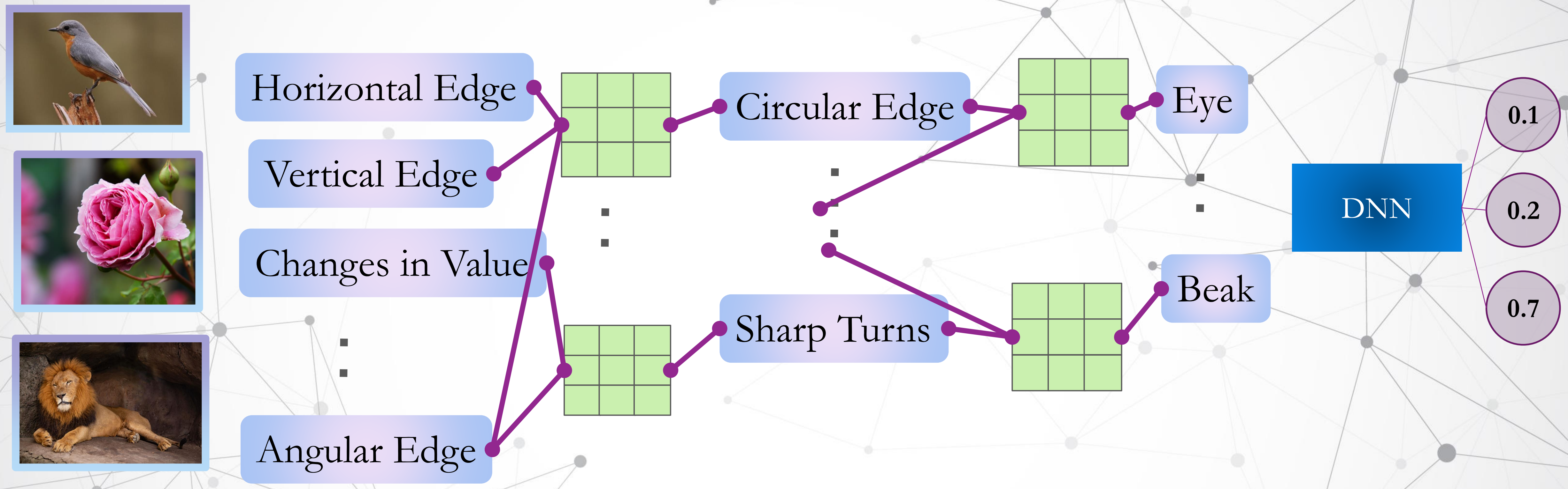
MLDL-I

Machine Learning and Deep Learning - I

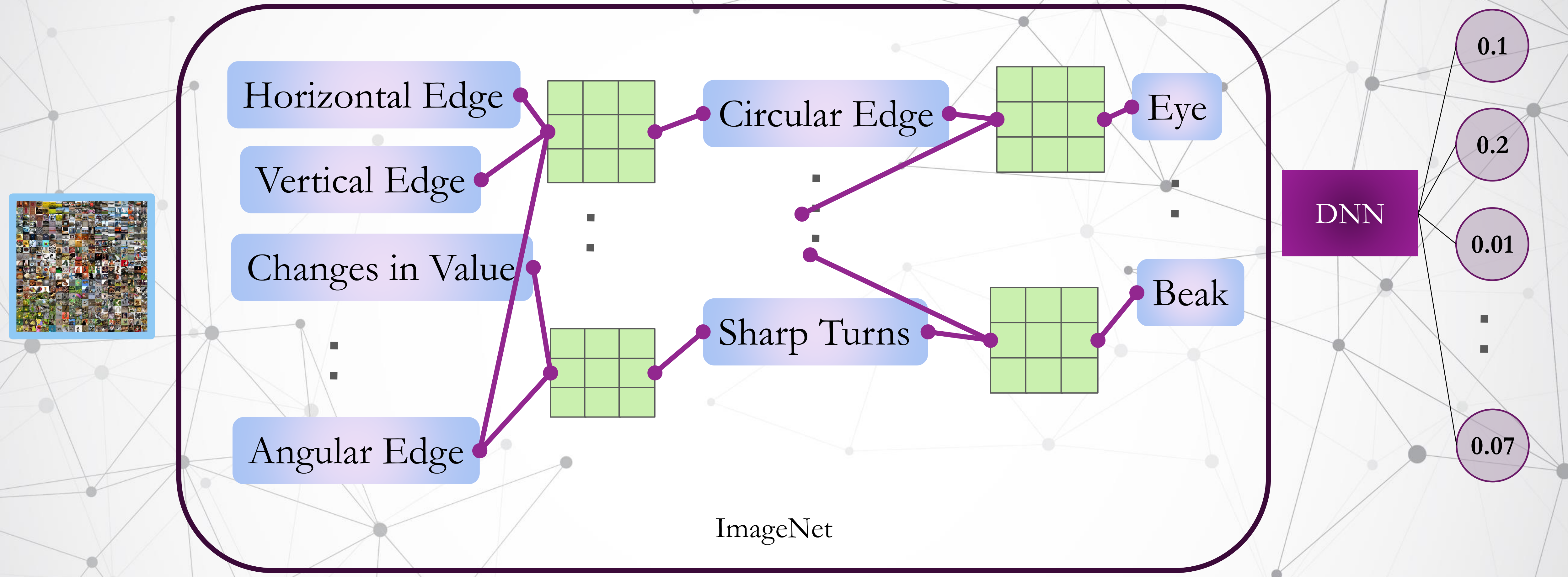



Horizontal Edge
Vertical Edge
Changes in Value
Angular Edge
Circular Edge
Sharp Turns
Eye
Beak
DNN
0.1
0.2
0.7

