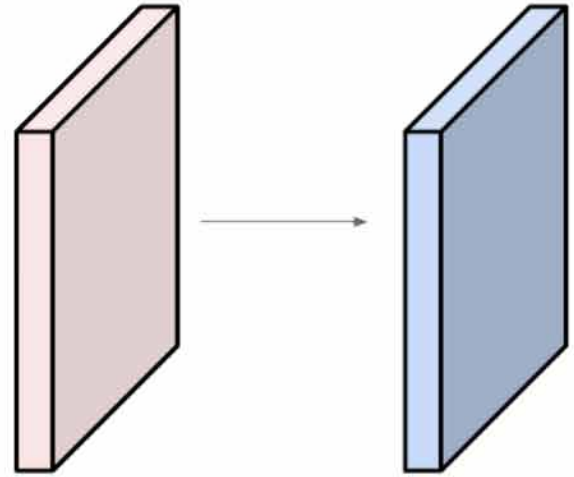
We stack these up to get a “new image” of size 28x28x6!

Convolutional Layers Revisited:
Multiple Input Channels (Feature maps)

- Example:
  - Input volume: 32x32x3
  - 10 filters of 5 by 5 with stride 1, pad 2
  - How many parameters in this layer?



Convolutional Layers Revisited:
Multiple Input Channels (Feature maps)



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Convolutional Layers Revisited: Multiple Input Channels (Feature maps)

http://cs231n.github.io/assets/conv-demo/

ReLU Revisited

- Rectified Linear Unit $f(x) = \max(0, x)$

- How to evaluate its derivative?
- In practice we don't directly use it, but a soft version of it.
- Soft ReLu is called **SoftPlus** function.

$$f(x) = \ln(1 + e^x)$$

SoftPlus function

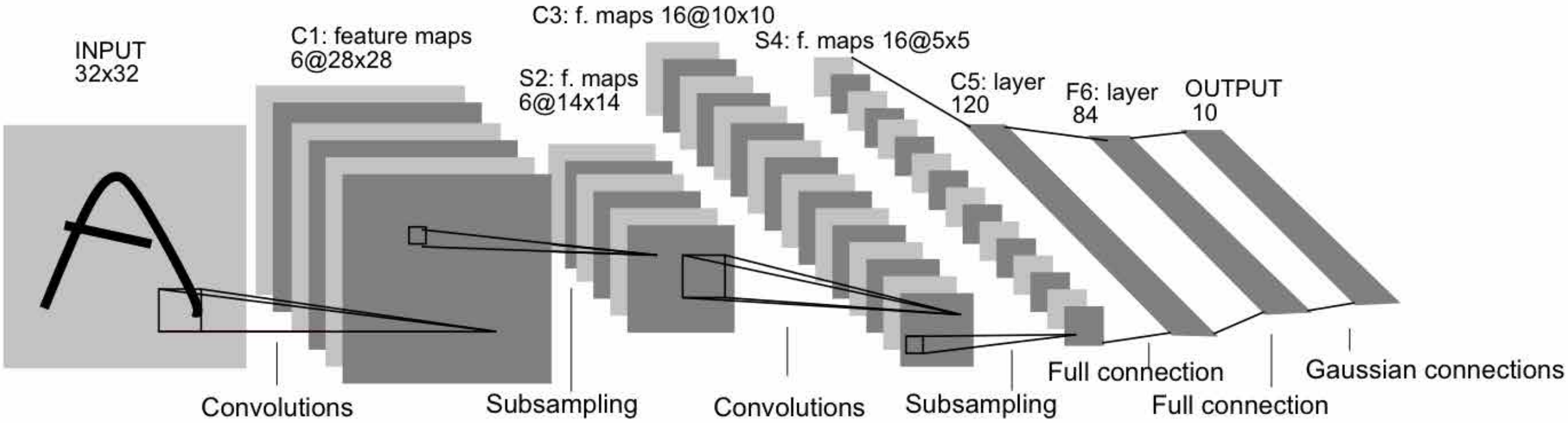
- SoftPlus function: $$f(x) = \max(0, x)$$

- The derivative of SoftPlus function is:

$$f'(x) = \frac{e^x}{(e^x + 1)} = \frac{1}{1 + e^{-x}}$$

- What is this function?
- Sigmoid!
- This enabled to train a deep neural network without unsupervised pre-training for the first time in 2011!

LeNet-5 Again



- Architecture of LeNet-5 (LeCun, 1998), a Convolutional Neural Network for digits recognition.
- Each plane is a feature map, a set of weights of a unique neuron applied over a local receptive field of previous layer's output.

The 82 errors made by LeNet5

Notice that most of the errors are cases that people find quite easy.

The human error rate is probably 20 to 30 errors but nobody has had the patience to measure it.

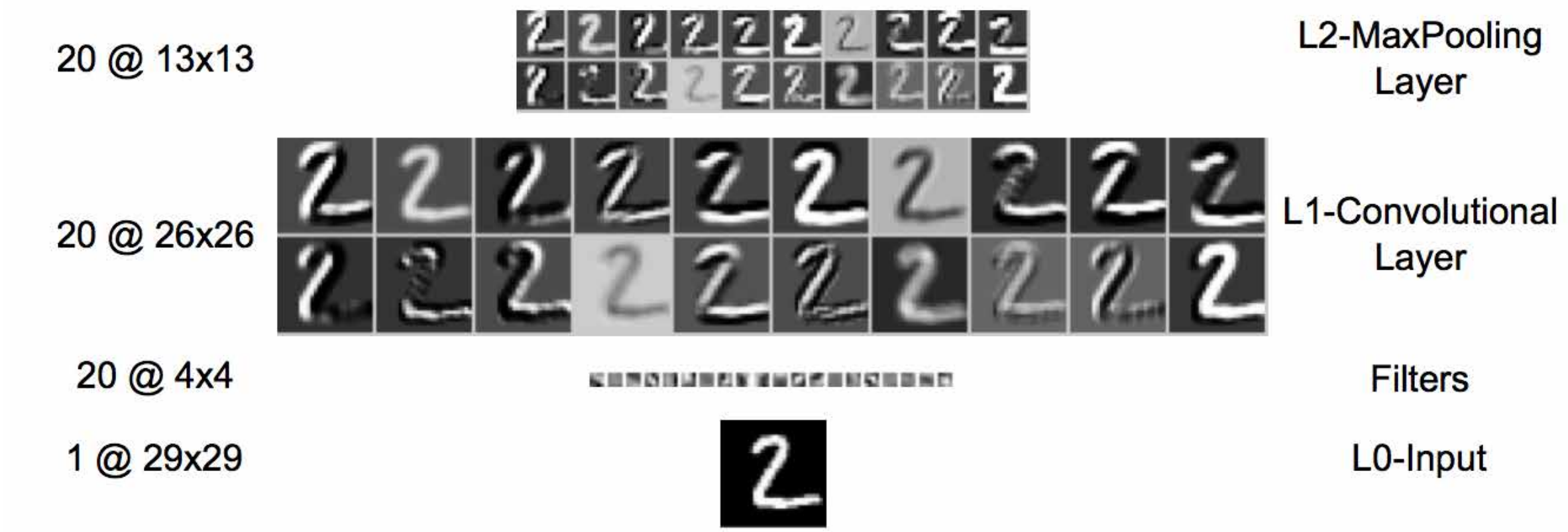
Multi-column Deep Neural Networks for Image Classification

