Convolutional Neural Networks Architectures

Xiaolong Wang

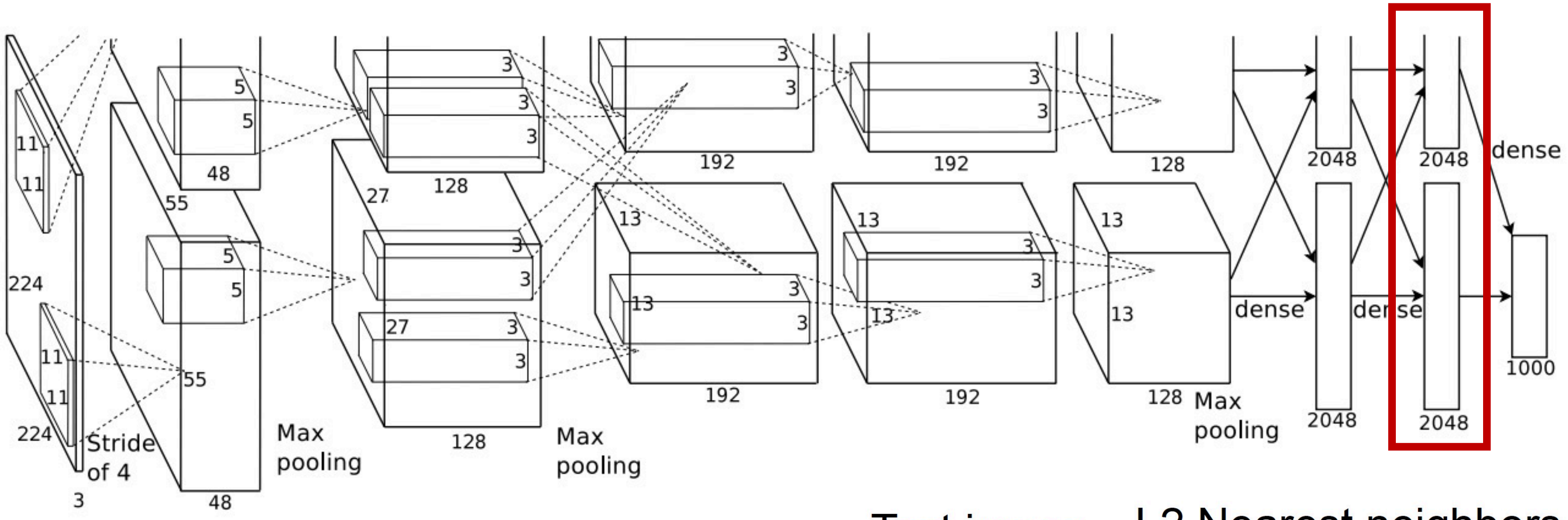
This Class

- Finetuning with CNN
- The developments and insights of CNN architectures

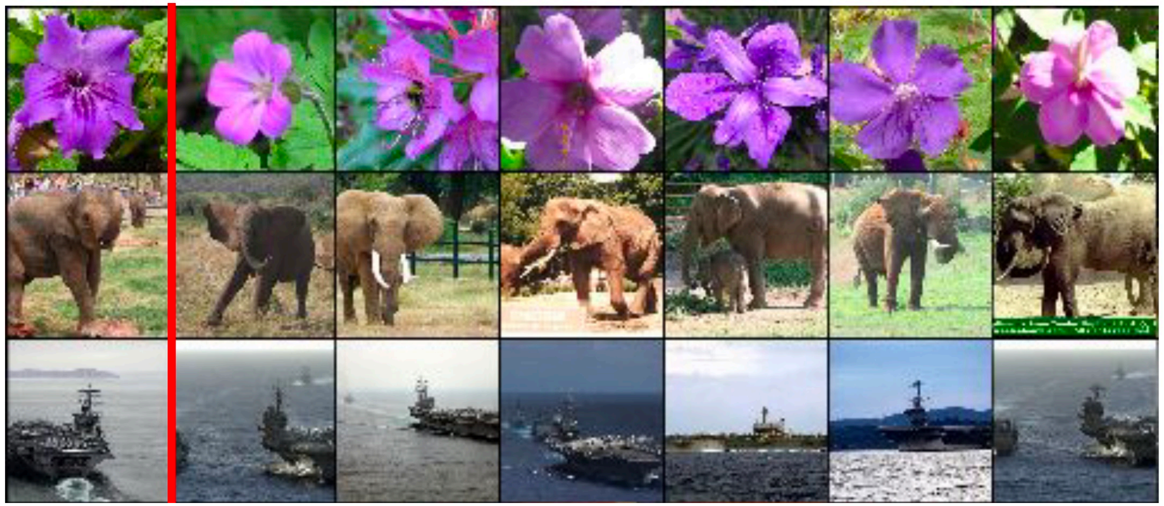
Slides partially from: http://cs231n.stanford.edu/

Finetuning CNN

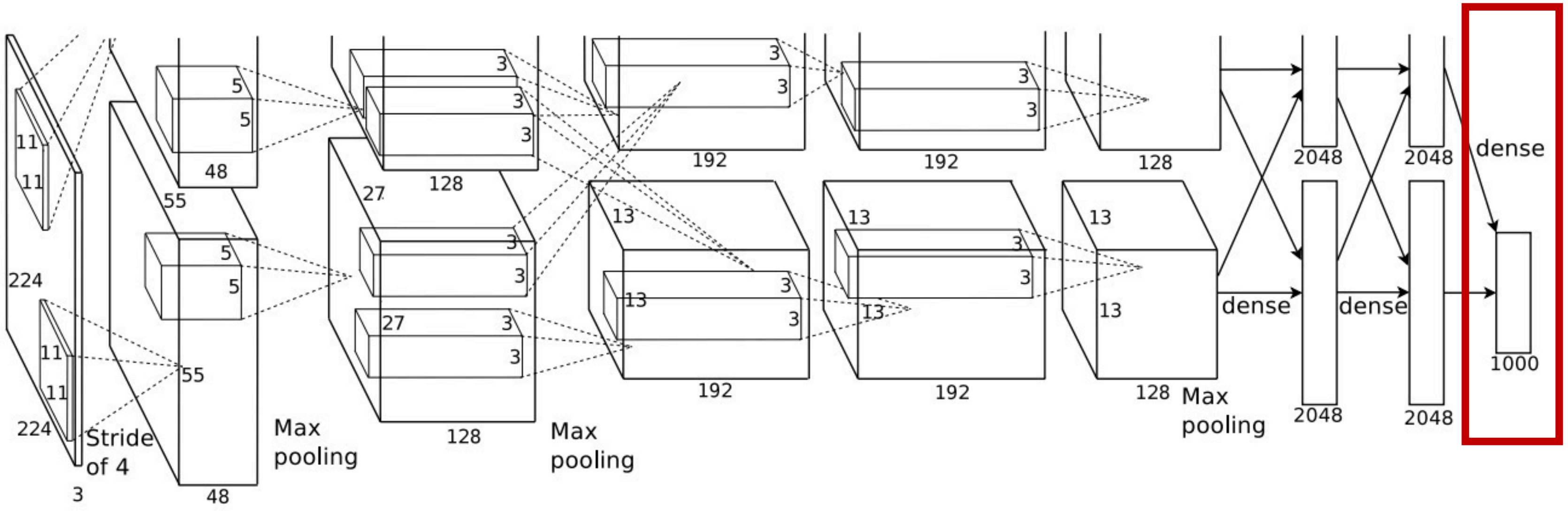
Given an ImageNet Pre-trained Network



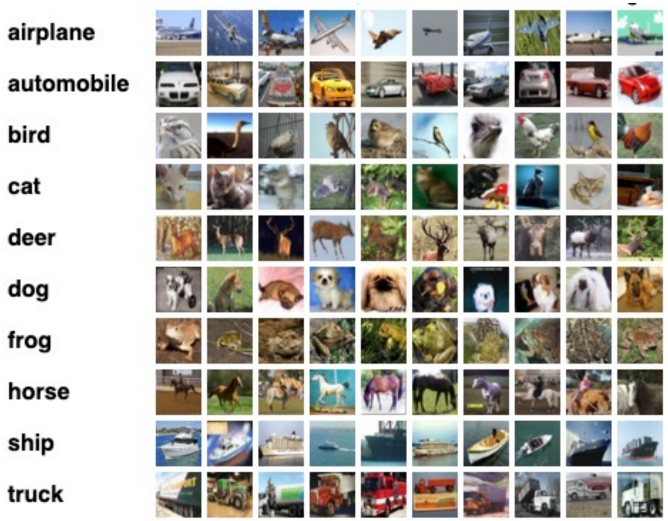
Test image L2 Nearest neighbors in <u>feature</u> space



Finetune an ImageNet Pre-trained Network

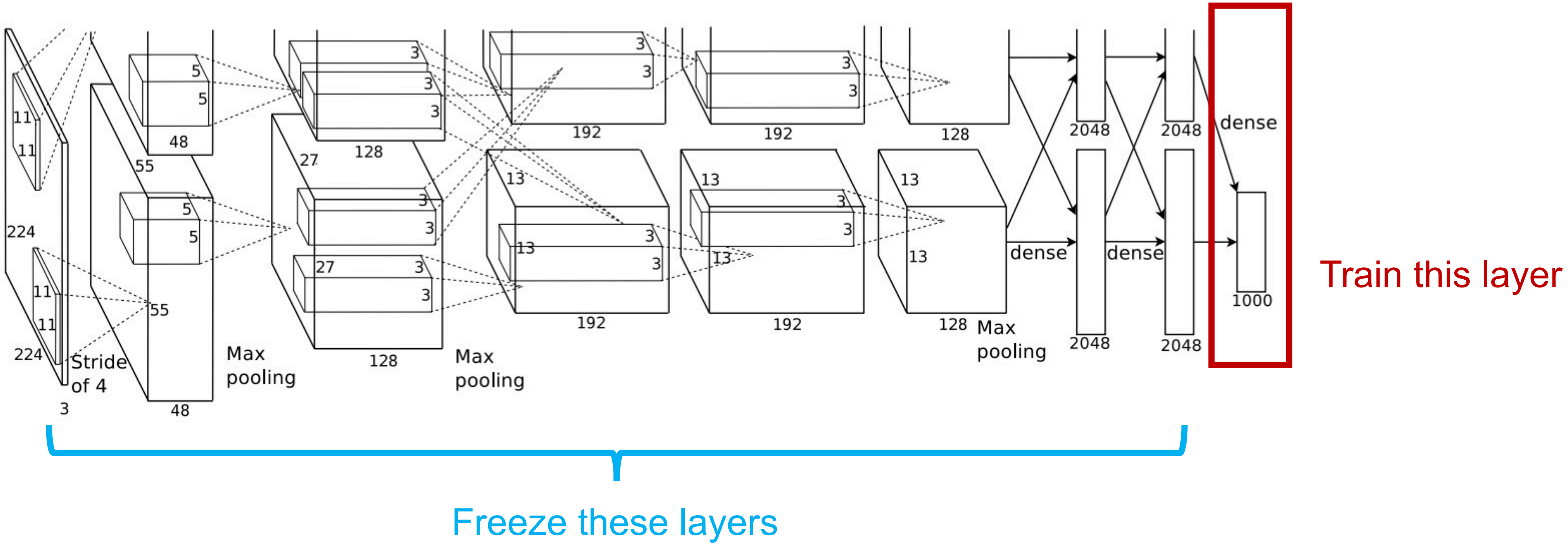


Remove and replace with another randomly initialized layer



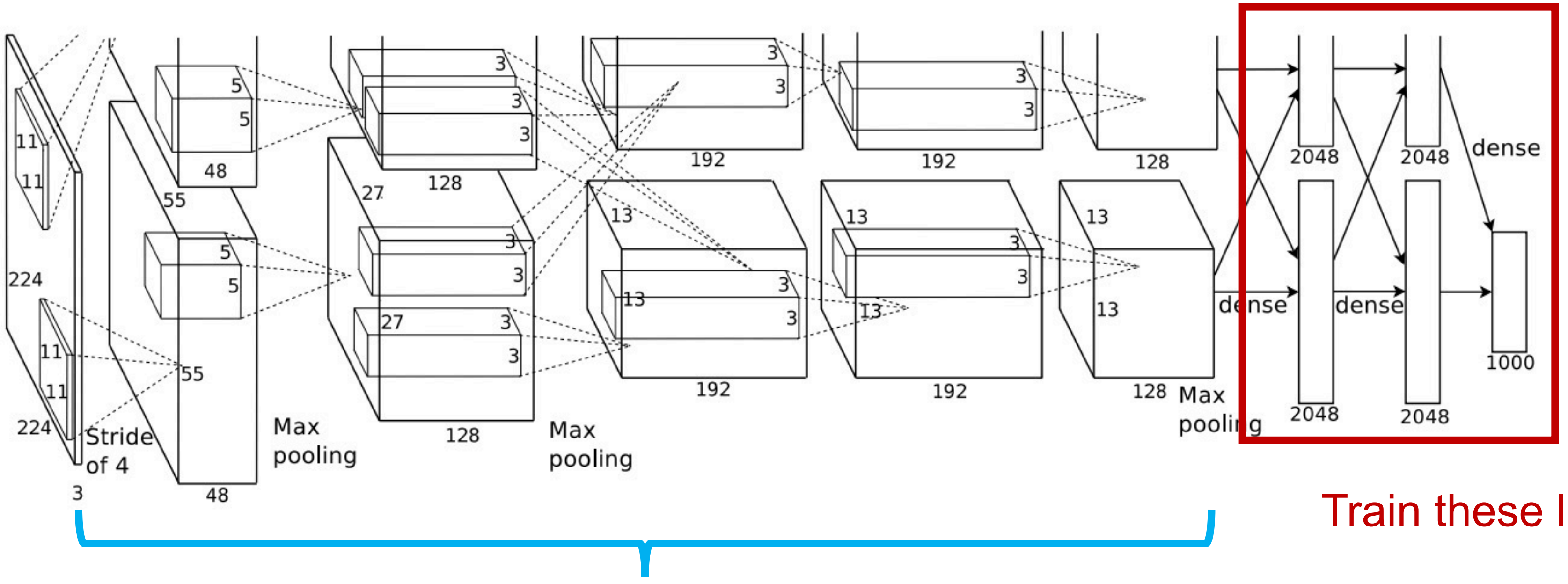
If the target dataset is Cifar-10, we should change the network output to 10 as well

Finetune an ImageNet Pre-trained Network



If we have a small dataset during fine-tuning, for example a dataset with a few hundred examples, we should freeze most layers.

Finetune an ImageNet Pre-trained Network

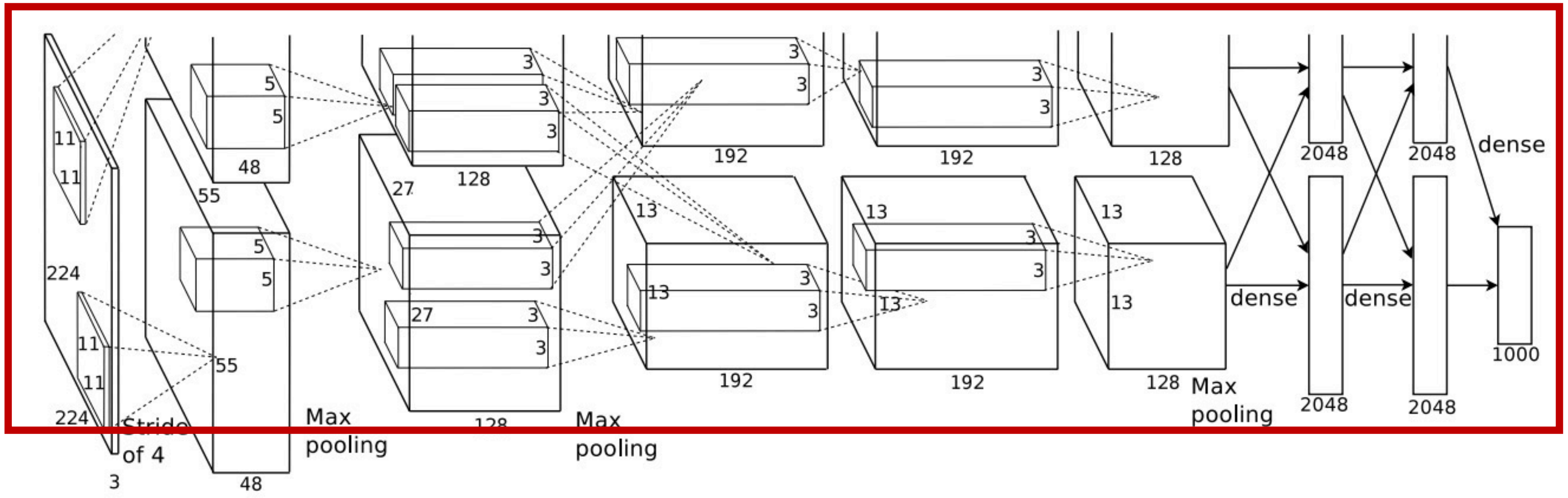


Train these layers (only re-initialize the last layer)

Freeze these layers

If we have a relatively larger dataset, for example a dataset with thousands of examples, we can tune more layers.