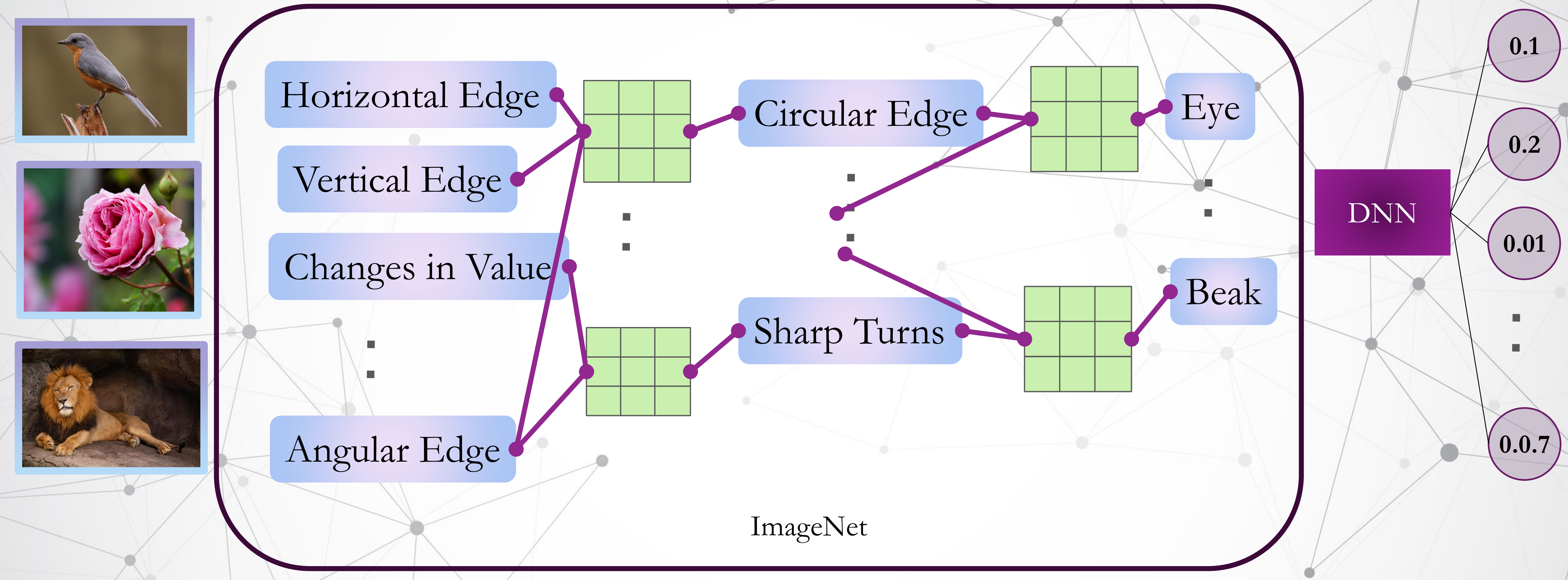
ImageNet

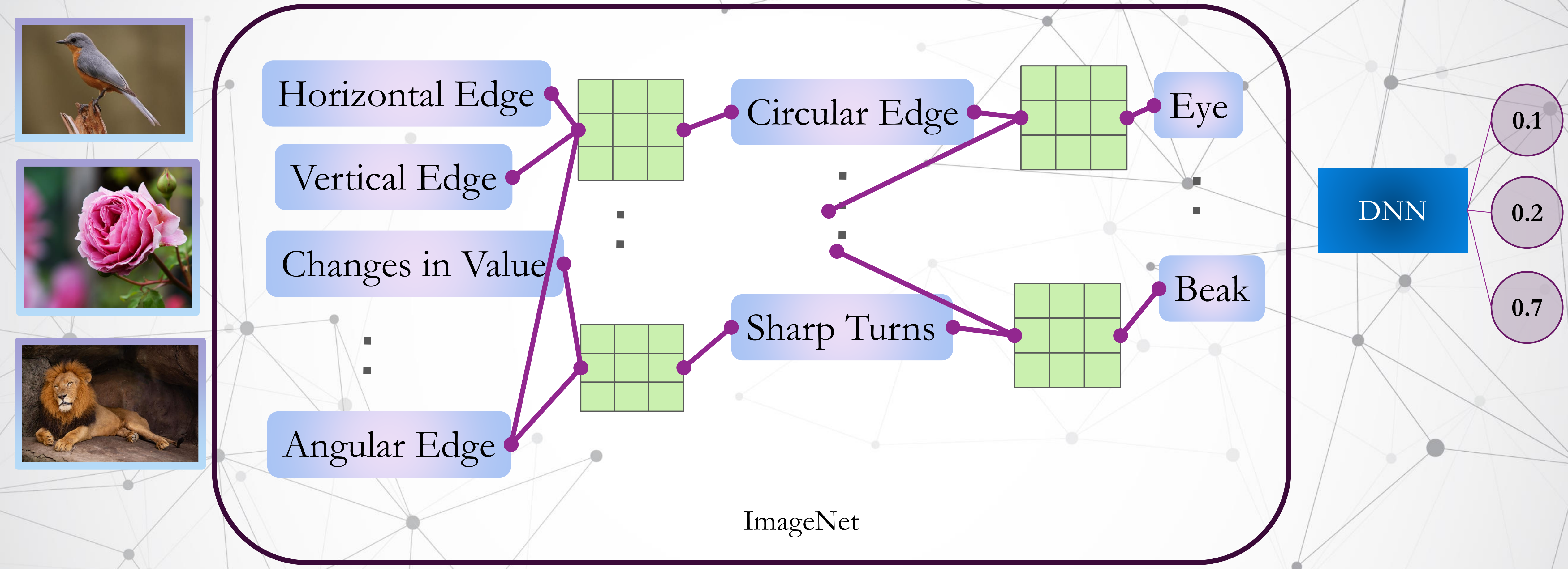
- 1.2 M Training Data*
- 50K Validation Data*
- 100K Test Data*
- 1000 Classes*