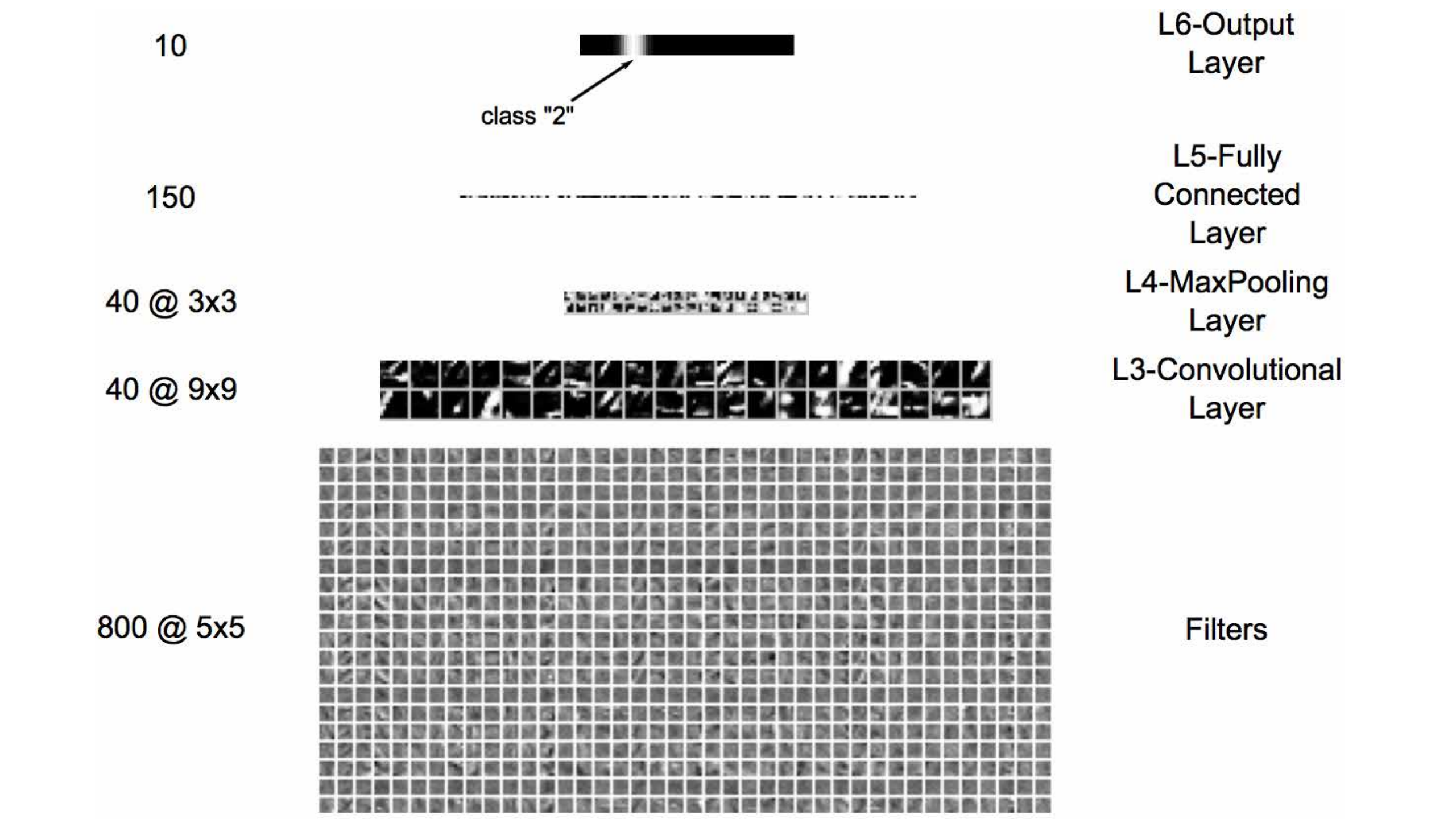
Dan Cireşan, Ueli Meier and Jürgen Schmidhuber

- Input batch and 4 different augmentations are fed in each iteration

Multi-column Deep Neural Networks for Image Classification

Dan Cireşan, Ueli Meier and Jürgen Schmidhuber




10
class "2"
L6-Output Layer
150
L5-Fully Connected Layer
40 @ 3x3
L4-MaxPooling Layer
40 @ 9x9
L3-Convolutional Layer
800 @ 5x5
Filters

The 23 errors made by the Cireşan *et. al.* net

- The top printed digit is the right answer. The bottom two printed digits are the network's best two guesses.
- The right answer is almost always in the top 2 guesses.

How to detect a significant drop in the error rate

- Is 30 errors in 10,000 test cases significantly better than 40 errors?
  - It all depends on the particular errors!
  - The McNemar test uses the particular errors and can be much more powerful than a test that just uses the number of errors.

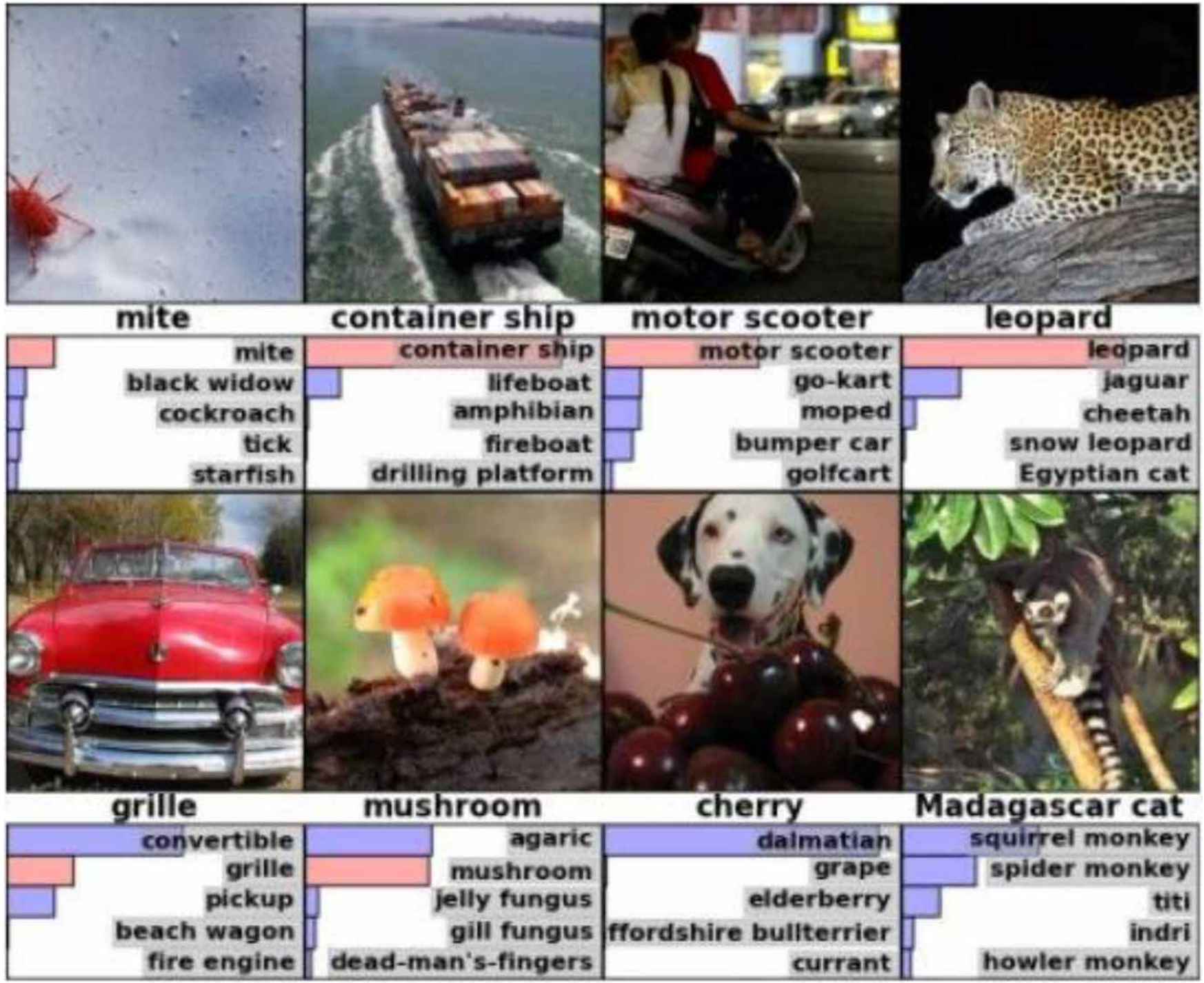
	model 1 wrong	model 1 right
model 2 wrong	29	1
model 2 right	11	9959