Finetune an ImageNet Pre-trained Network



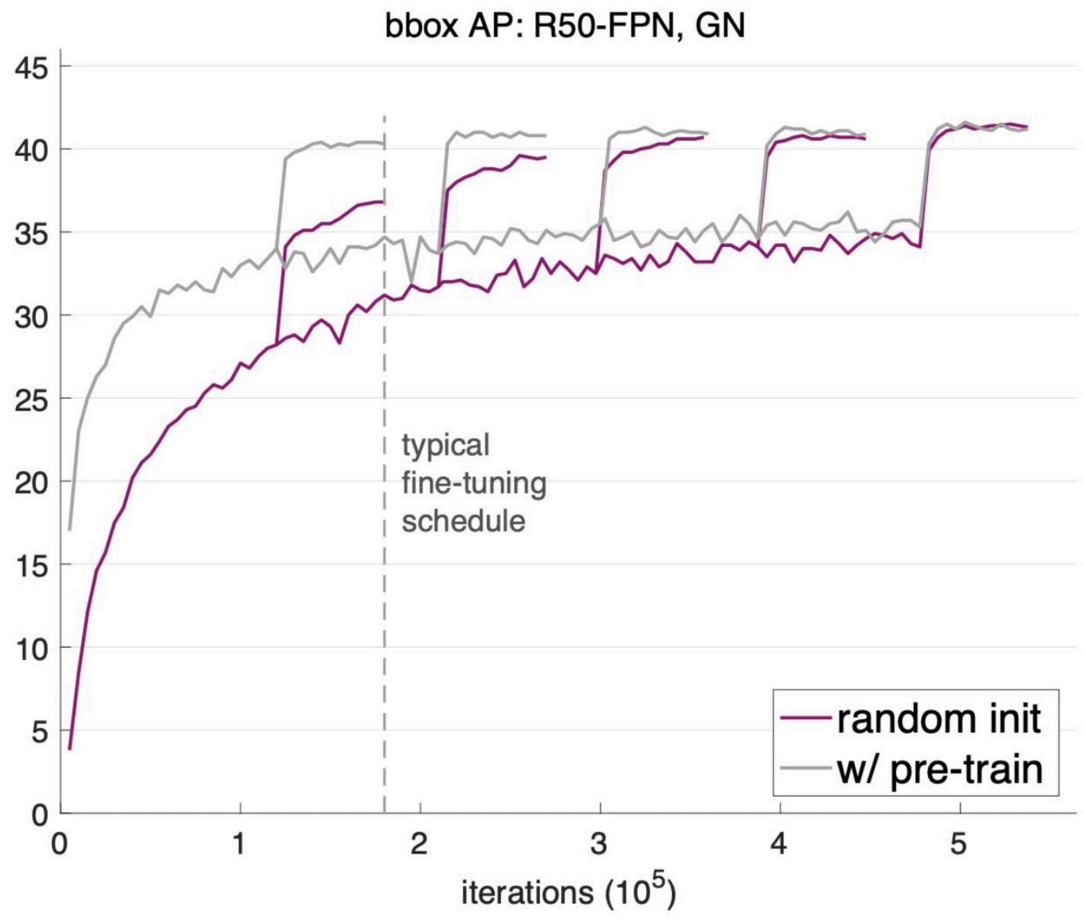
Train these layers (only re-initialize the last layer)

If we have a large dataset, for example a dataset with tens of thousands of examples, we can tune all layers.

Fine-tuning and Pre-training

- When fine-tuning, try to see how many layers you need to tune, it is task dependent.

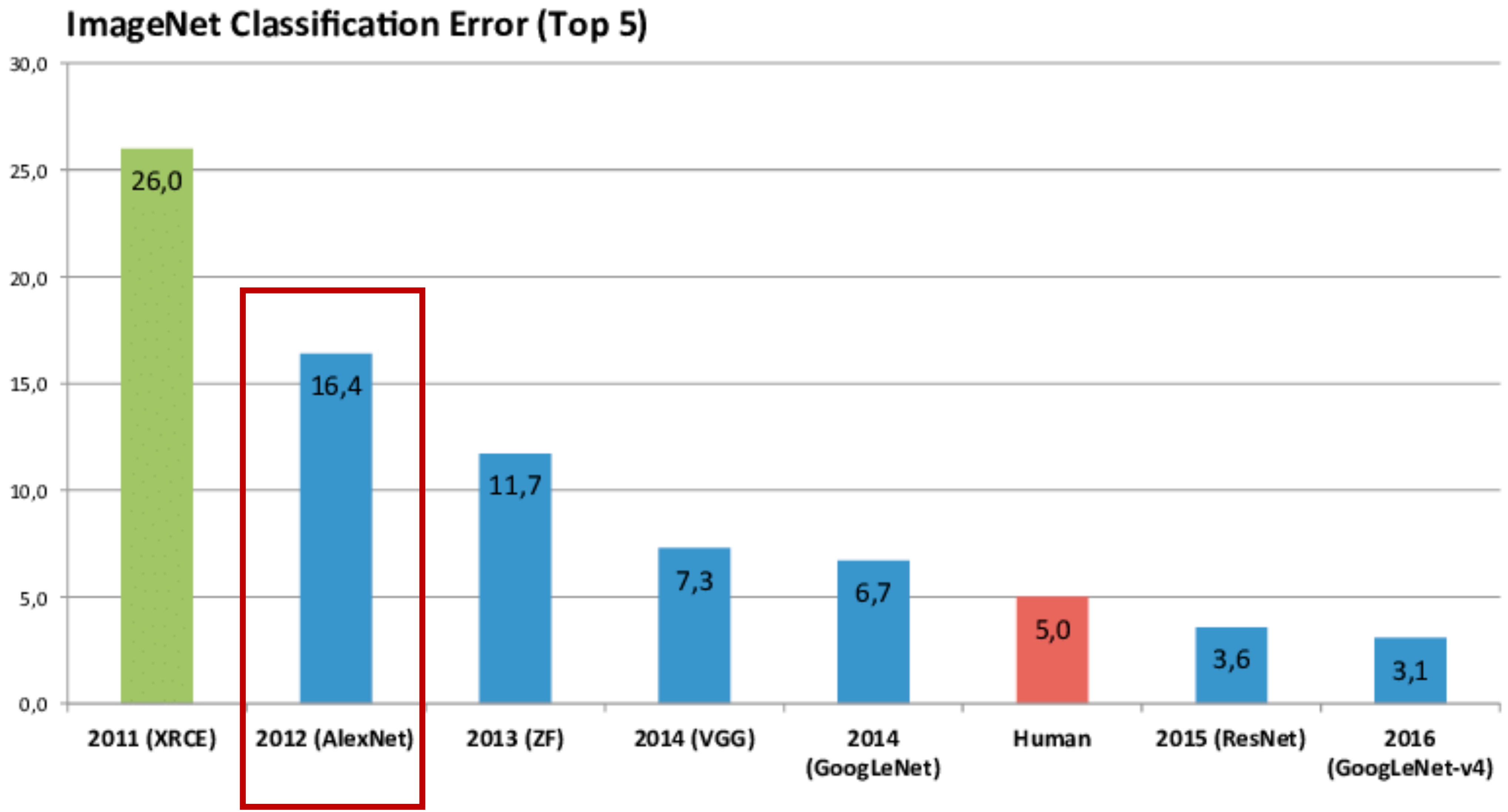
- If you have a very large dataset to fine-tune on already, the pre-training step might not be always necessary.



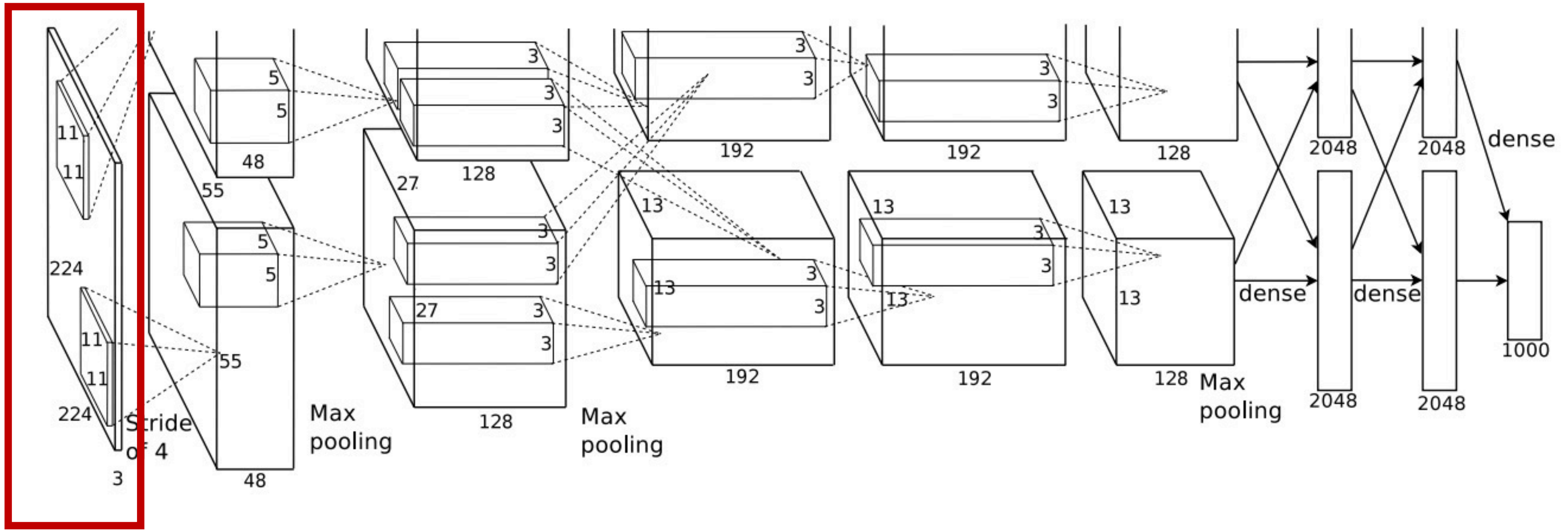
He et al, “Rethinking ImageNet Pre-training”, ICCV 2019

CNN Architectures

ImageNet Performance



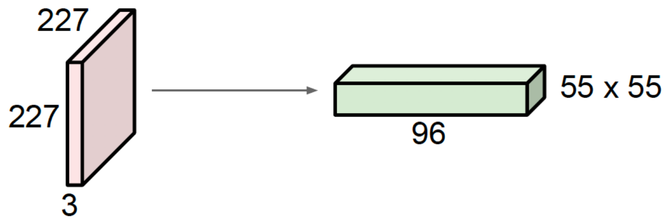
AlexNet



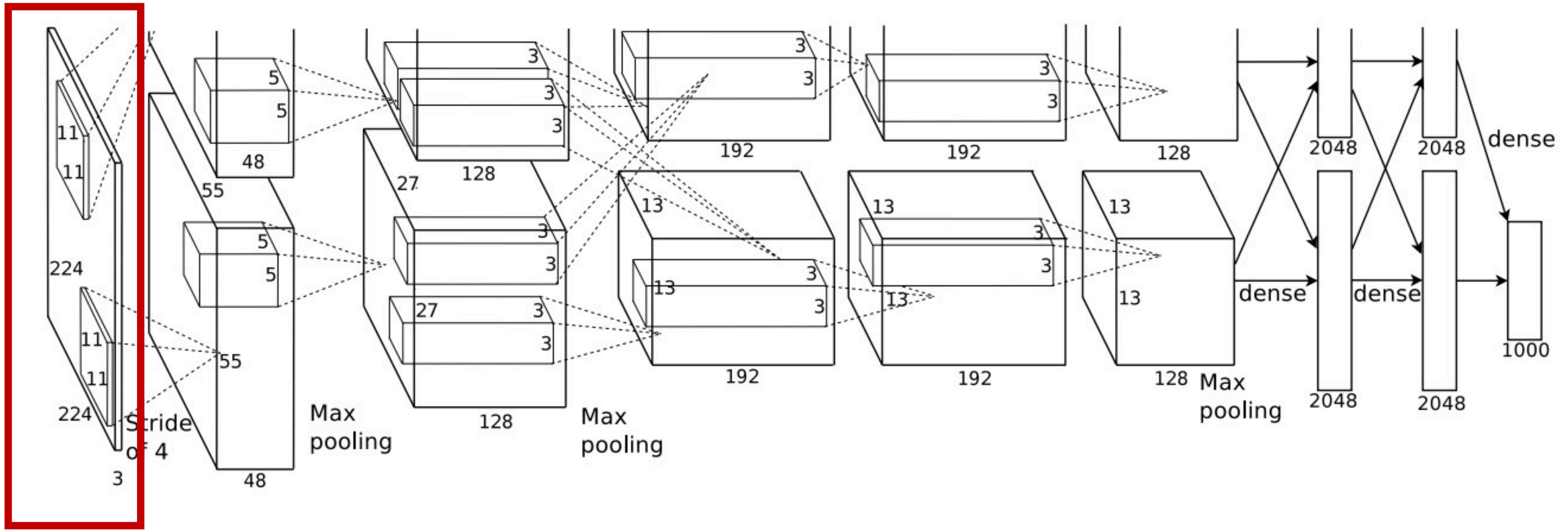
Conv1 -> Maxpool -> Conv2 -> Maxpool -> Conv3 -> Conv4 -> Conv5 -> Maxpool -> FC6 -> FC7 -> FC8

- Input: 227 x 227 x 3 image

- First layer (Conv1): 96 11x11 filters applied at stride 4
  - Output size of first layer: (227 - 11) / 4 + 1 = 55