	model 1 wrong	model 1 right
model 2 wrong	15	15
model 2 right	25	9945

ImageNet Large Scale Visual Recognition Competition (ILSVRC)

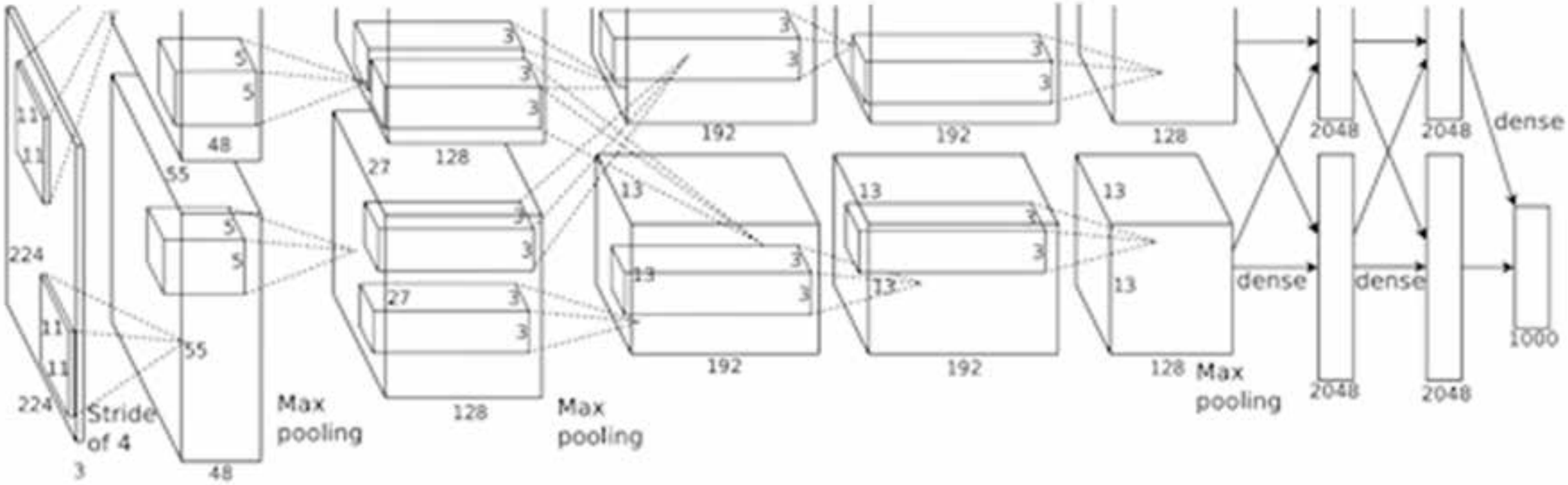
ImageNet Large Scale Visual Recognition Competition (ILSVRC)

- ~ 1 million images
- 1000 object categories in the training set
- Task: What is the object in the image?
  - Classify the image into one of 1000 categories
- Evaluation
  - Is one of the best 5 guesses is correct?
- **Human Performance is around 5.1% error.**


mite
container ship
motor scooter
leopard
mite
black widow
cockroach
tick
starfish
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
go-kart
moped
bumper car
golfcart
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
mushroom
cherry
Madagascar cat
convertible
grille
pickup
beach wagon
fire engine
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
dalmatian
grape
elderberry
ffordshire bullterrier
currant
squirrel monkey
spider monkey
titi
indri
howler monkey

AlexNet (2012)

The one that started all again!



AlexNet architecture (May look weird because there are two different "streams". This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs)

AlexNet (2012)

The one that started all again!

By

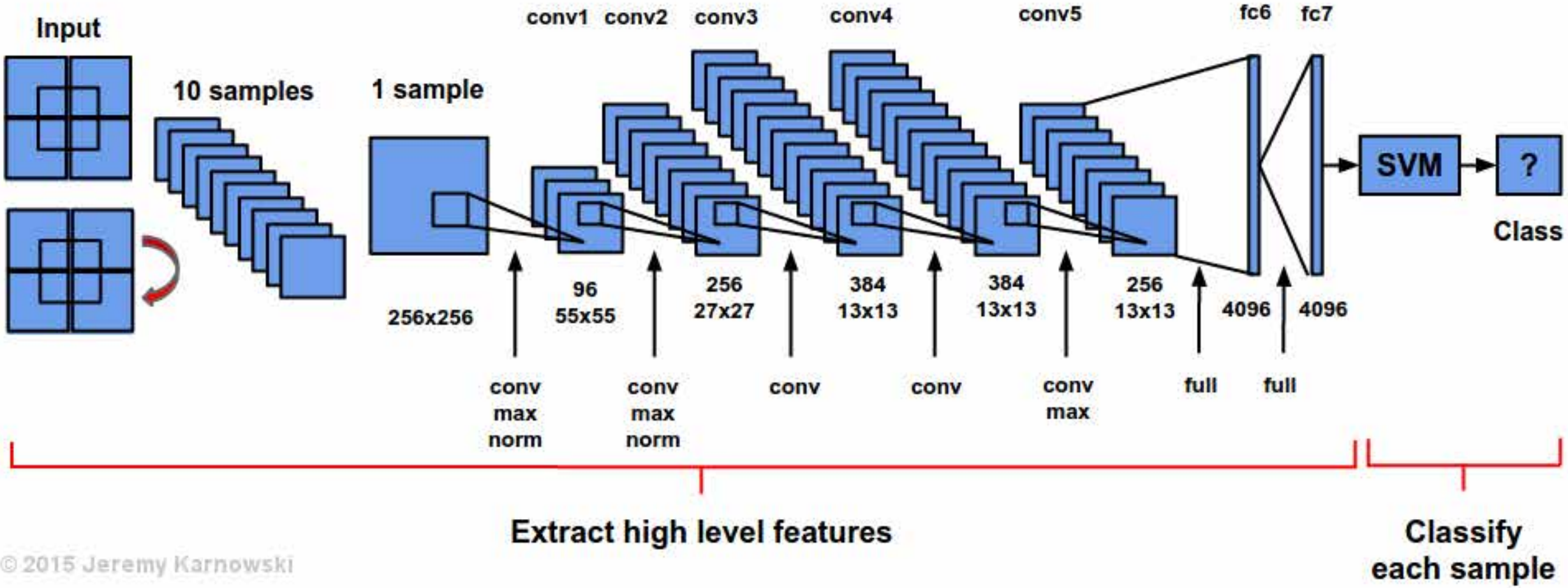
(Alex) Krizhevsky, Sutskever, and **Hinton**

Top-1 error: 37.5% (previous best was %47.1)

Top-5 error: 17.0% (previous best was %28.2)

AlexNet (2012)

Another variant of AlexNet with SVM at the end



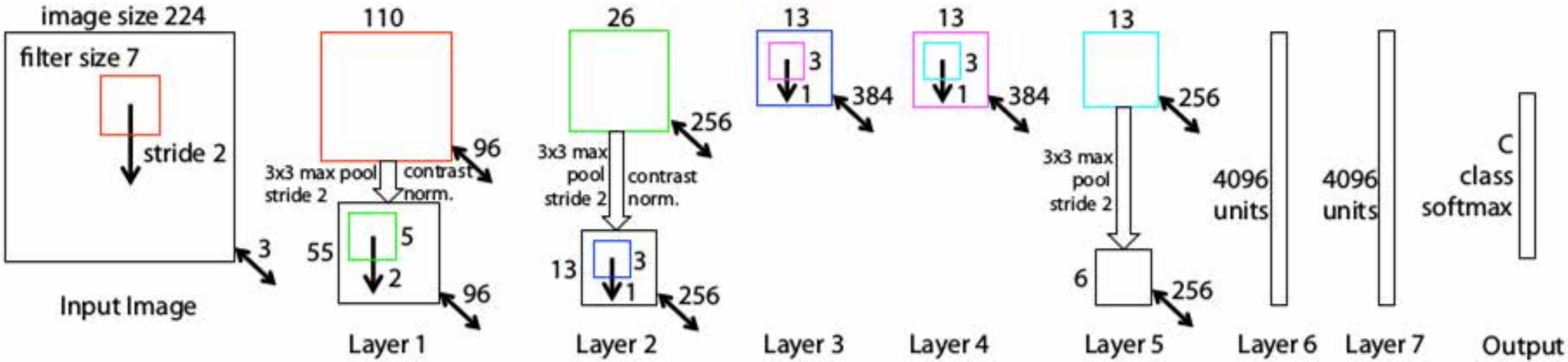
AlexNet (2012)

- Trained the network on **ImageNet** data, which contained over **15 million annotated images** from a total of over **22,000 categories**.
- Used **ReLU** for the nonlinearity functions (Found to decrease training time as ReLUs are several times faster than the conventional tanh function).
- Used **data augmentation** techniques that consisted of image translations, horizontal reflections, and patch extractions.
- Implemented **dropout layers** in order to combat the problem of overfitting to the training data.
- Trained the model using **batch stochastic gradient descent**, with specific values for momentum and weight decay.
- Trained on two GTX 580 GPUs for **five to six days**.

AlexNet (2012)

- It was the debut of CNNs in the computer vision community.
- This was the first time a model performed so well on a historically difficult ImageNet dataset.
- Utilizing techniques that are still used today, such as **data augmentation** and **dropout**, this paper really illustrated the benefits of CNNs and backed them up with record breaking performance in the competition.

ZFNet (2013)



ZF Net Architecture

ZFNet (2013)

By

Matthew Zeiler and Rob Fergus

ZFNet (2013)

- The architecture was more of a fine tuning to the previous AlexNet structure, but still developed some very keys ideas about improving performance.
- Another reason this was such a great paper is that the authors spent a good amount of time explaining a lot of the intuition behind ConvNets and showing how to visualize the filters and weights correctly.
- The main contributions of this paper are details of a slightly modified AlexNet model and a very interesting way of visualizing feature maps.

- This model achieved an 11.2% error rate.

ZFNet (2013)

- Very similar architecture to AlexNet, except for a few minor modifications.
- AlexNet trained on 15 million images, while ZF Net trained on only 1.3 million images.
- Instead of using 11x11 sized filters in the first layer (which is what AlexNet implemented), ZF Net used filters of size 7x7 and a decreased stride value.
- The reasoning behind this modification is that a smaller filter size in the first conv layer helps retain a lot of original pixel information in the input volume. A filtering of size 11x11 proved to be skipping a lot of relevant information, especially as this is the first conv layer.
- As the network grows, we also see a rise in the number of filters used.
- Trained on a GTX 580 GPU for **twelve days**.

VGGNet (2014)

By

Karen Simonyan,
Andrew Zisserman

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 imag)					
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					

The 6 different architecures of VGG Net. Configuration D produced the best results

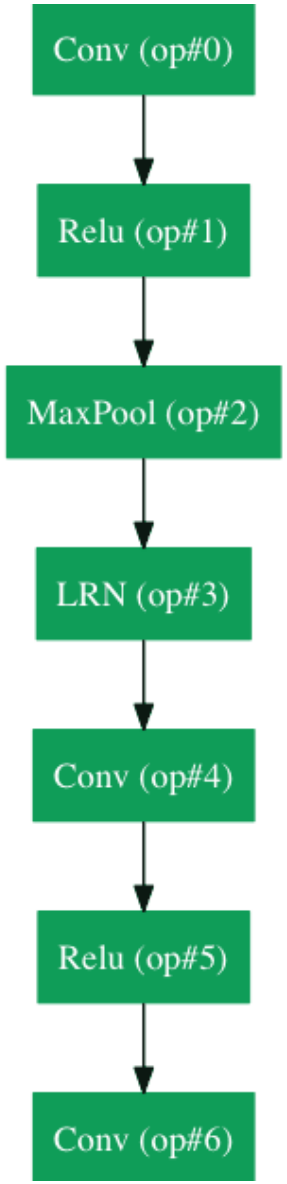
VGGNet (2014)

- Use of only 3x3 sized filters is quite different from AlexNet's 11x11 filters in the first layer and ZF Net's 7x7 filters.
- The authors' reasoning is that the combination of two 3x3 conv layers has an effective receptive field of 5x5. This in turn simulates a larger filter while keeping the benefits of smaller filter sizes.
- One of the benefits is a decrease in the number of parameters. Also, with two conv layers, we're able to use two ReLU layers instead of one.
- 3 conv layers back to back have an effective receptive field of 7x7.
- As the spatial size of the input volumes at each layer decrease (result of the conv and pool layers), the depth of the volumes increase due to the increased number of filters as you go down the network.