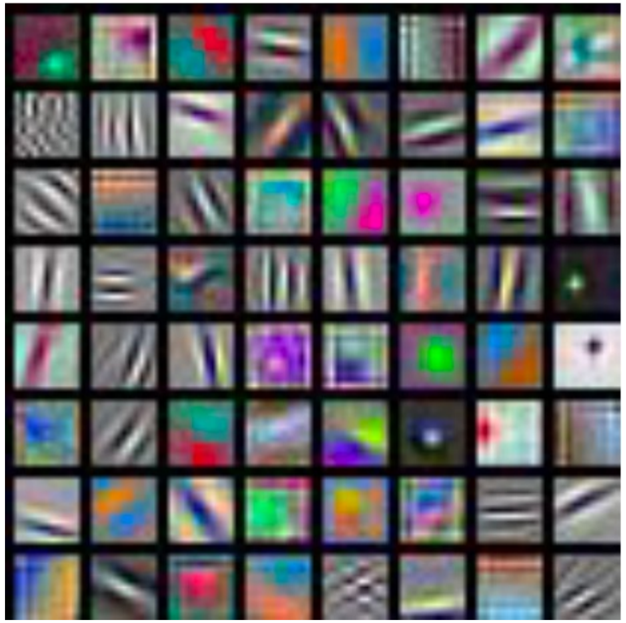


AlexNet

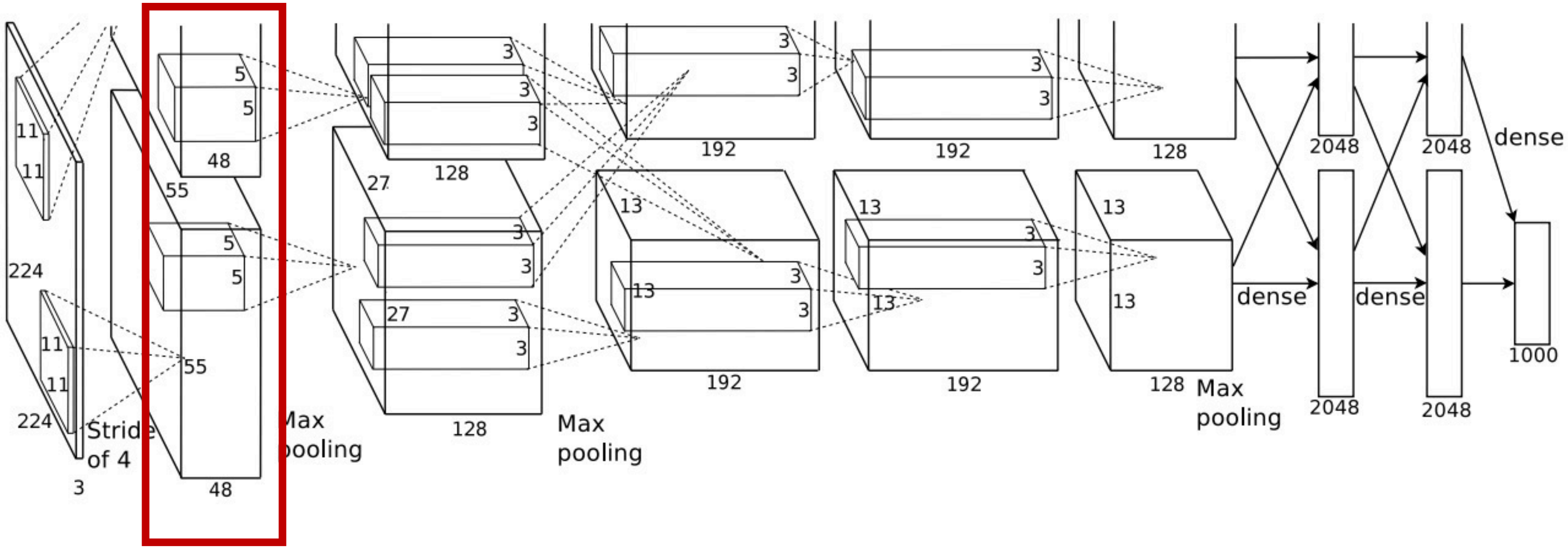


Conv1 -> Maxpool -> Conv2 -> Maxpool -> Conv3 -> Conv4 -> Conv5 -> Maxpool -> FC6 -> FC7 -> FC8

- Learned filters for Conv1

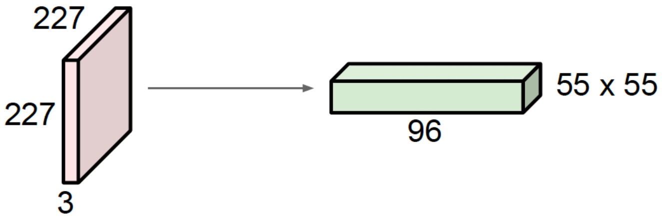


AlexNet



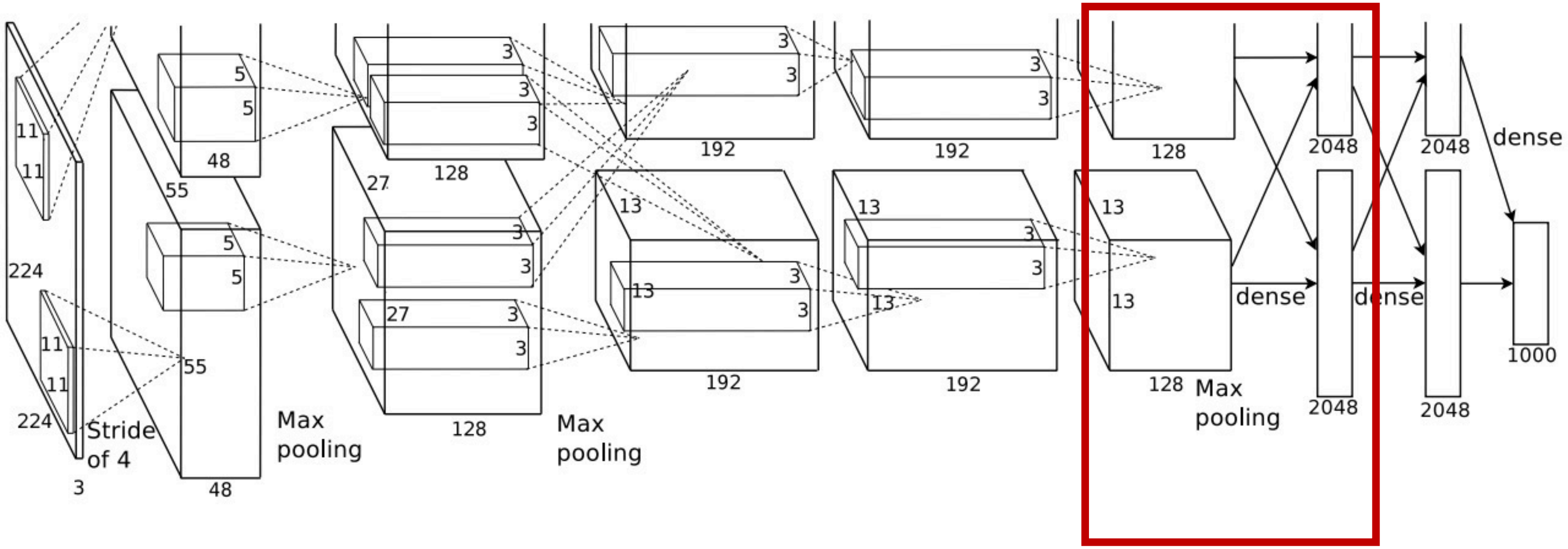
Conv1 -> Maxpool -> Conv2 -> Maxpool -> Conv3 -> Conv4 -> Conv5 -> Maxpool -> FC6 -> FC7 -> FC8

- Input: 55 x 55 x 96 feature map



- Second layer (Maxpool): 3 x 3 filters applied at stride 2
  - Output size of second layer: (55 - 3) / 2 + 1 = 27

AlexNet

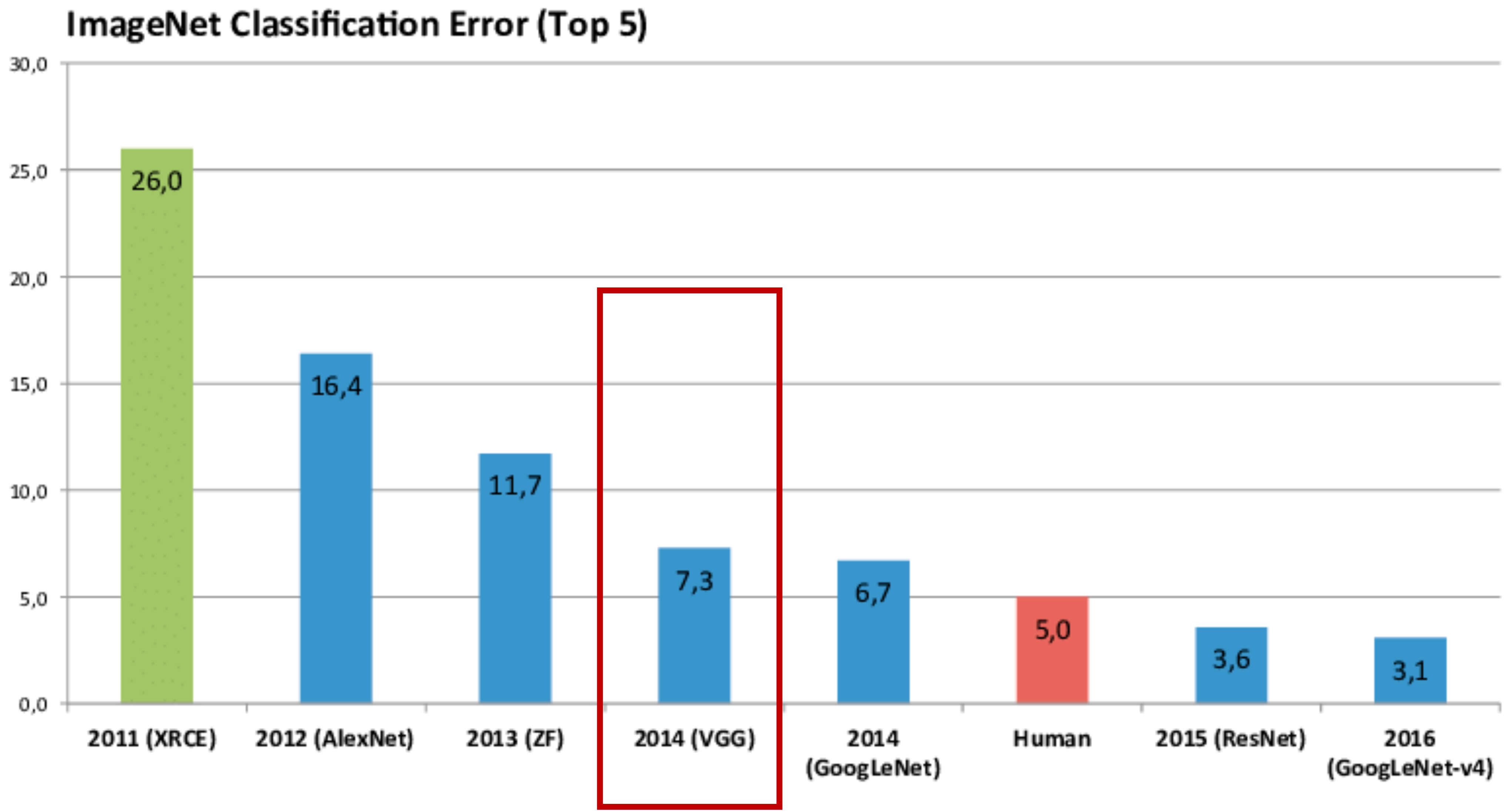


Conv1 -> Maxpool -> Conv2 -> Maxpool -> Conv3 -> Conv4 -> Conv5 -> Maxpool -> FC6 -> FC7 -> FC8

- Input for FC6: 6 x 6 x 256 feature map

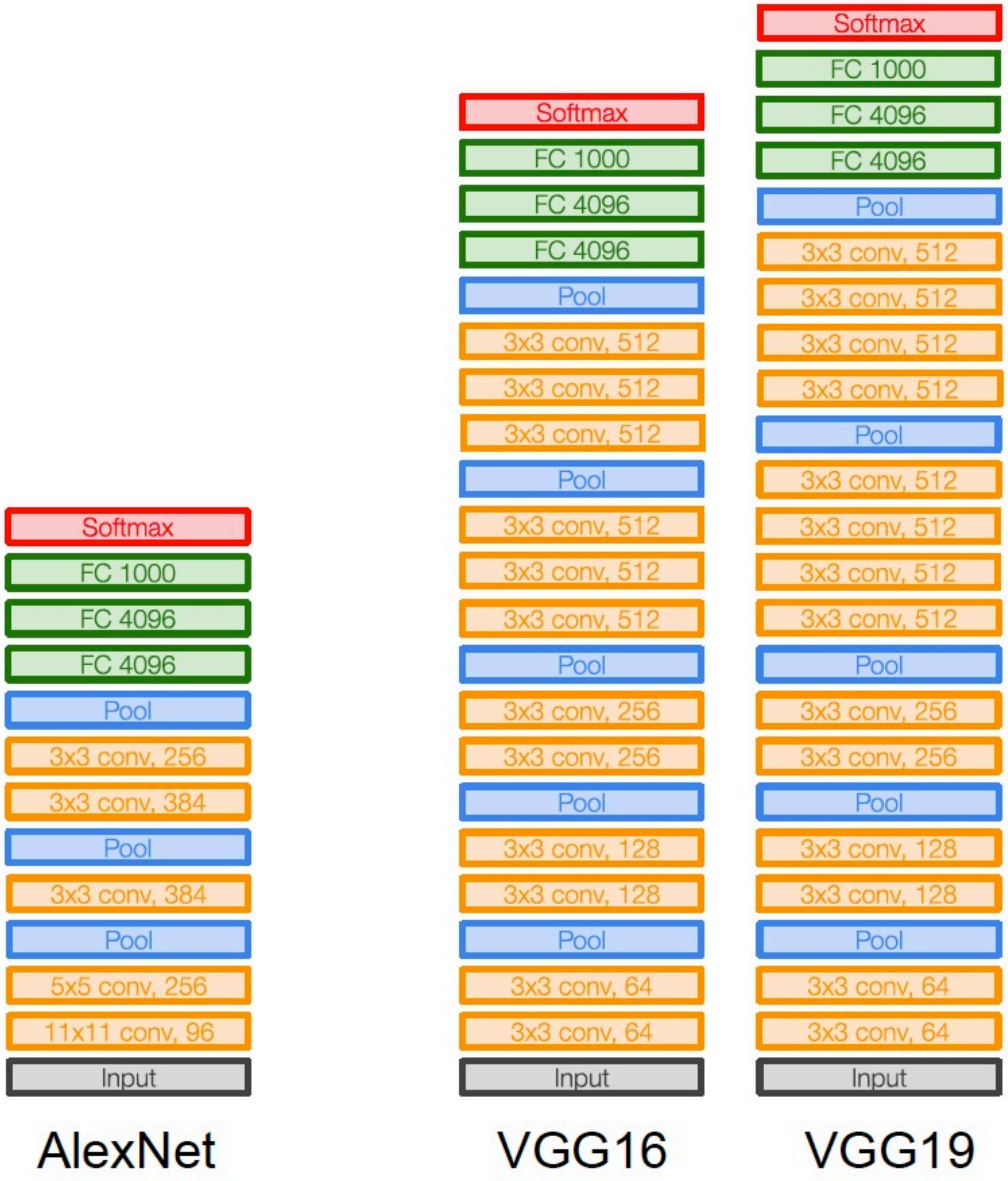
- Output for FC6: 4096. Since the layer is fully-connected, the number of parameter is: 6 x 6 x 256 x 4096 = 38 million

ImageNet Performance



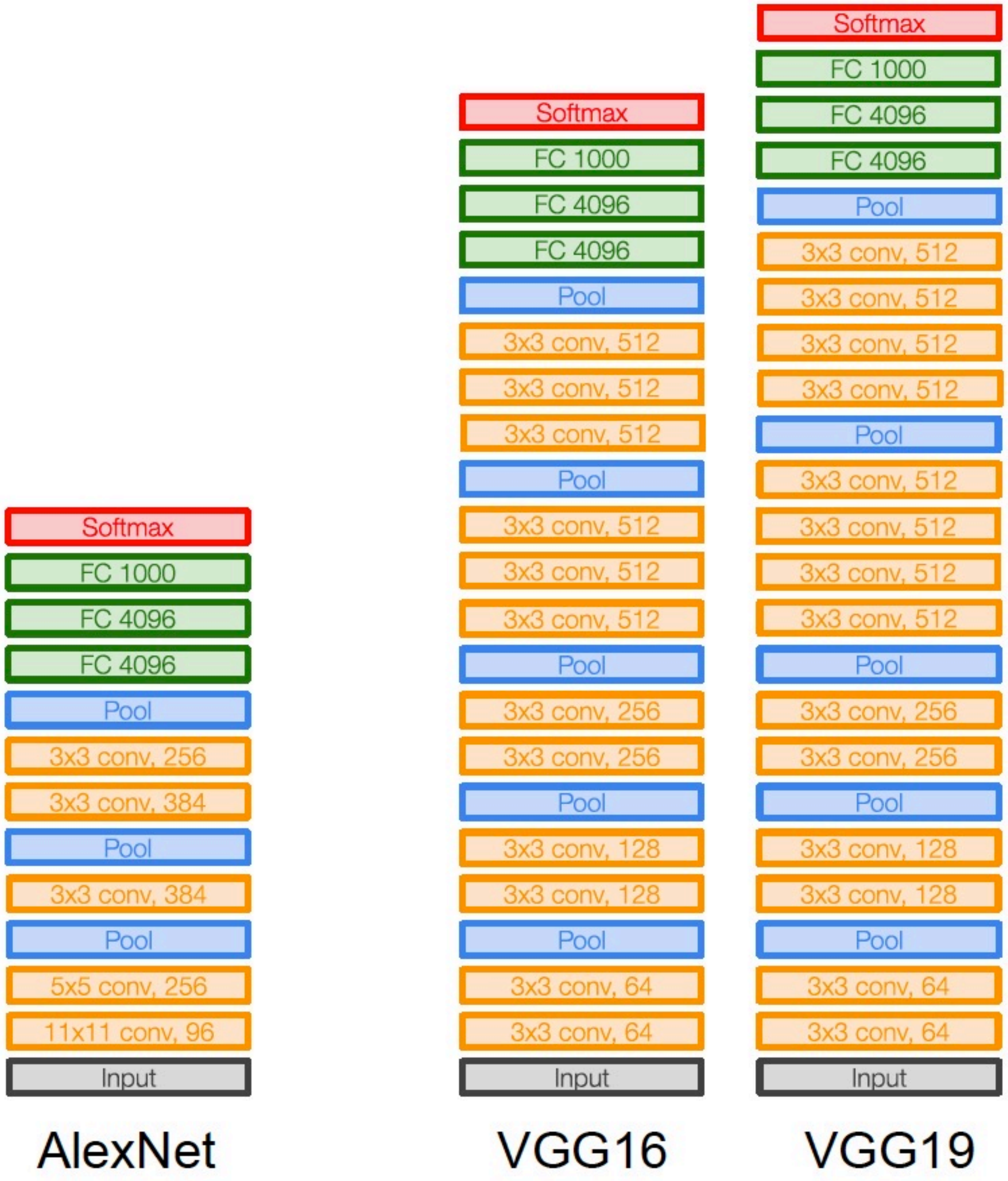
VGGNet

- AlexNet: Larger filters, less layers (8 layers).
- VGG: smaller filters, more layers (16 or 19 layers).