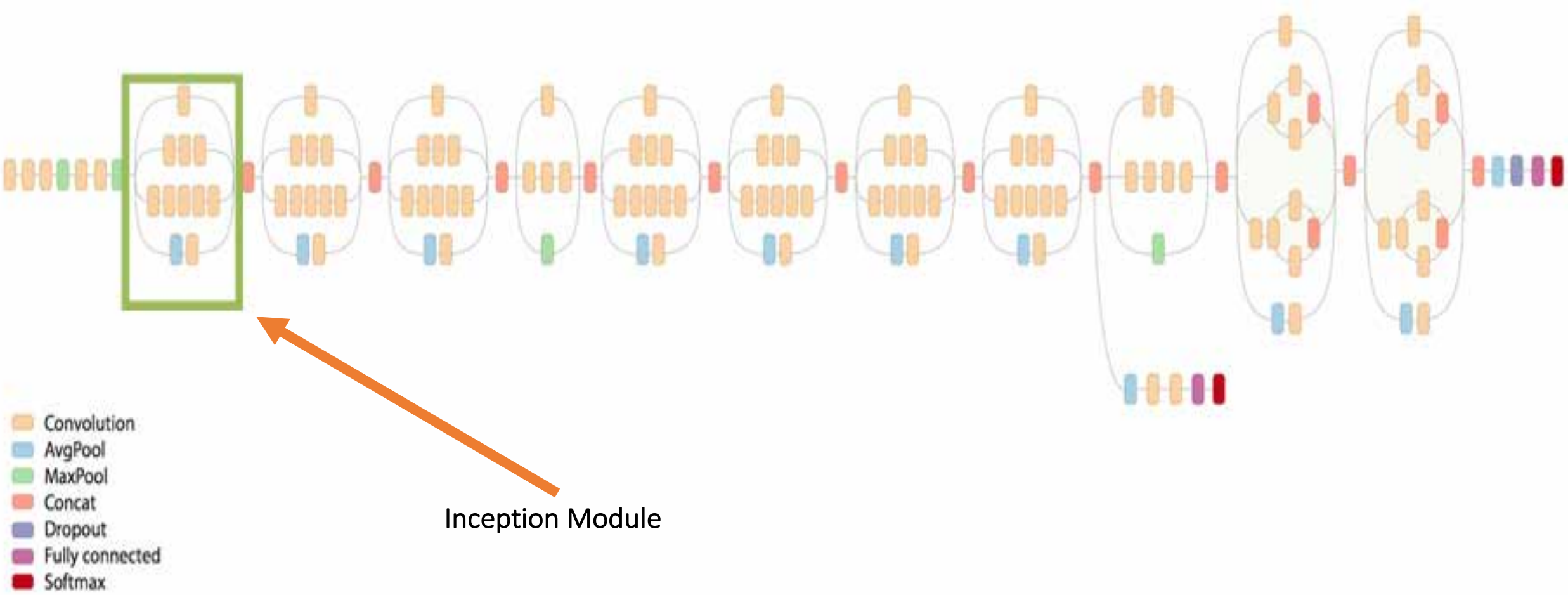
VGGNet (2014)

- The number of filters doubles after each max-pool layer. This reinforces the idea of shrinking spatial dimensions, but growing depth.
- Used scale jittering as one data augmentation technique during training.
- Used ReLU layers after each conv layer and trained with batch gradient descent.
- Trained on 4 nVidia Titan Black GPUs for **two to three weeks**.
- VGG Net is one of the most influential papers because it reinforced the notion that convolutional neural networks have to have a deep network of layers in order for this hierarchical representation of visual data to work. Keep it deep. Keep it simple.

GoogLeNet (2015)

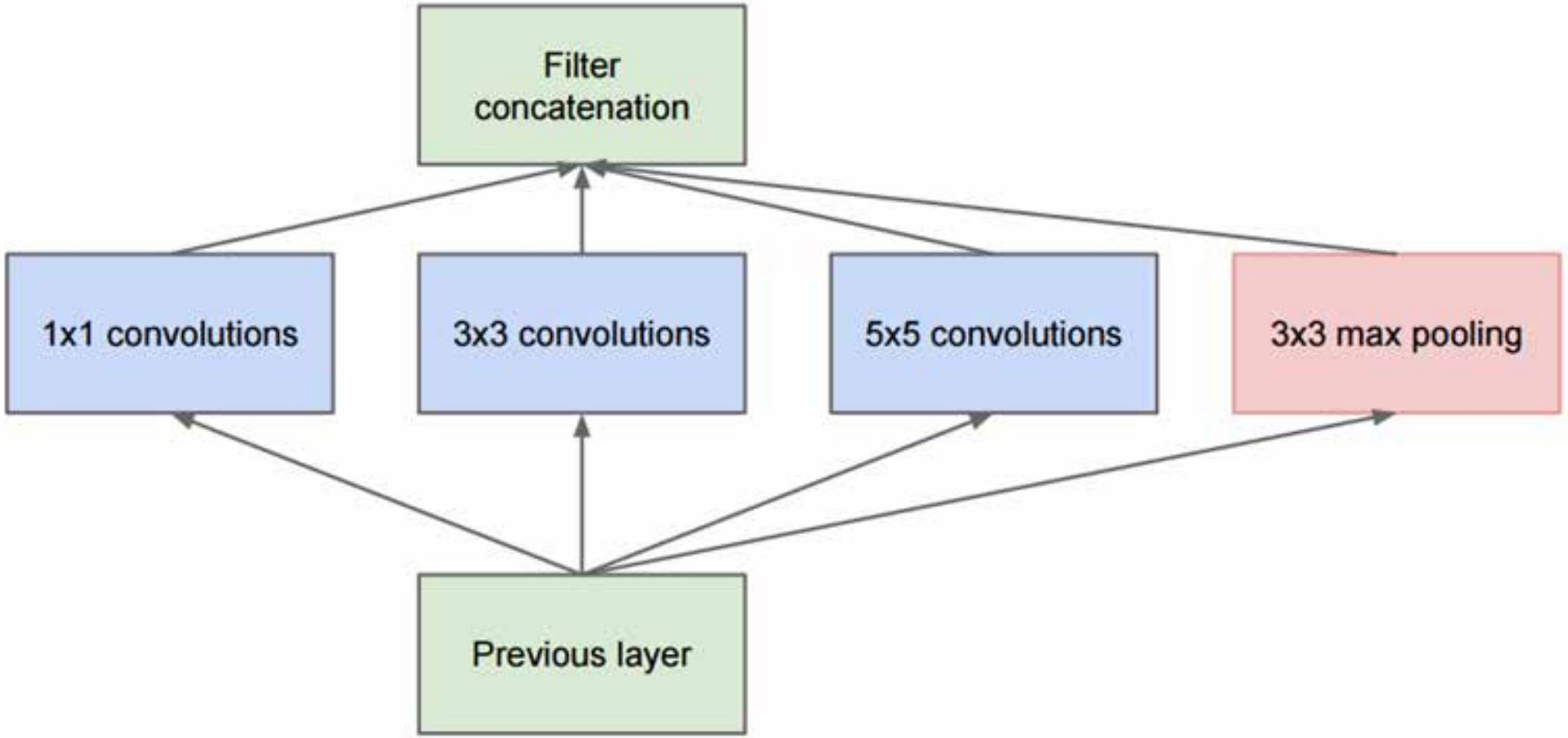


GoogLeNet (2015)



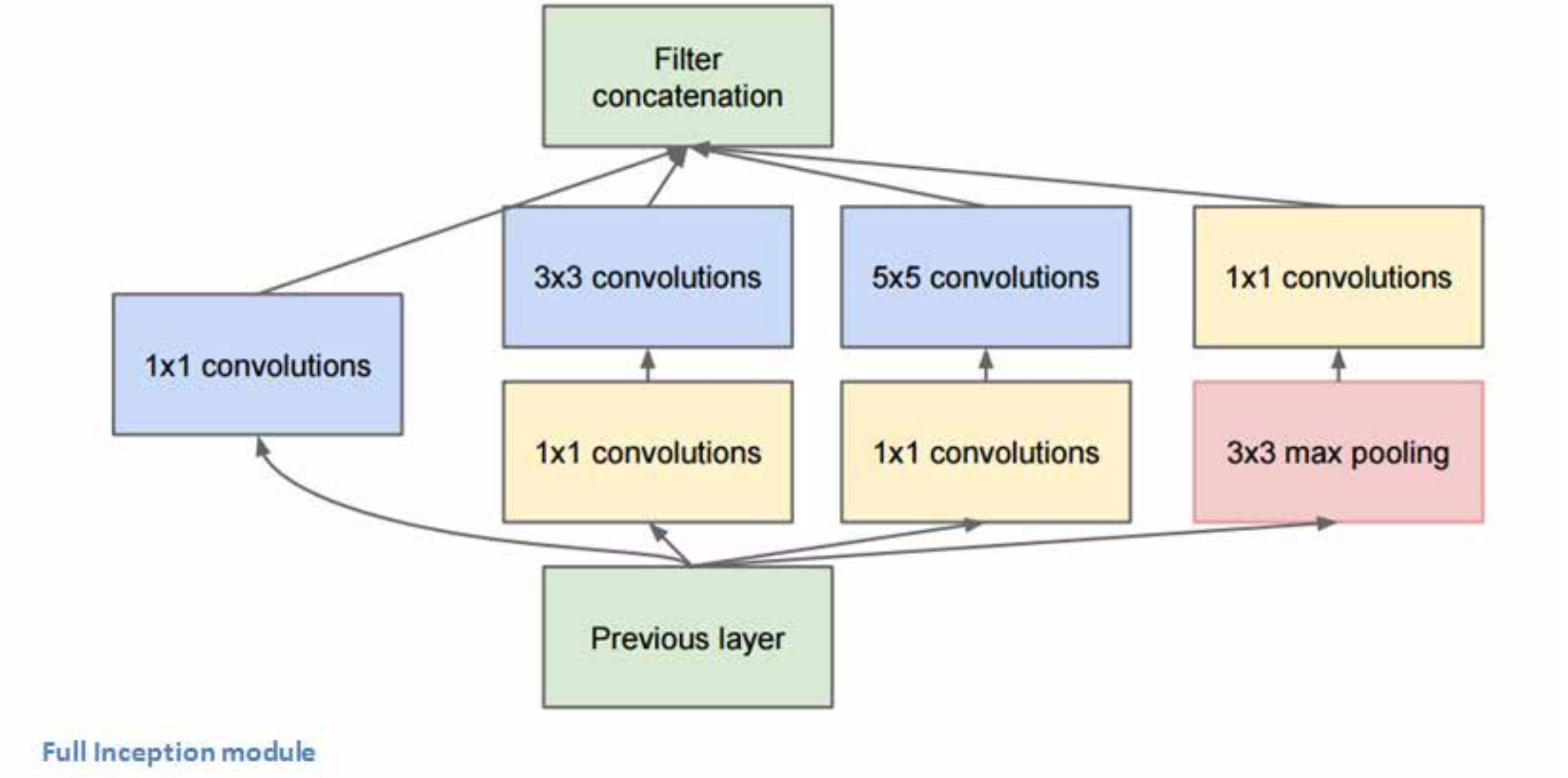
Green box shows parallel region of GoogLeNet

GoogLeNet (2015)



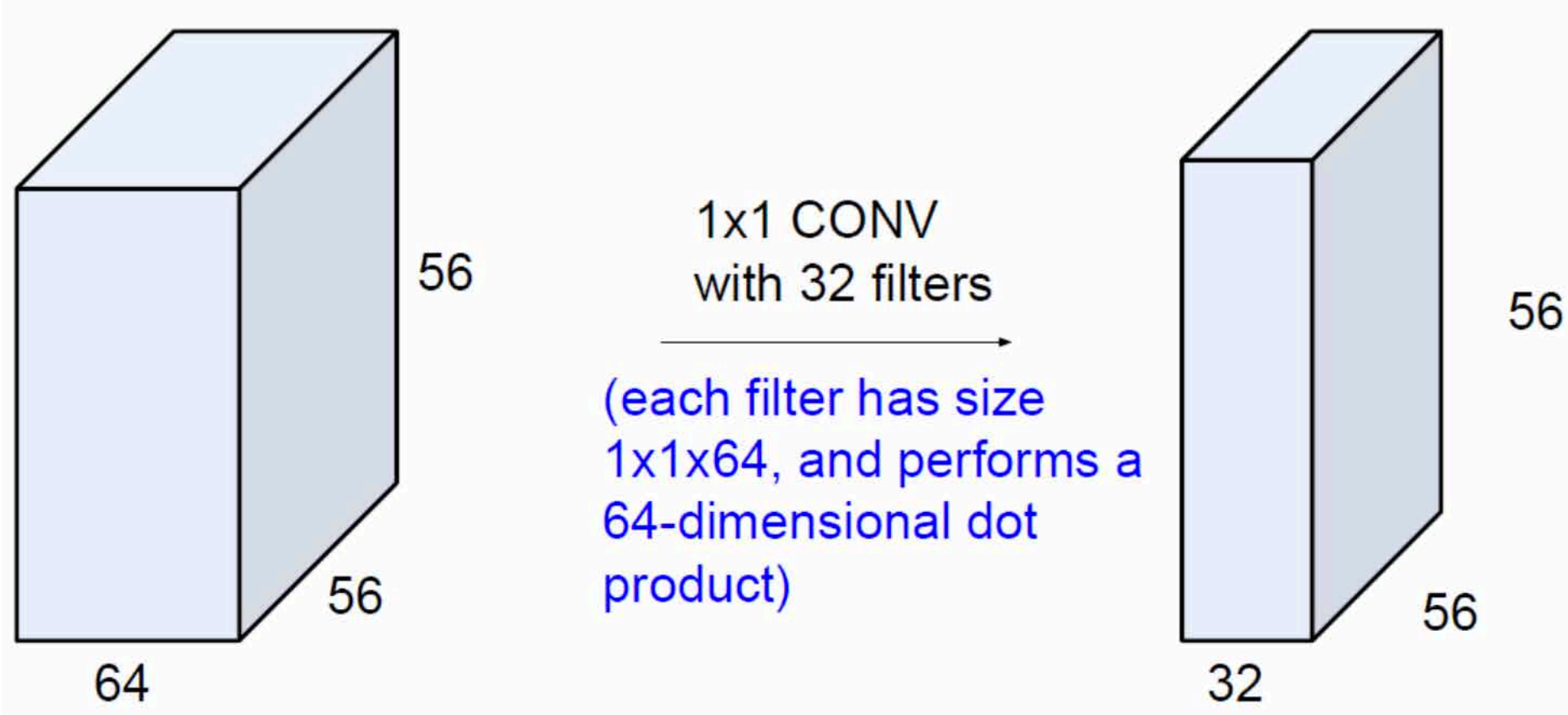
Naïve idea of an Inception module

GoogLeNet (2015)