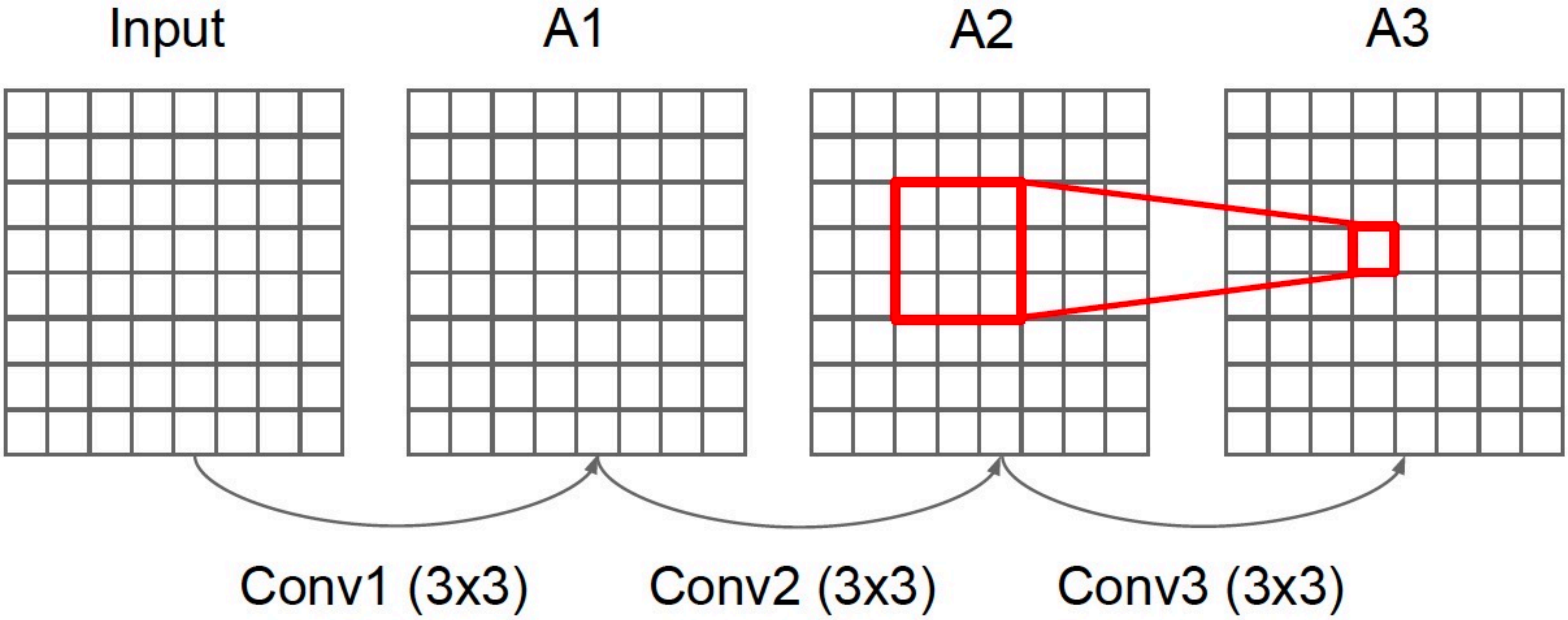


VGGNet

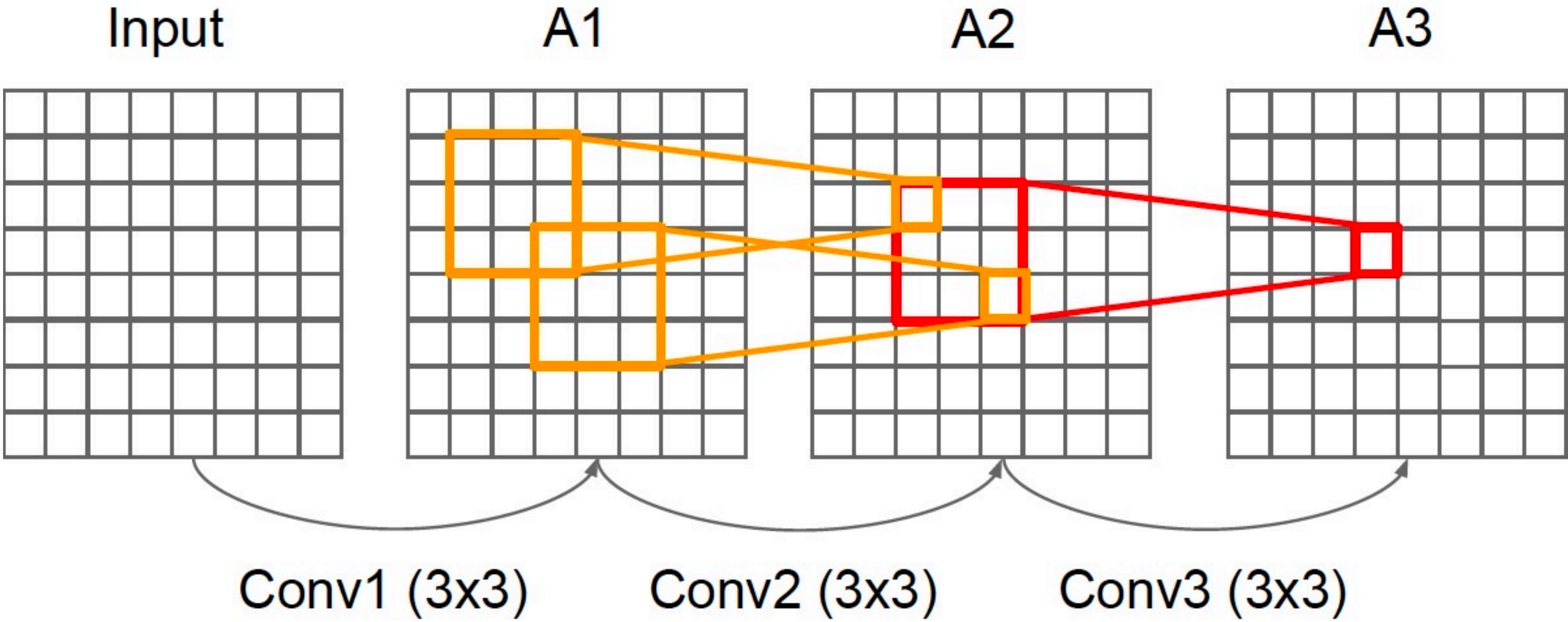
- A stack of three 3x3 conv filters has the same receptive field as a 7x7 conv filter
- Three 3x3 conv filters have more non-linear transformation



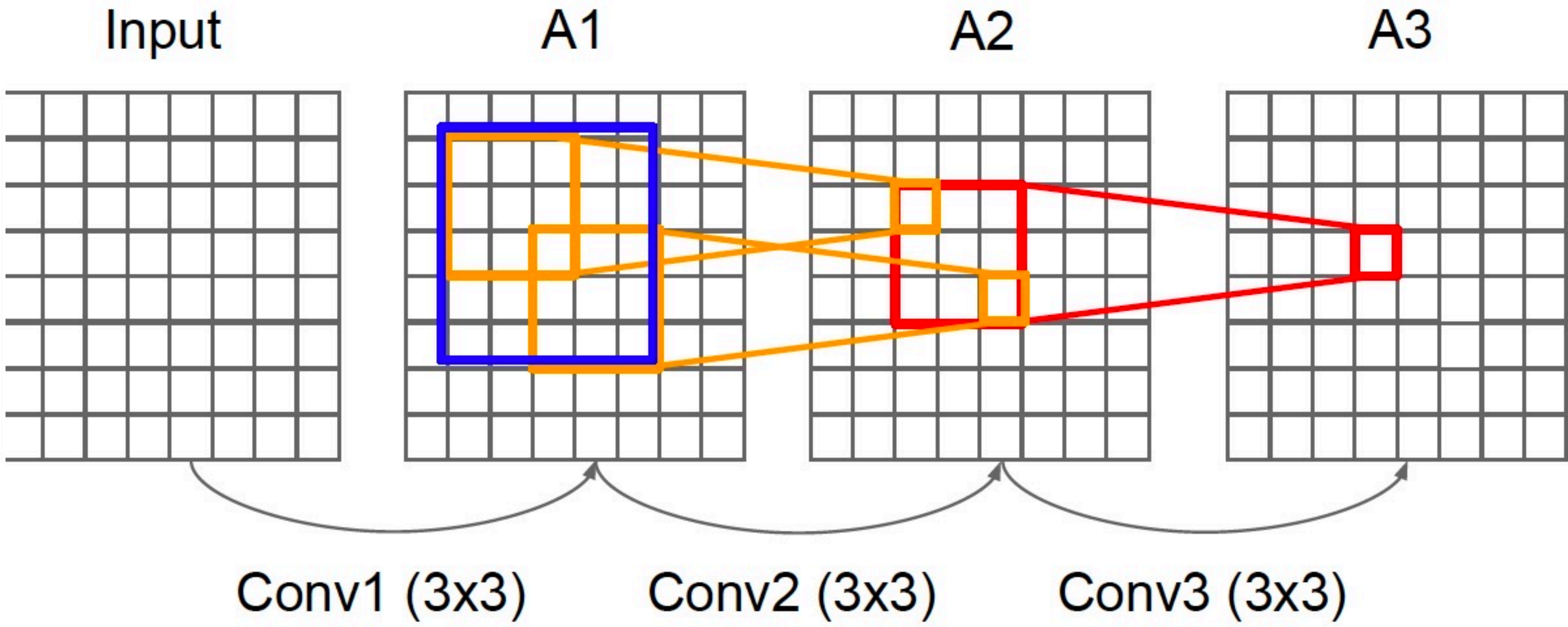
VGGNet-Receptive Fields



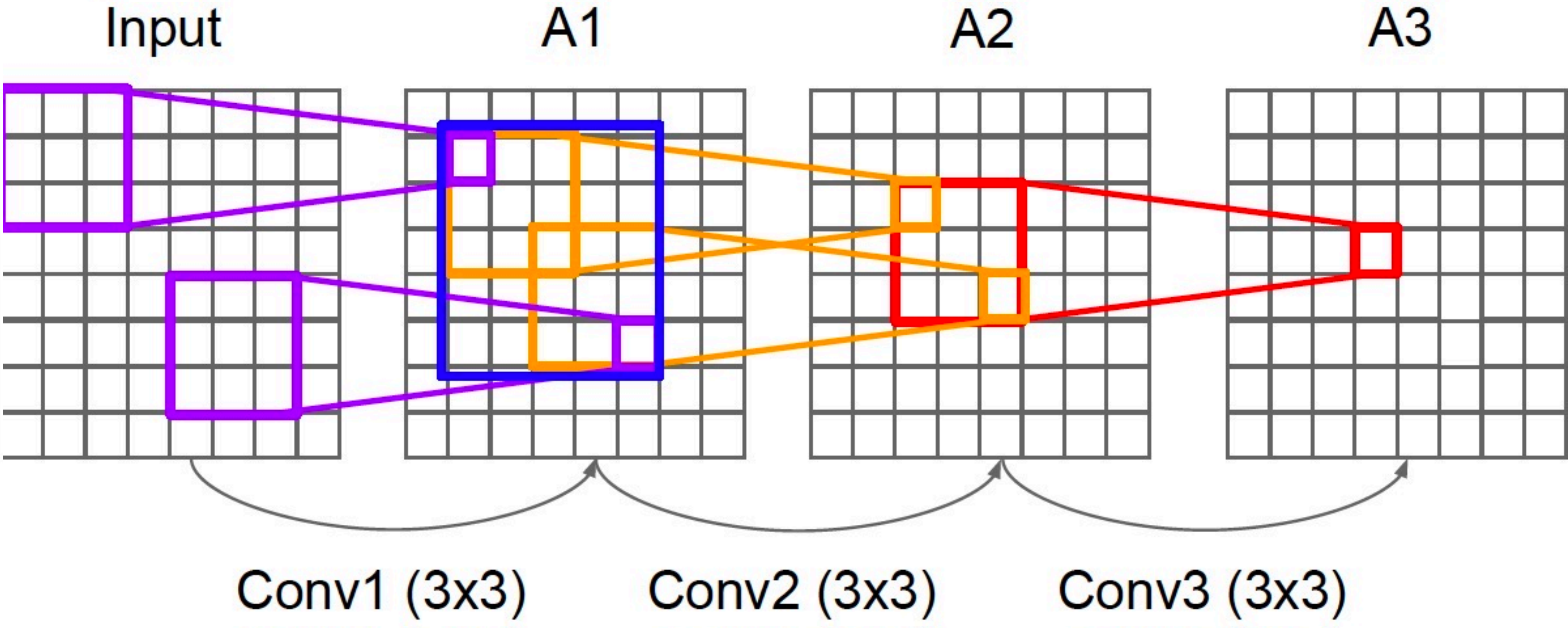
VGGNet-Receptive Fields



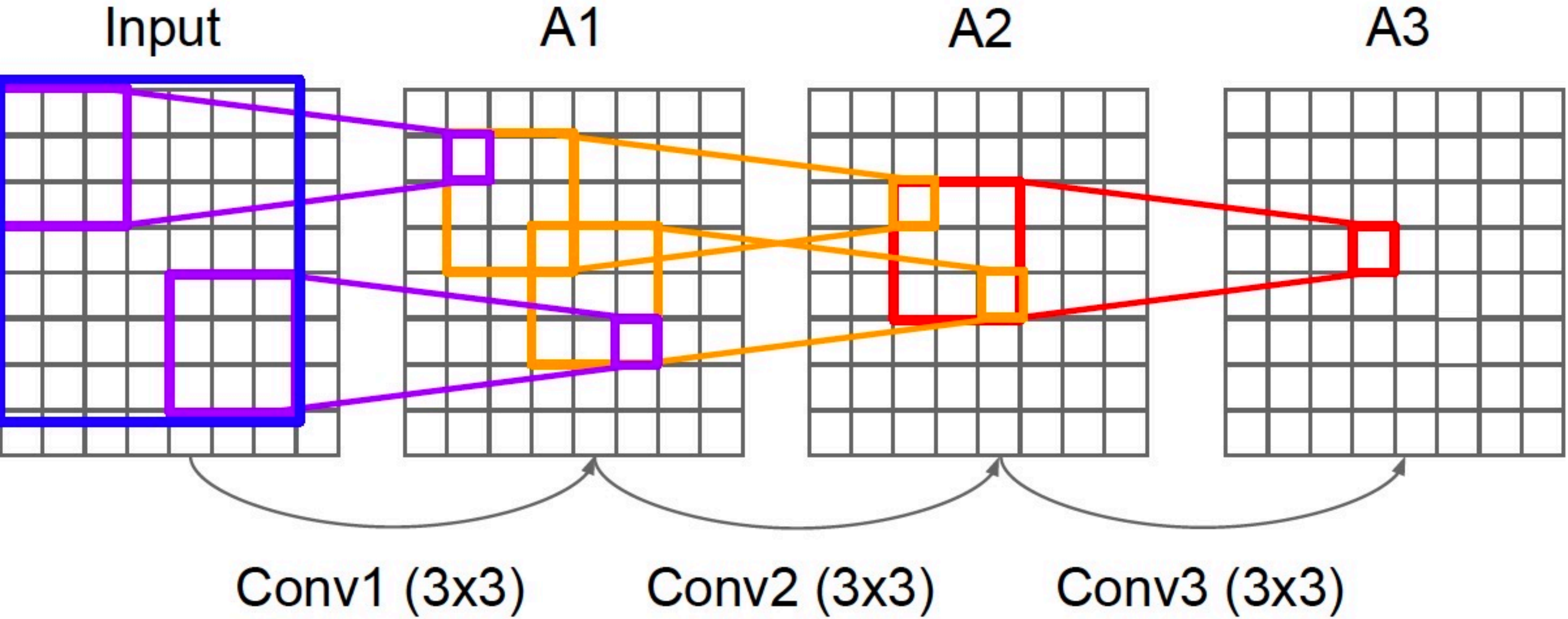
VGGNet-Receptive Fields



VGGNet-Receptive Fields

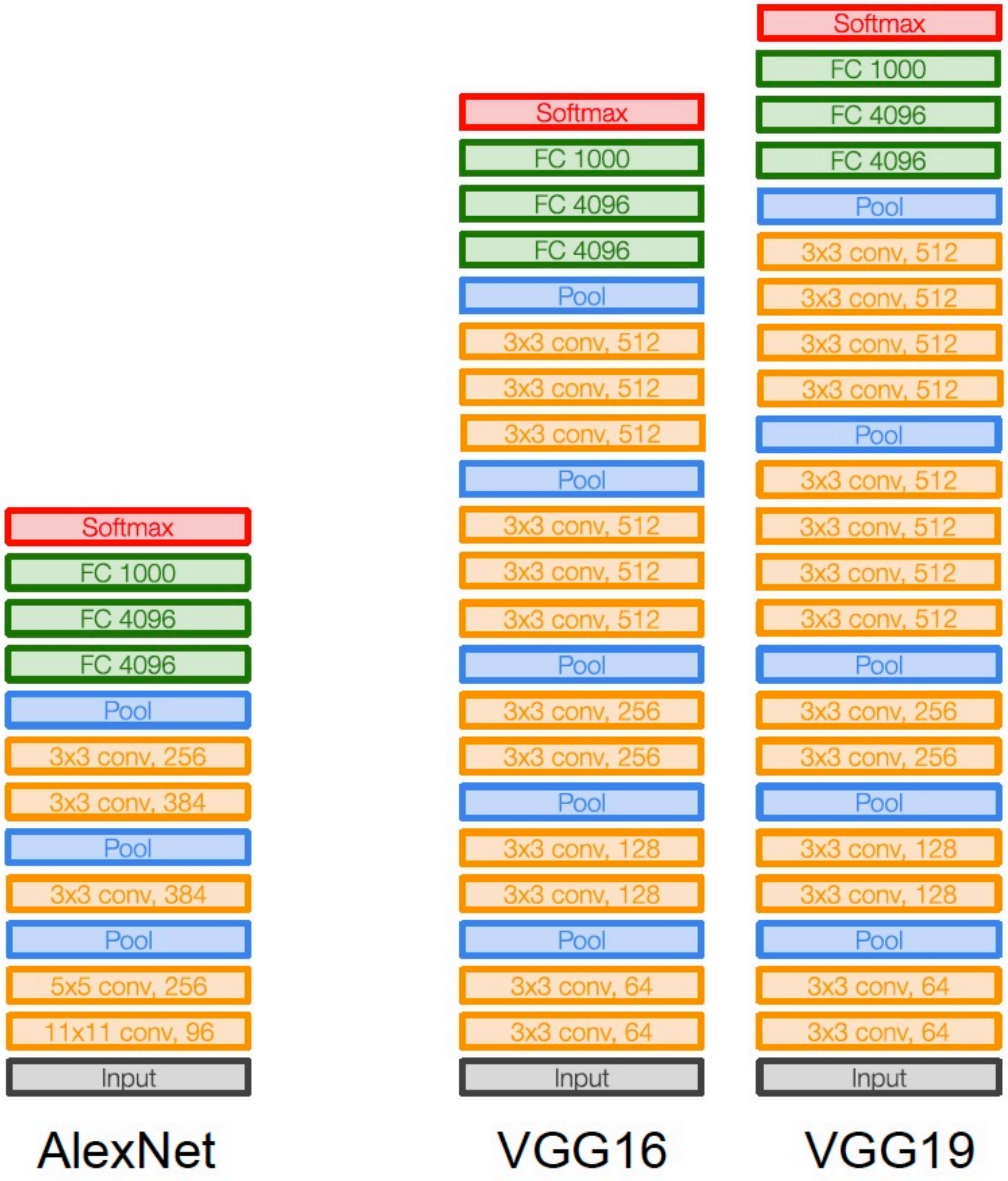


VGGNet-Receptive Fields

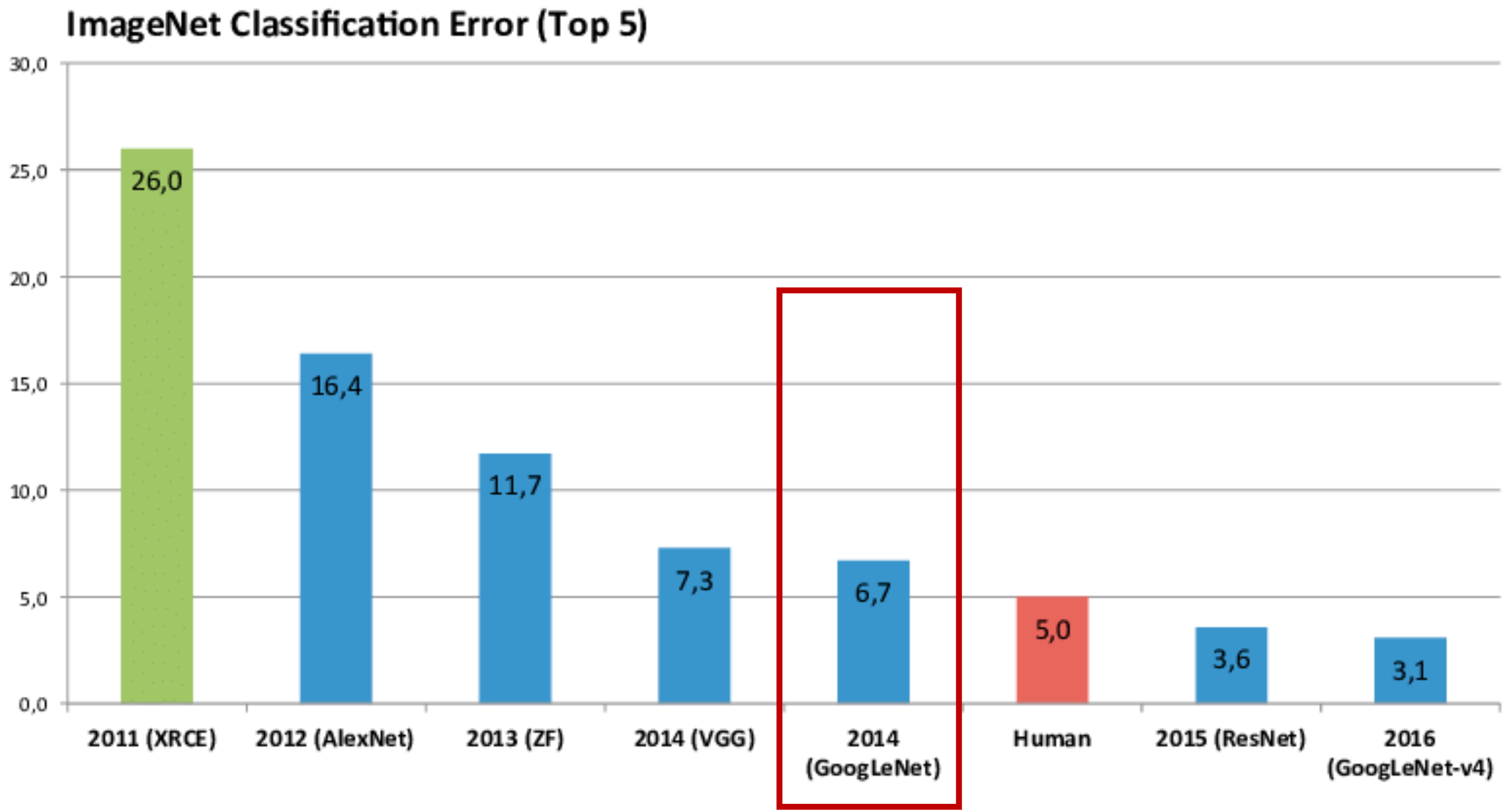


VGGNet

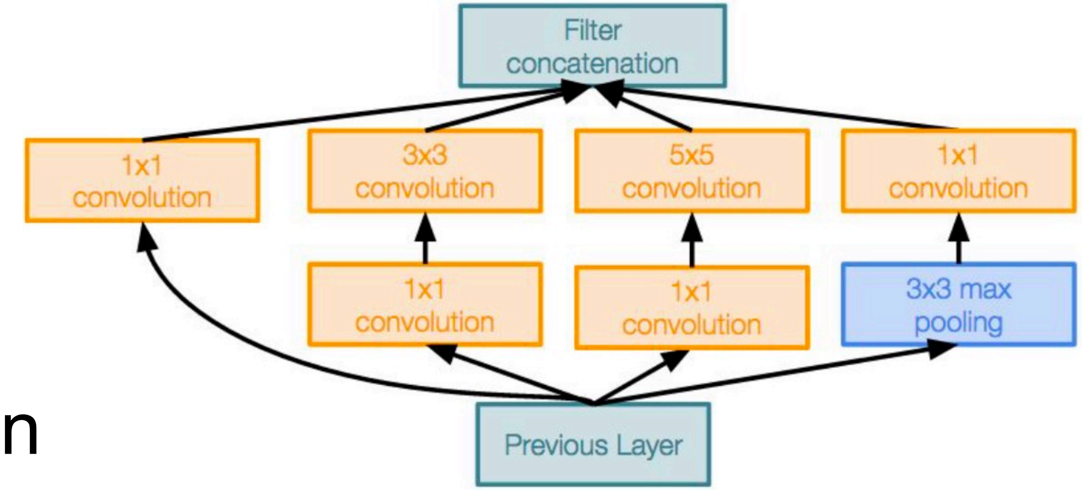
- A general direction: Going deeper with 3x3 convolution