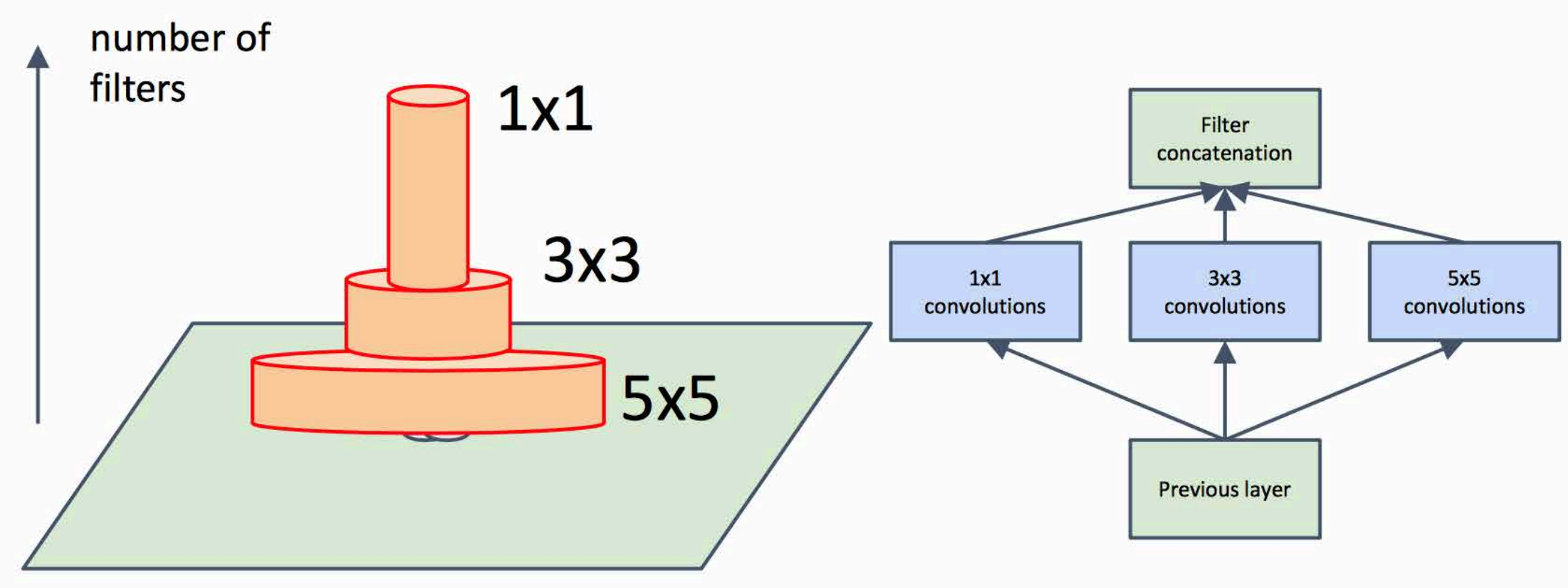


Full Inception module

Convolutional Layers Revisited:
1x1 convolutions?



GoogLeNet (2015)



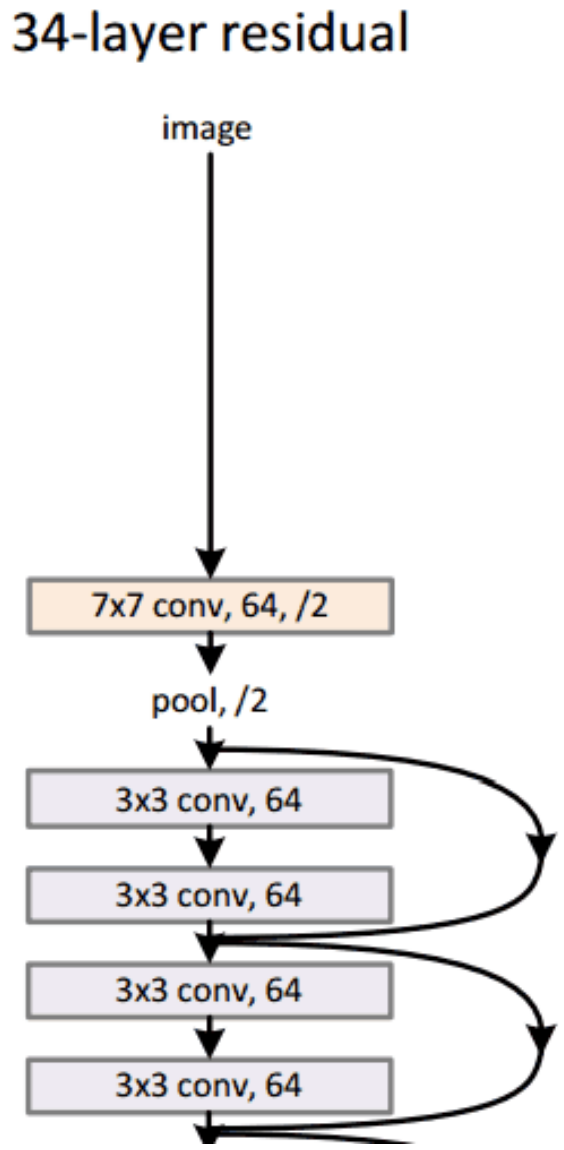
GoogLeNet (2015)

- Used 9 Inception modules in the whole architecture, with over 100 layers in total!
- No use of fully connected layers! They use an average pool instead, to go from a 7x7x1024 volume to a 1x1x1024 volume. This saves a huge number of parameters.
- Uses 12x fewer parameters than AlexNet.
- There are updated versions to the Inception module (Versions 6 and 7).
- Trained on “a few high-end GPUs **within a week**”.
- **6.7% error on ImageNet**

GoogLeNet (2015)