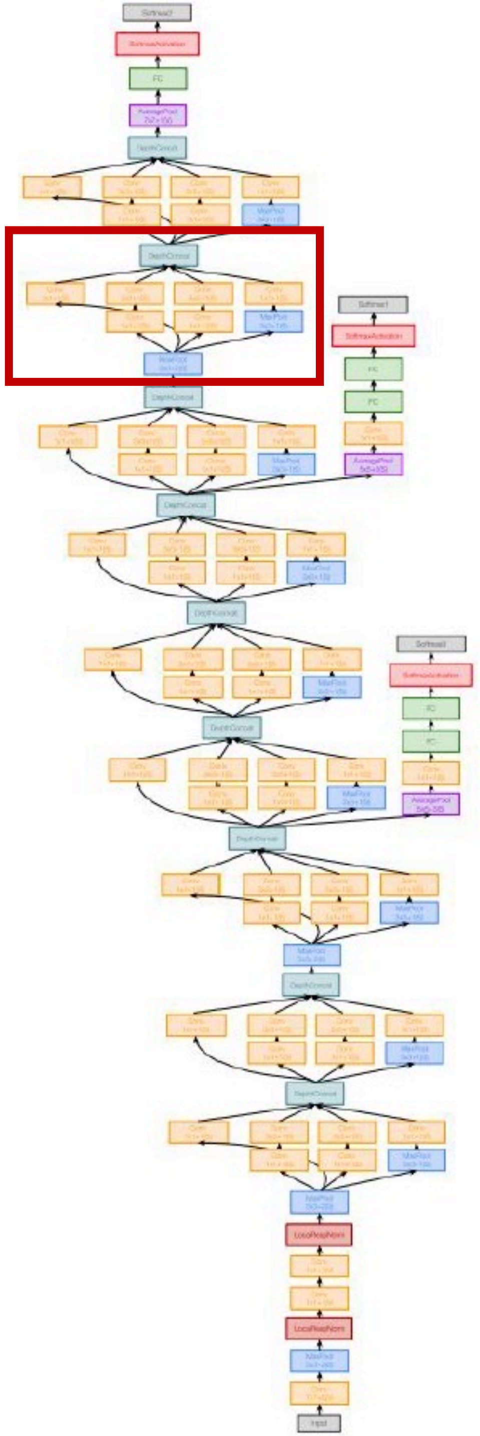


ImageNet Performance



GoogleNet

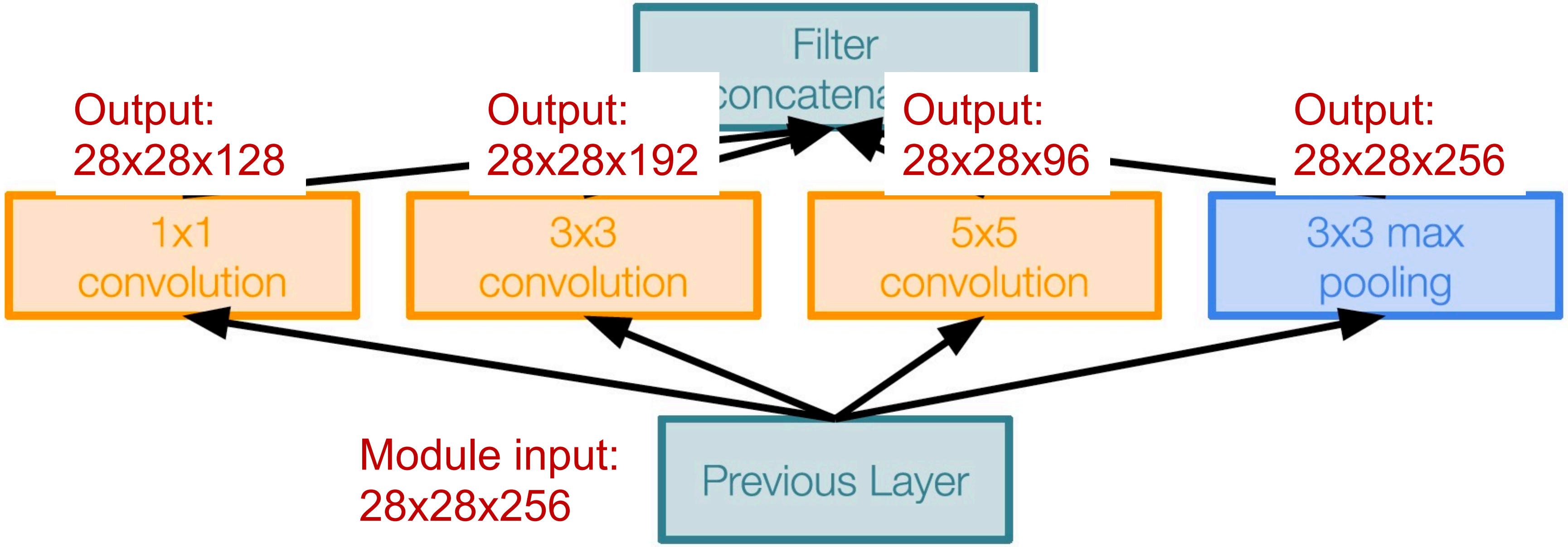




- Apply multiple filters in parallel

- Concat the results of multiple filters for the next layer

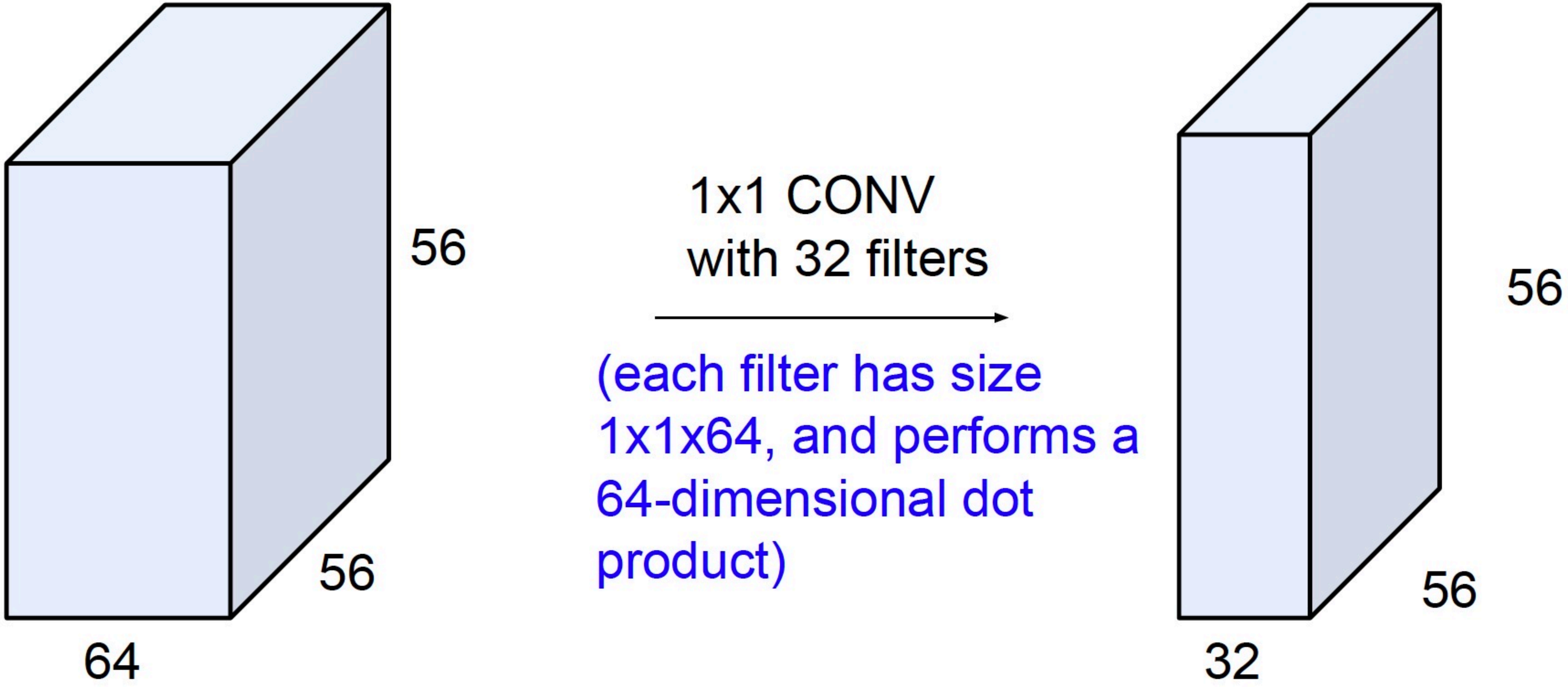
GoogleNet -- A naive inception module



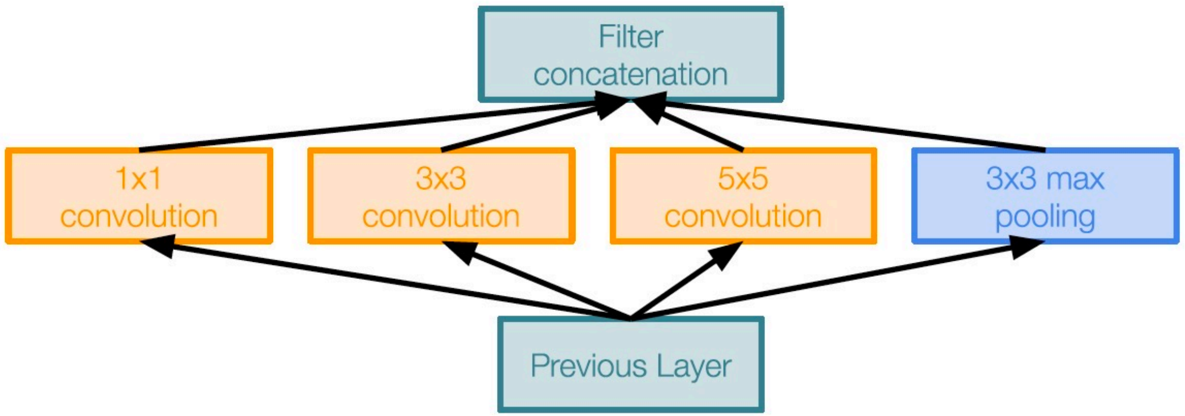
- Take 3x3 convolution as an example:
- Filter size: 3x3x192x256
- Conv Ops: 28x28x**3x3x192x256**

Can we reduce the computation?

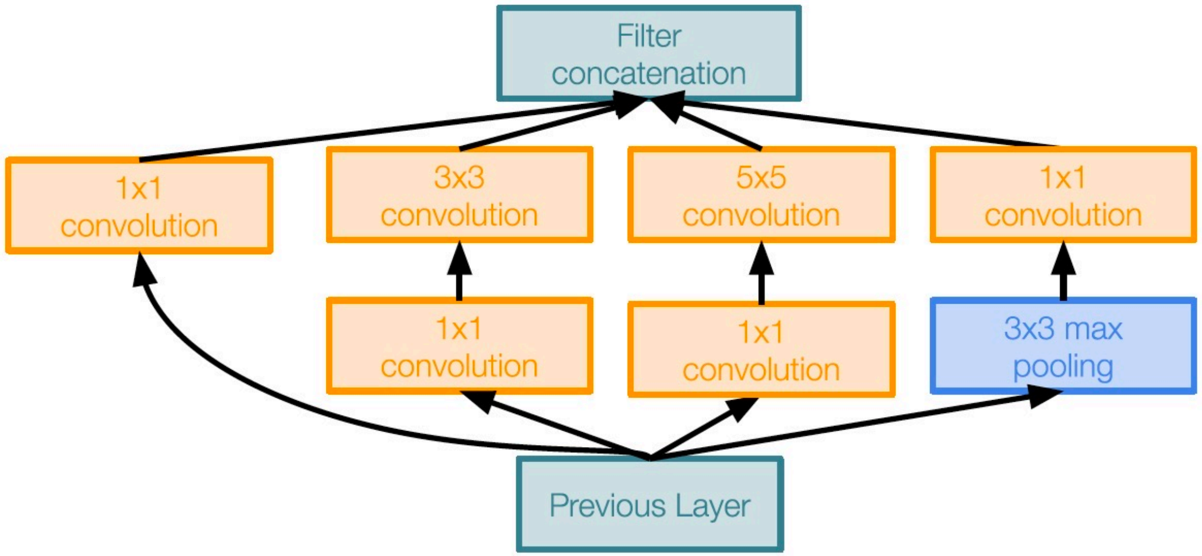
1 x 1 convolutions: dimension reduction



GoogleNet

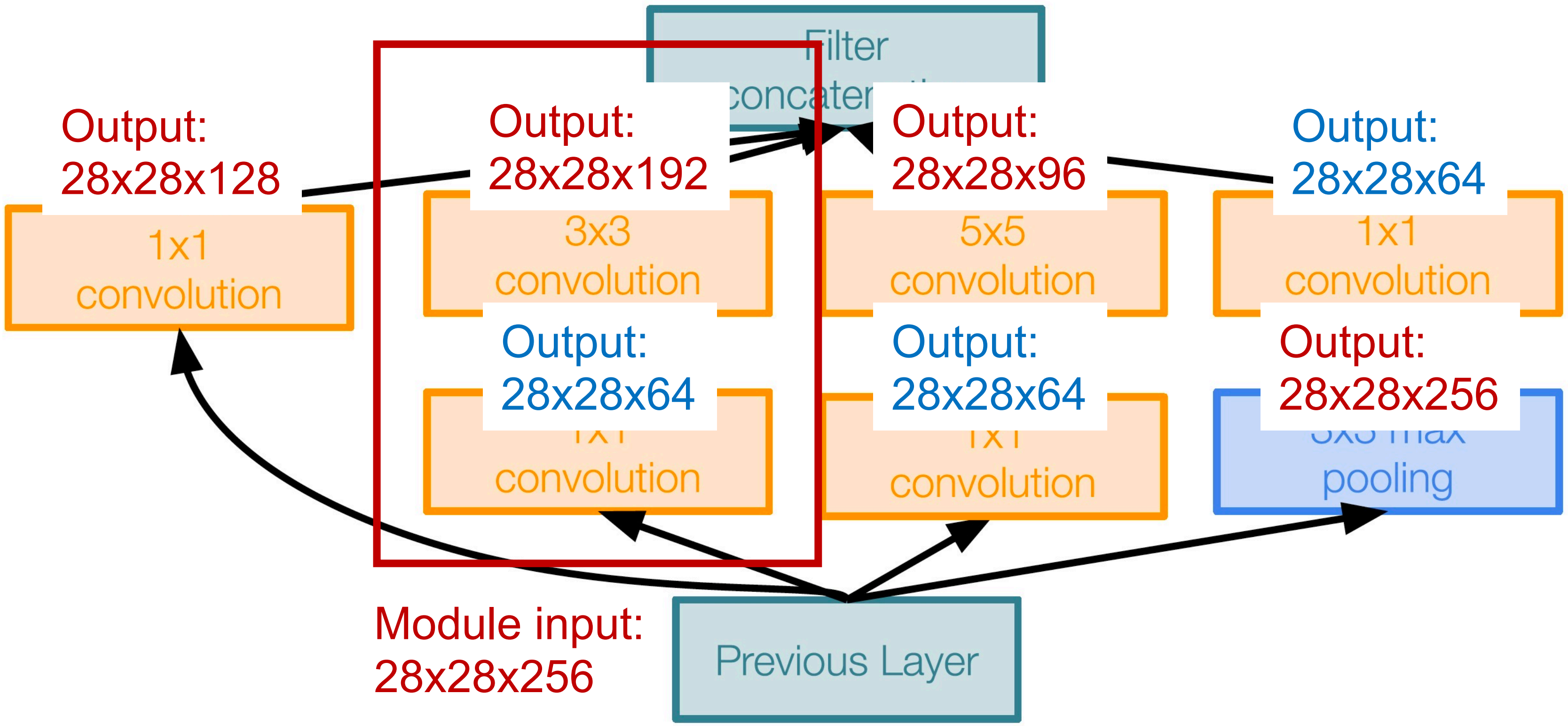


Naive Inception module



Inception module with dimension reduction

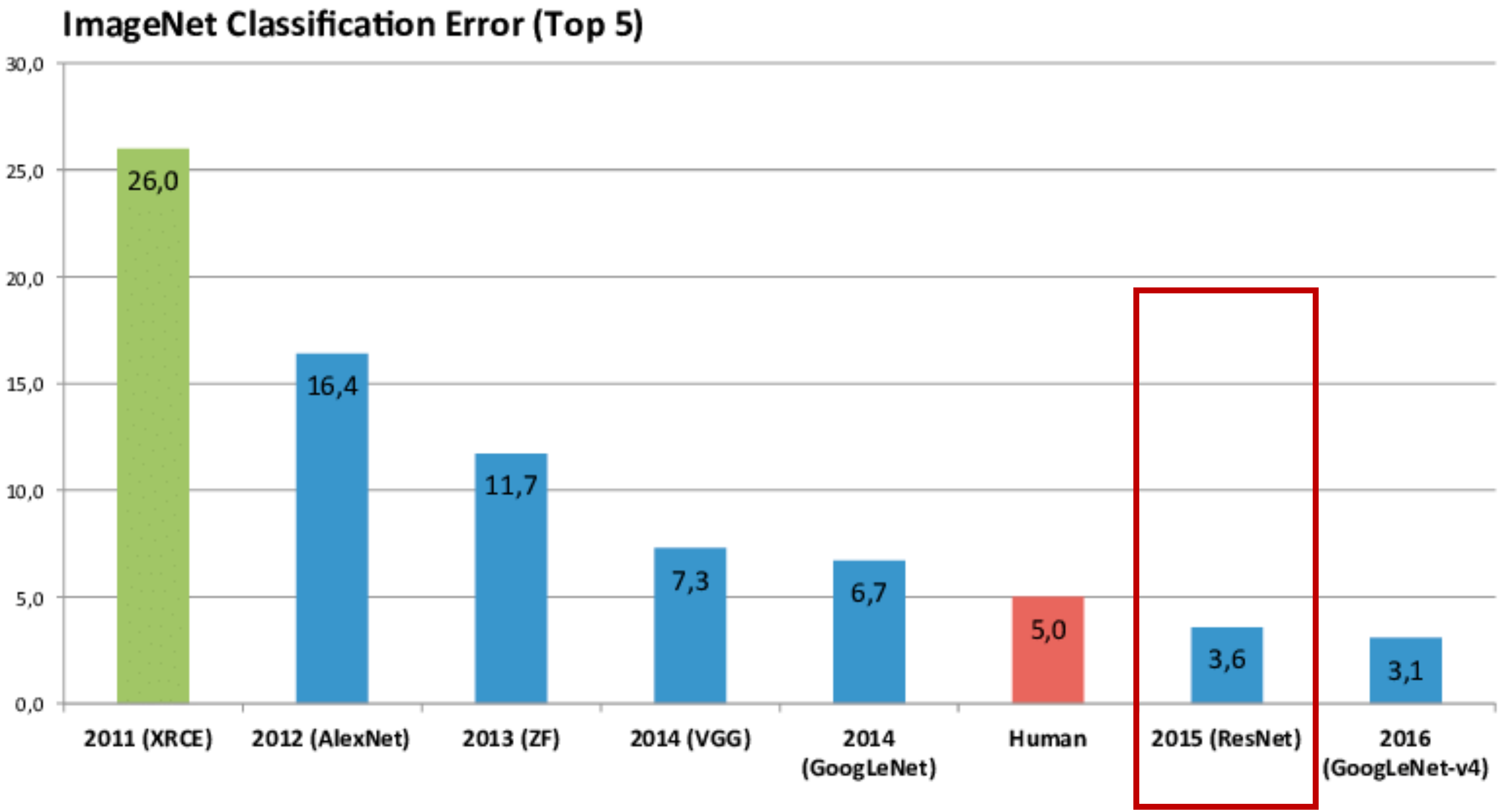
GoogleNet



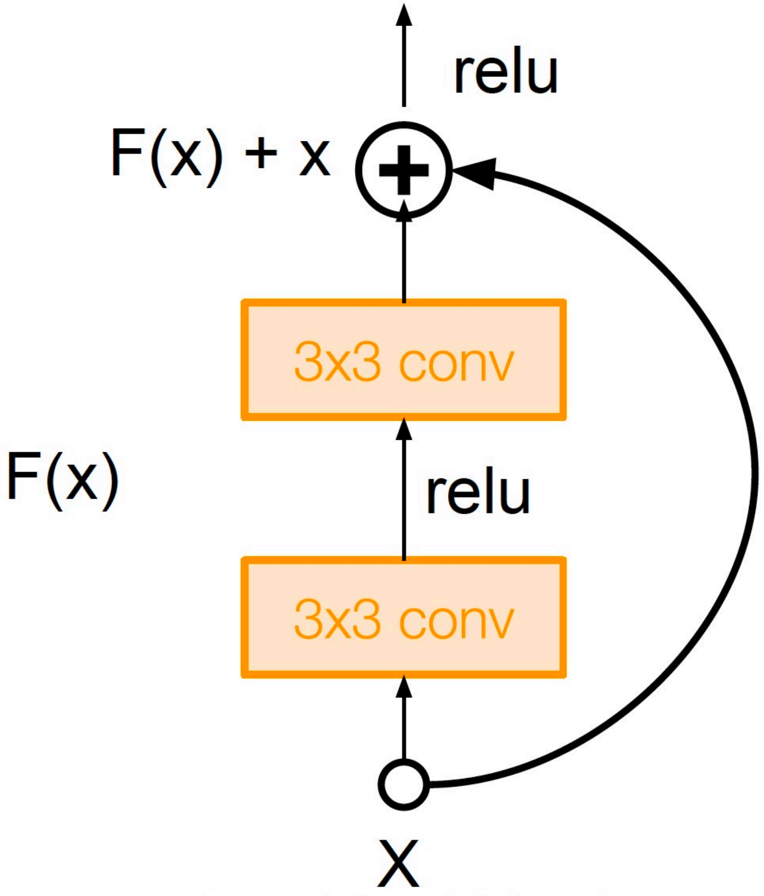
- Take 3x3 + 1x1 convolutions as an example:
- Filter size:
  3x3x192x64
  1x1x64x256
- Conv Ops:
  28x28x**3x3x192x64**
  28x28x**1x1x64x256**

Previous: 28x28x**3x3x192x256**

ImageNet Performance

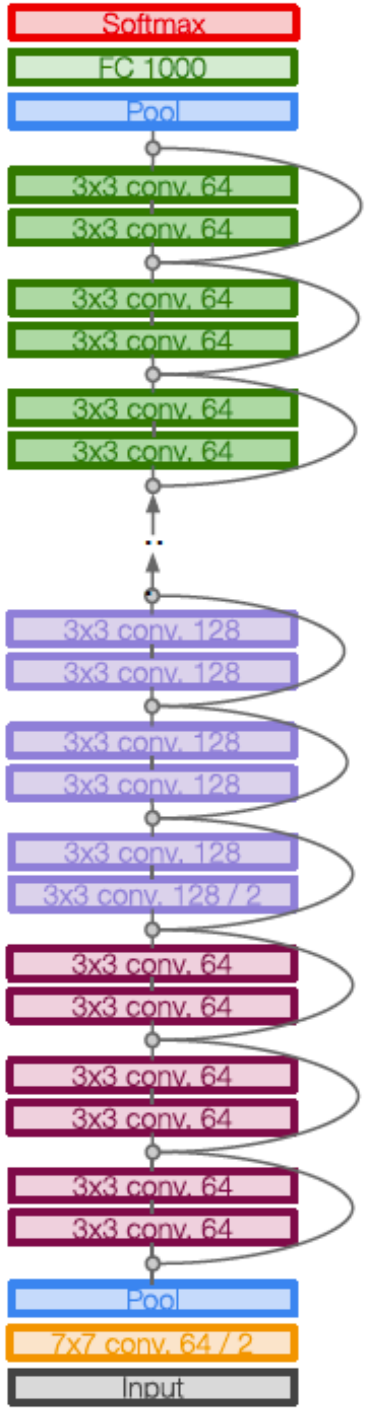


ResNet



Residual block

$$y = F(x) + x$$



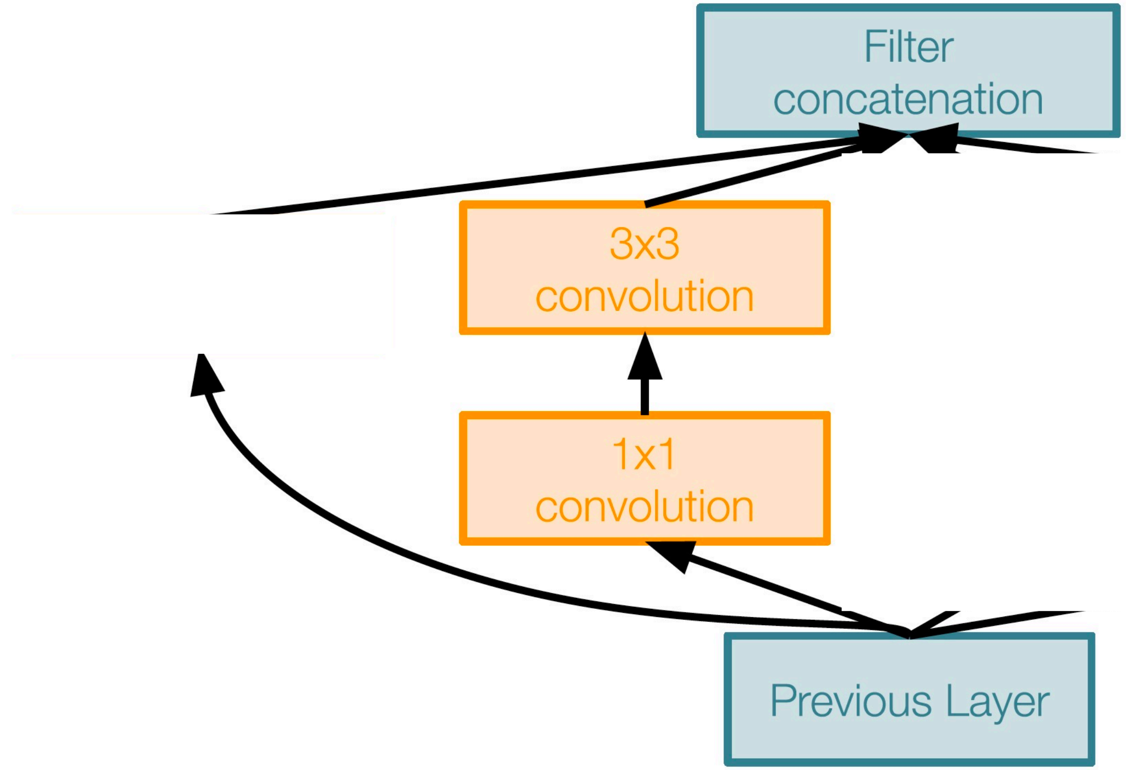
How is ResNet developed?

- Simplifying GoogleNet Inception module!

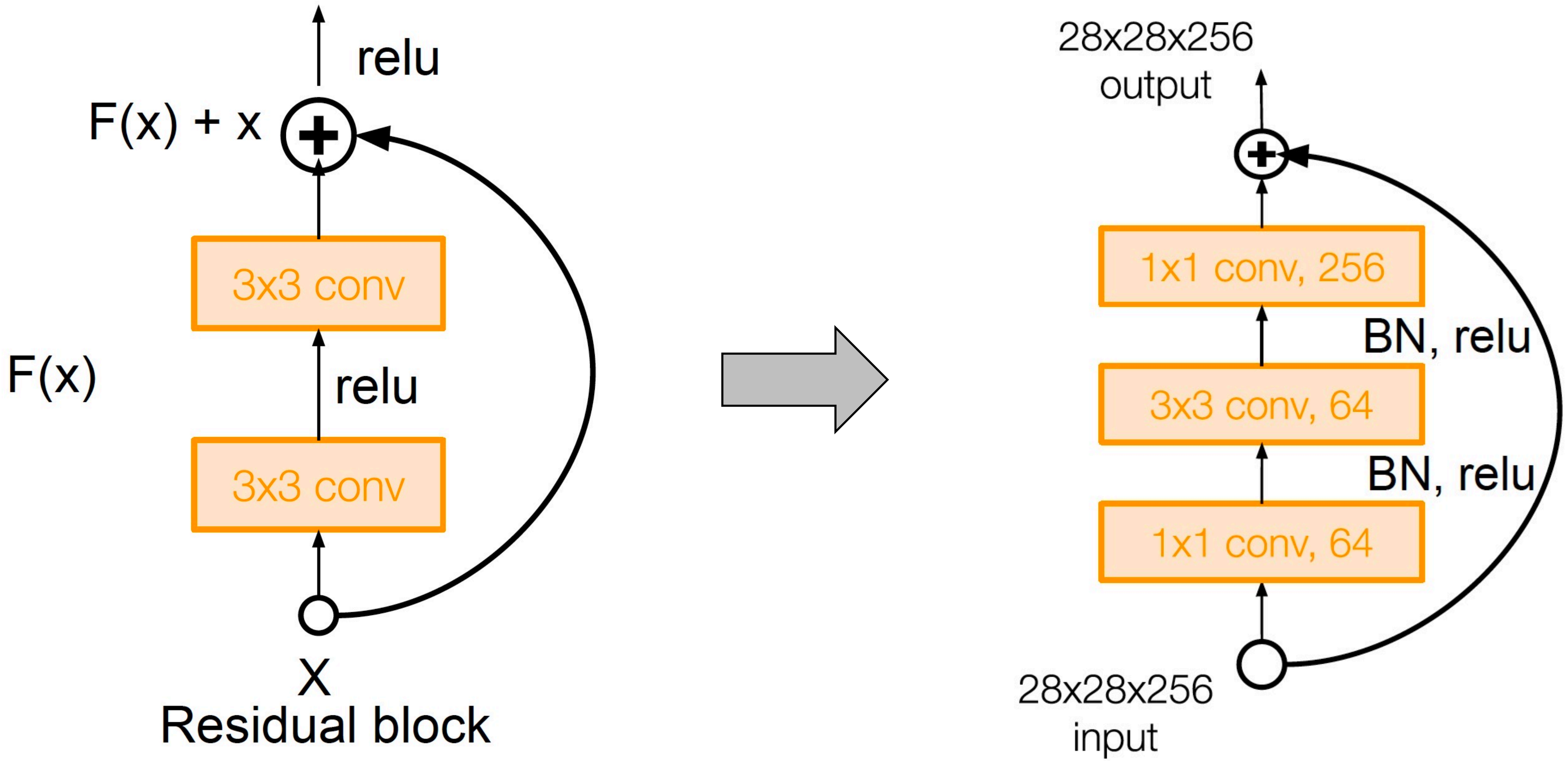
GoogleNet ResNet VGG16

How is ResNet developed?

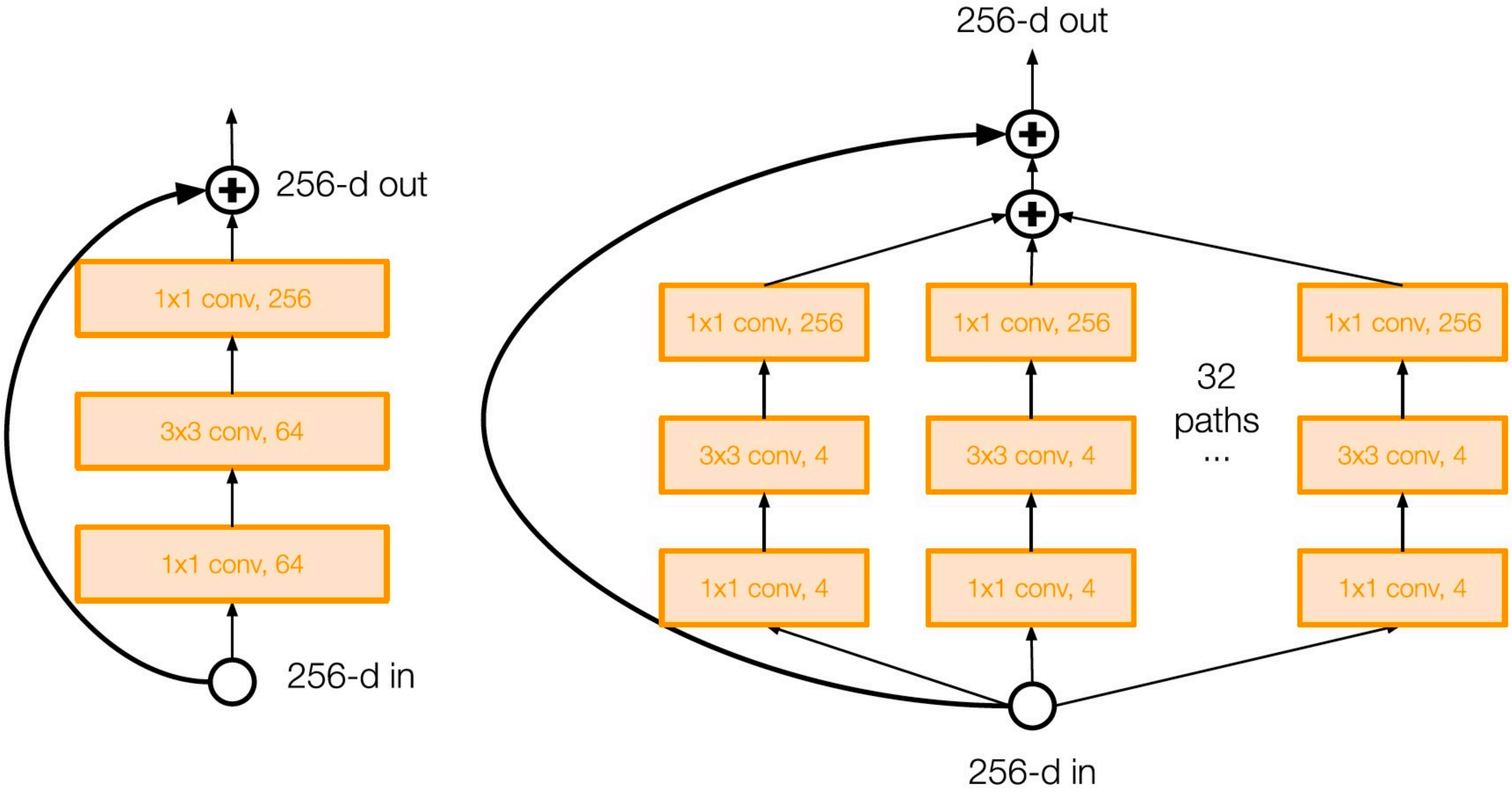
- Simplifying Inception module!