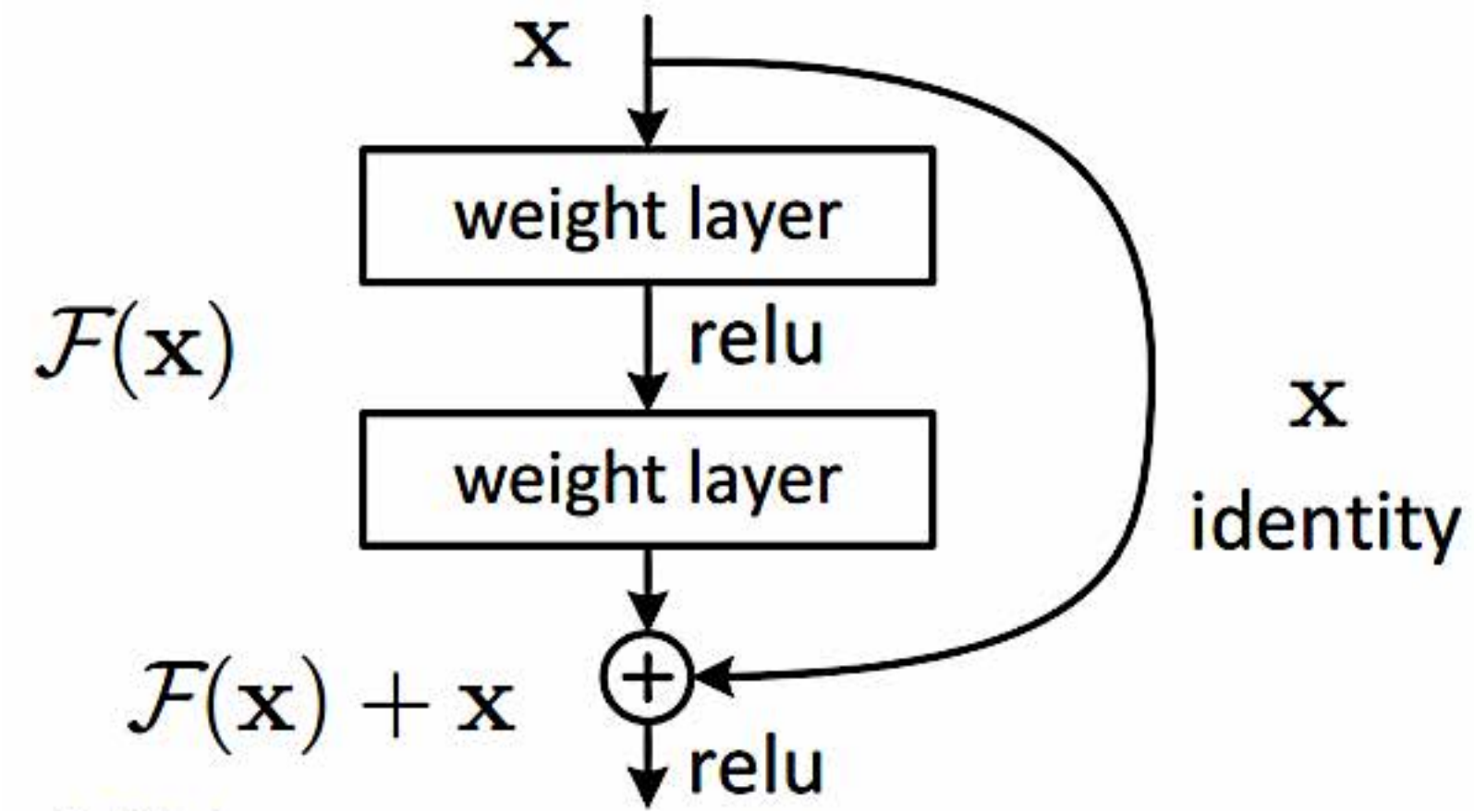
- GoogLeNet was one of the first models that introduced the idea that CNN layers didn't always have to be stacked up sequentially.
- Coming up with the Inception module, the authors showed that a creative structuring of layers can lead to improved performance and computationally efficiency.
- This paper has really set the stage for some amazing architectures that we could see in the coming years.

Microsoft ResNet (2015)



Microsoft ResNet (2015)

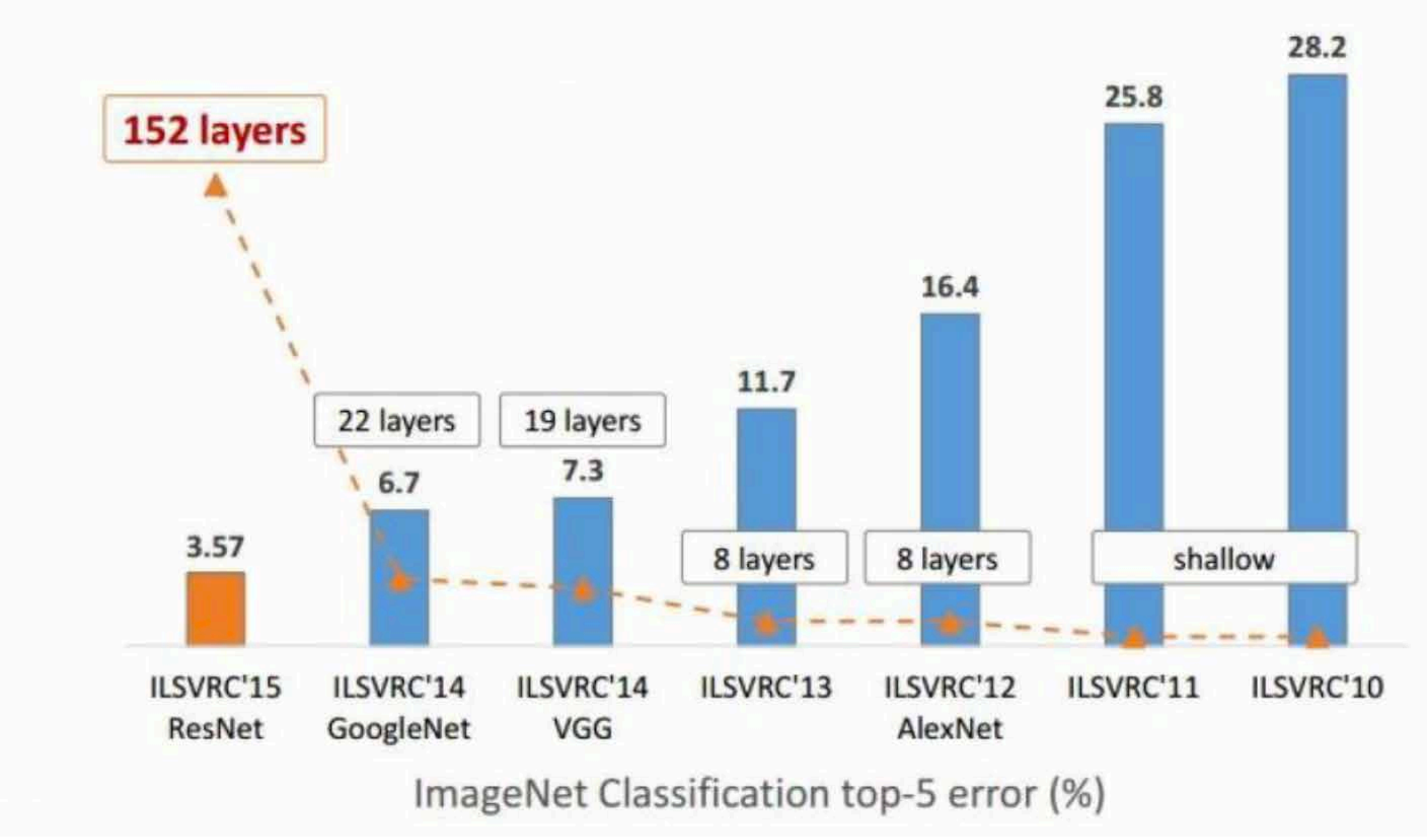
- Residual module (block)



Microsoft ResNet (2015)

- “Ultra-deep”. 152 layers…
- Interesting note that after only the *first 2* layers, the spatial size gets compressed from an input volume of 224x224 to a 56x56 volume.
- The group tried a 1202-layer network, but got a lower test accuracy, presumably due to overfitting.
- Trained on an 8 GPU machine for **two to three weeks**.
- 3.6% error rate!

Revolution of Depth



Trimps-Soushen, FAIR (2016)

Trimps-Soushen 0.02991% error

FAIR team 0.03031% error

Ensemble Learning…