BottleNeck with 1x1 convolution

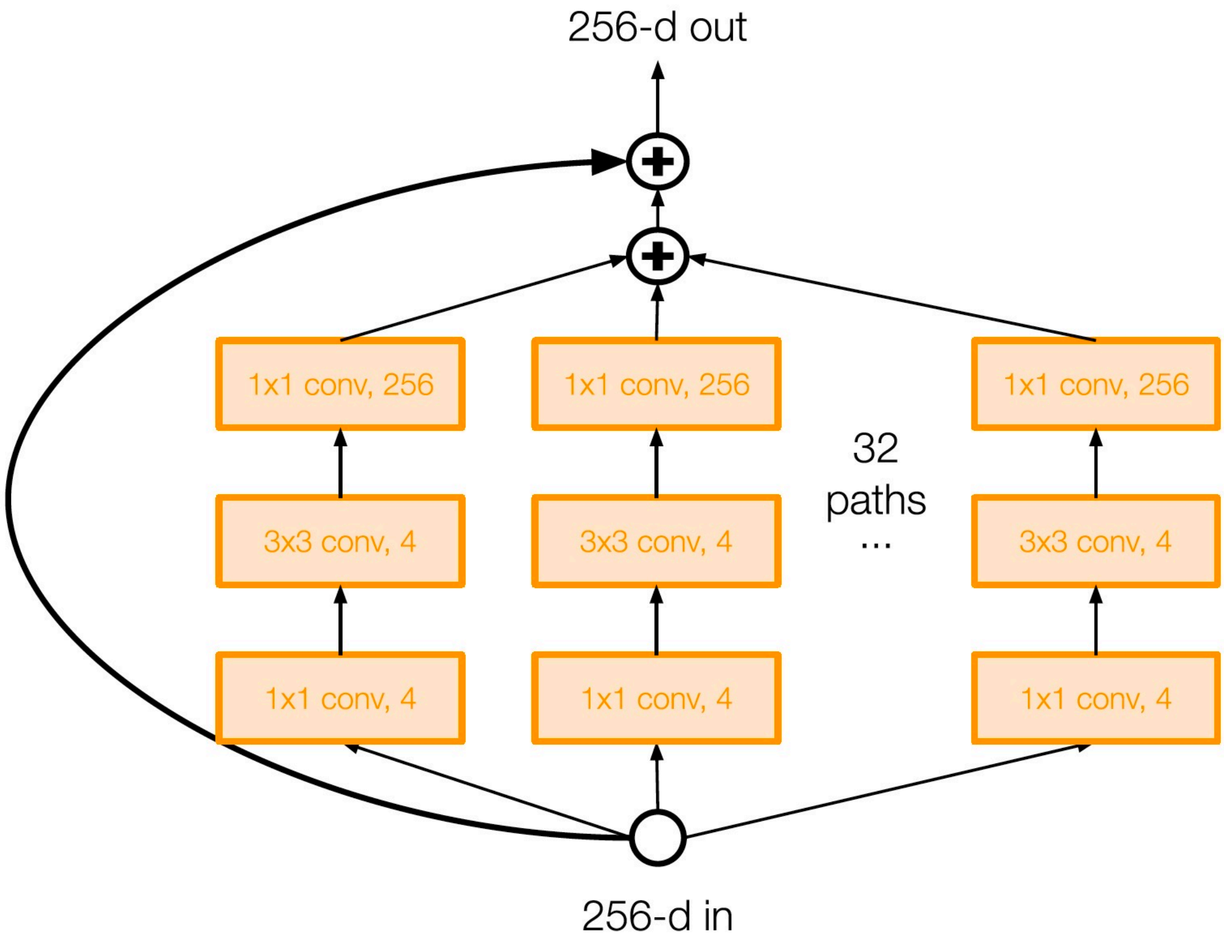


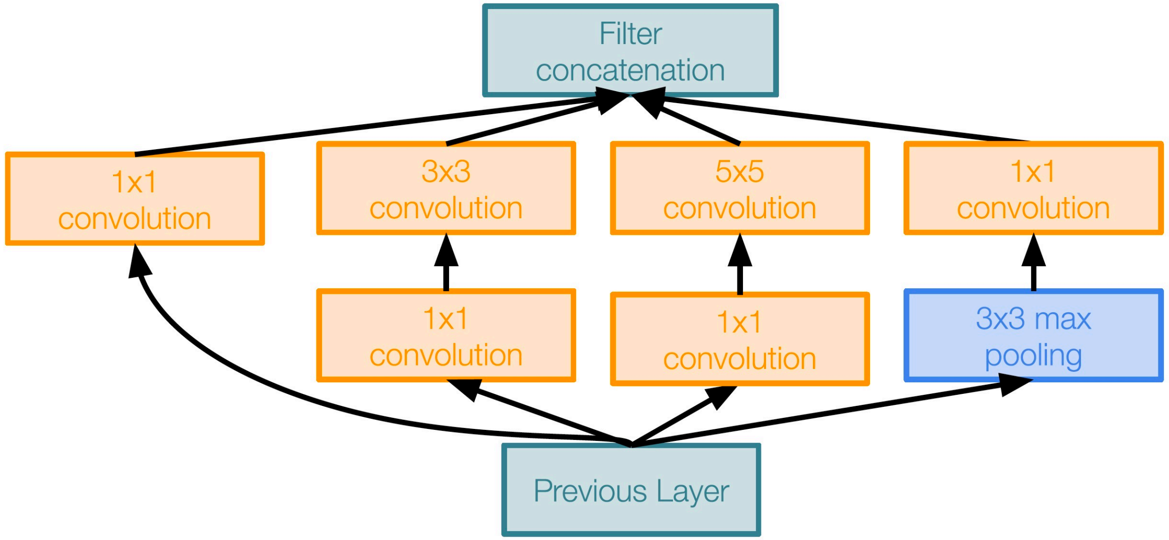
ResNeXt



Looks familiar?

ResNeXt

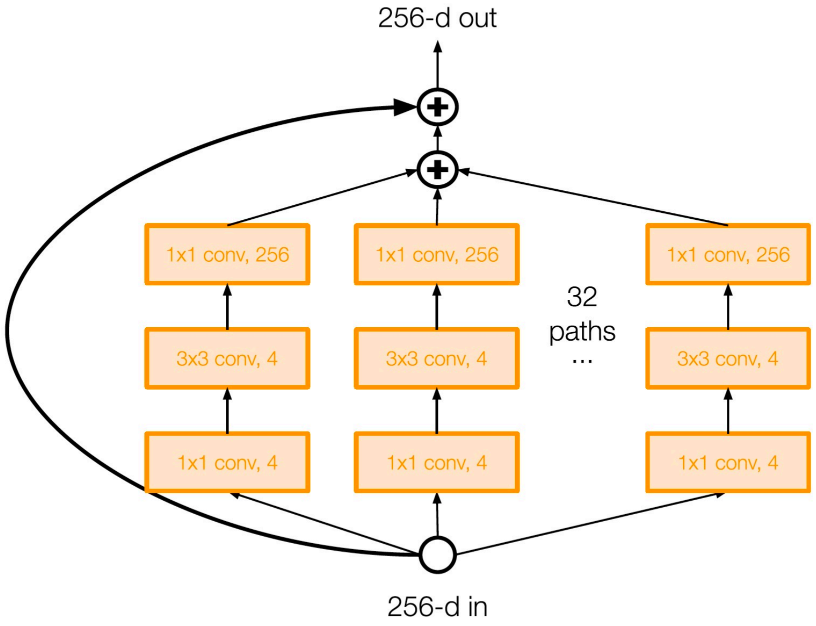

256-d out
1x1 conv, 256
1x1 conv, 256
1x1 conv, 256
32
paths
...
3x3 conv, 4
3x3 conv, 4
3x3 conv, 4
1x1 conv, 4
1x1 conv, 4
1x1 conv, 4
256-d in

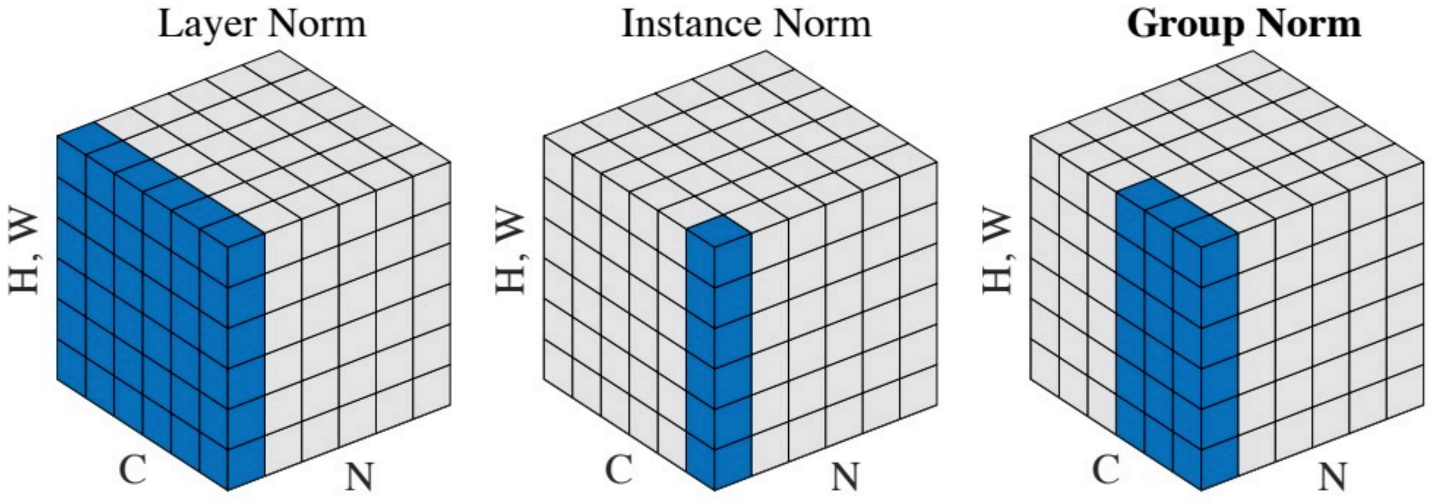

Filter
concatenation
1x1
convolution
3x3
convolution
5x5
convolution
1x1
convolution
1x1
convolution
1x1
convolution
3x3 max
pooling
Previous Layer

Compare ResNet and ResNeXt

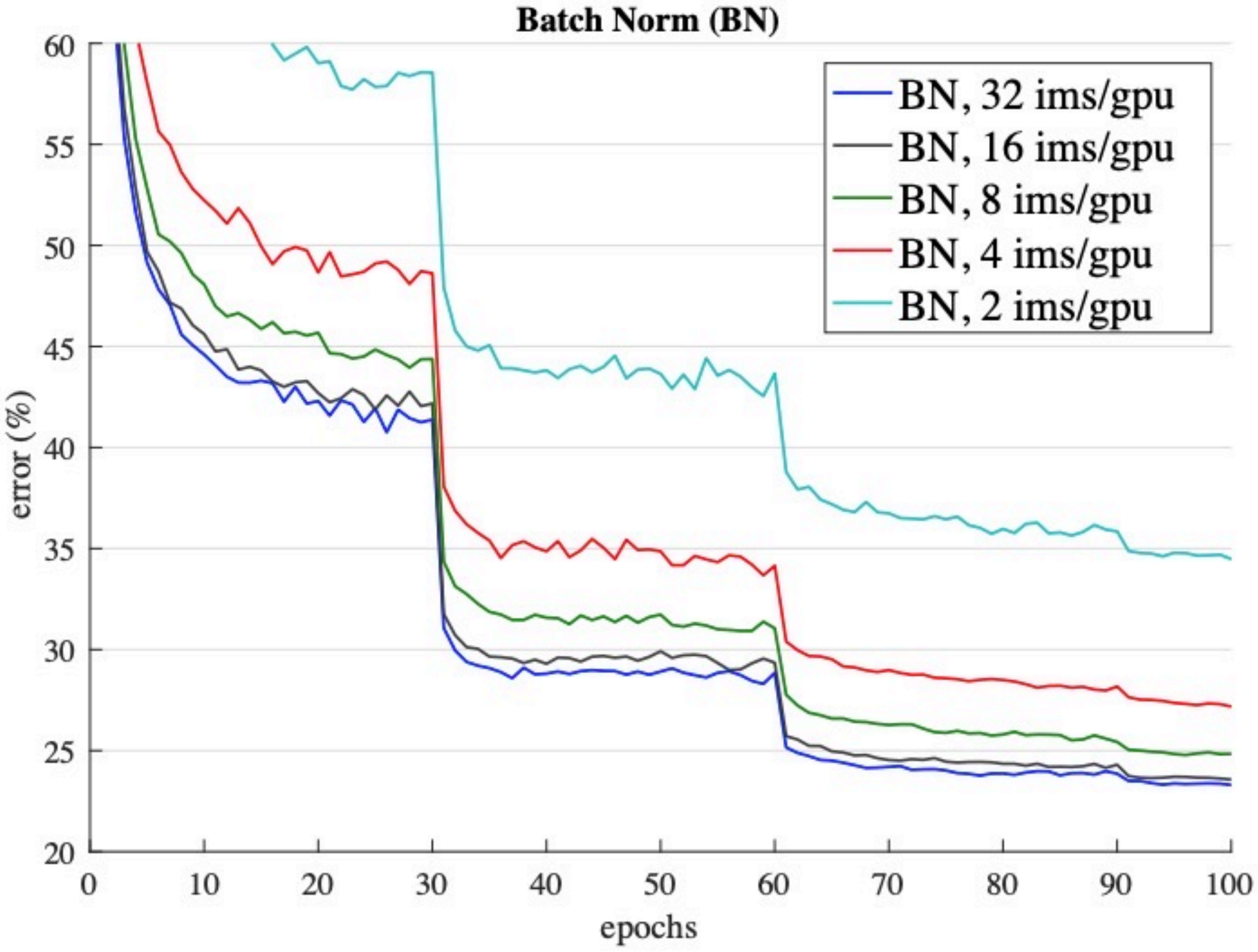
- Performance: ResNeXt will give 1 to 2% improvement in general in many recognition tasks
- Efficiency: ResNeXt is much slower, 1.5 times to 2 times slower (Group Conv was not very well optimized)

About Groups

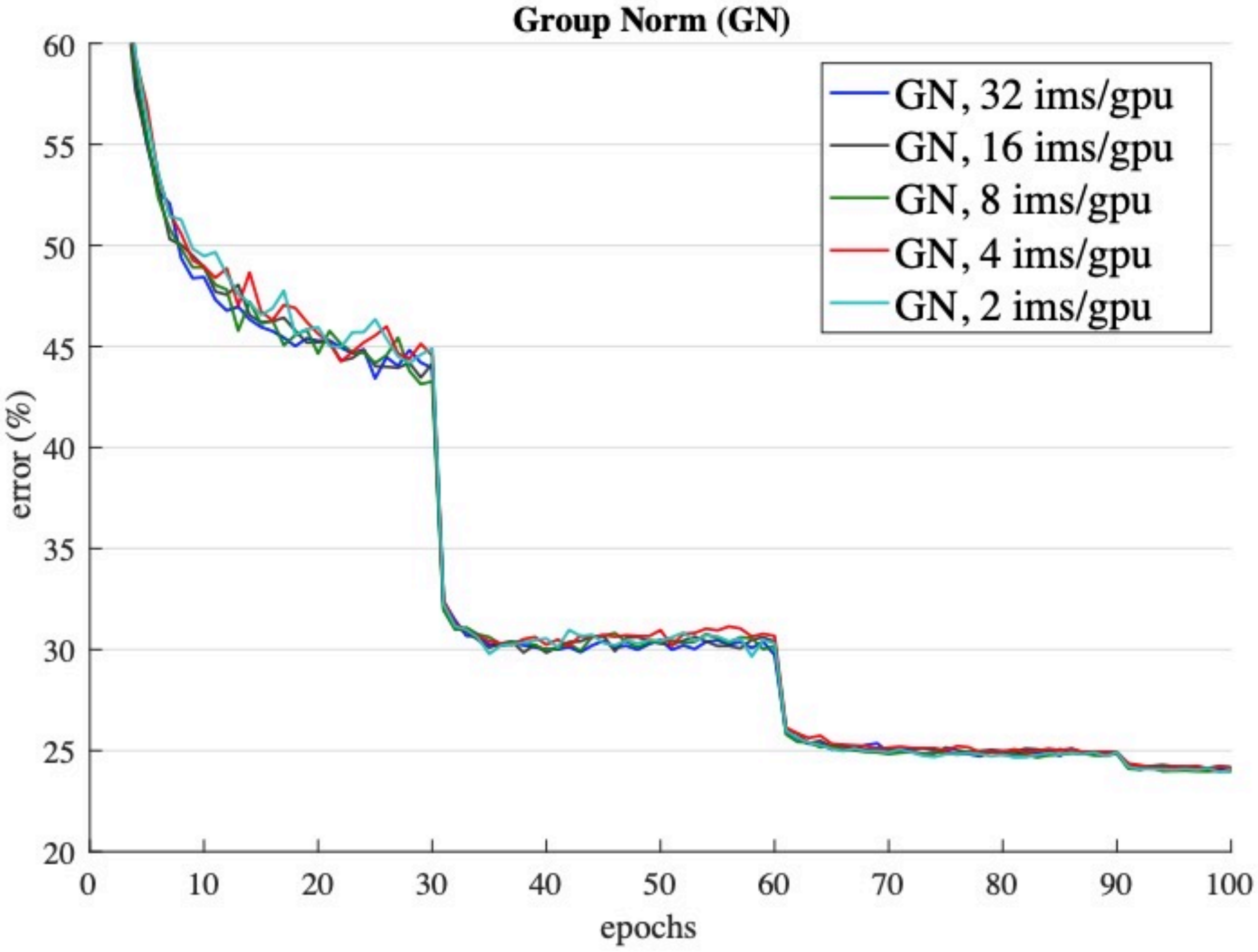




Group Norm



Batch Norm (BN)
BN, 32 ims/gpu
BN, 16 ims/gpu
BN, 8 ims/gpu
BN, 4 ims/gpu
BN, 2 ims/gpu
error (%)
epochs



Group Norm (GN)
GN, 32 ims/gpu
GN, 16 ims/gpu
GN, 8 ims/gpu
GN, 4 ims/gpu
GN, 2 ims/gpu
error (%)
epochs

This Class

- Fine-tuning CNN
- Different network architectures
- Design trend of the architectures

Next Class

Semantic Segmentation