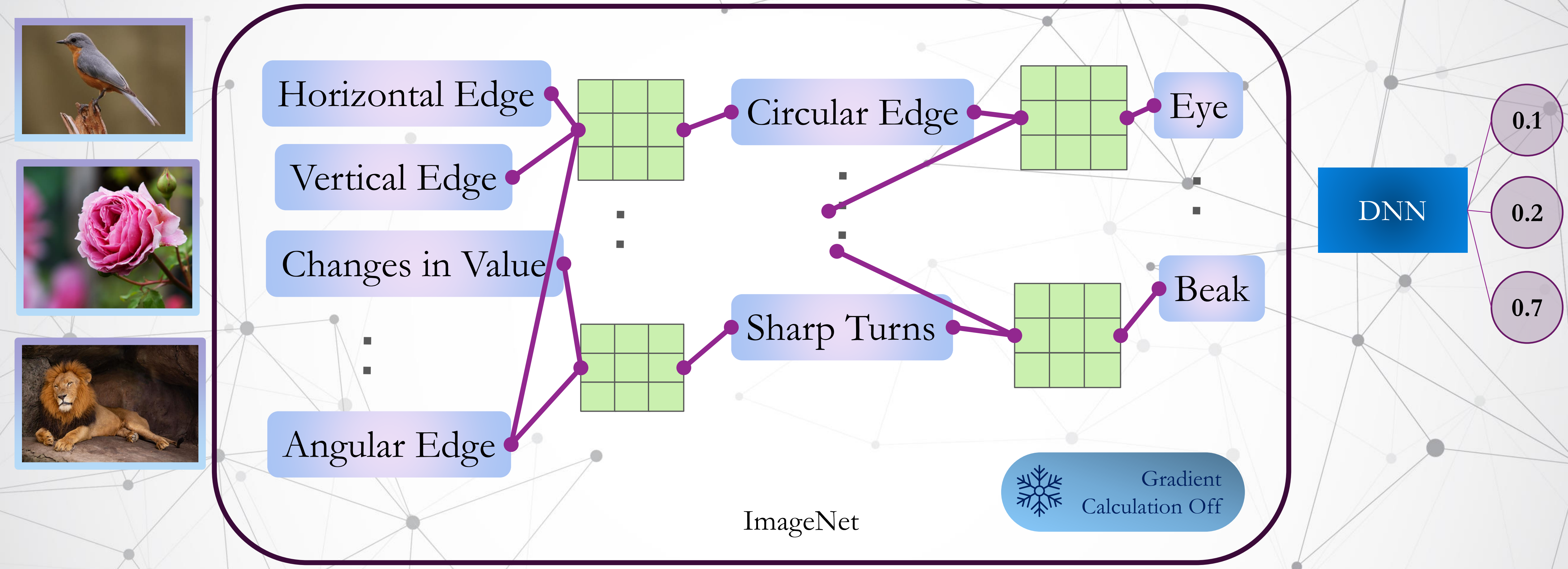
*indicates the 2010 Competition



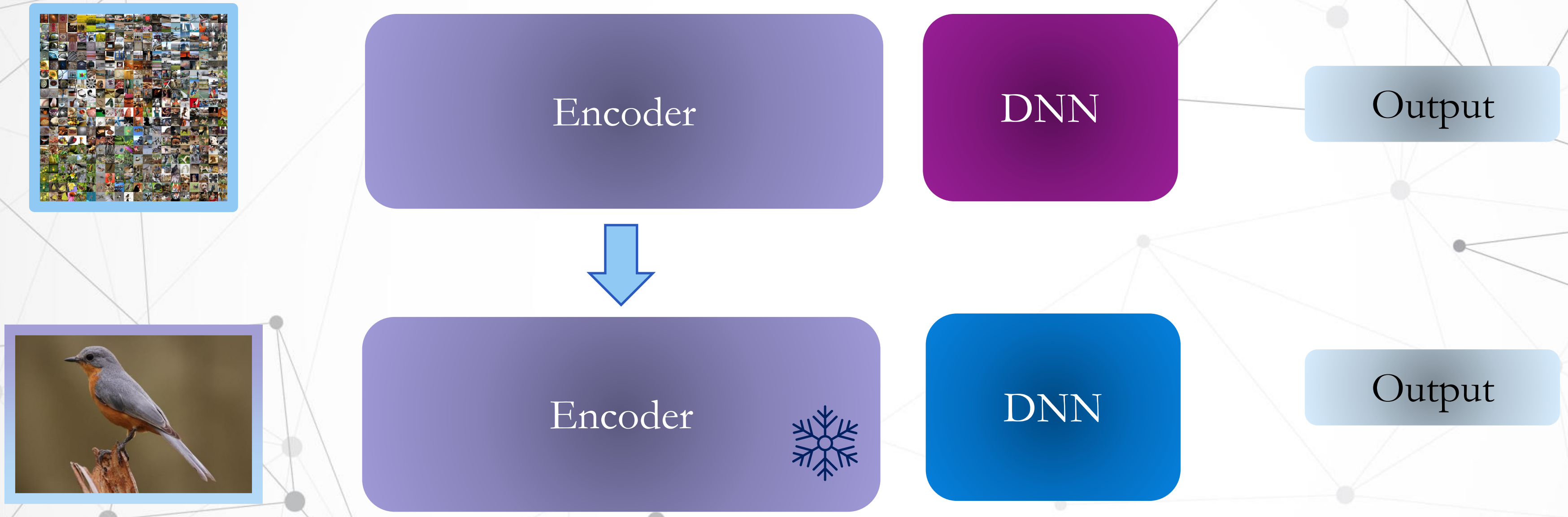
Horizontal Edge
Vertical Edge
Changes in Value
Angular Edge
Circular Edge
Sharp Turns
Eye
Beak
ImageNet
DNN
0.1
0.2
0.01
0.07


Horizontal Edge
Vertical Edge
Changes in Value
Angular Edge
Circular Edge
Sharp Turns
Eye
Beak
ImageNet
DNN
0.1
0.2
0.01
0.0.7

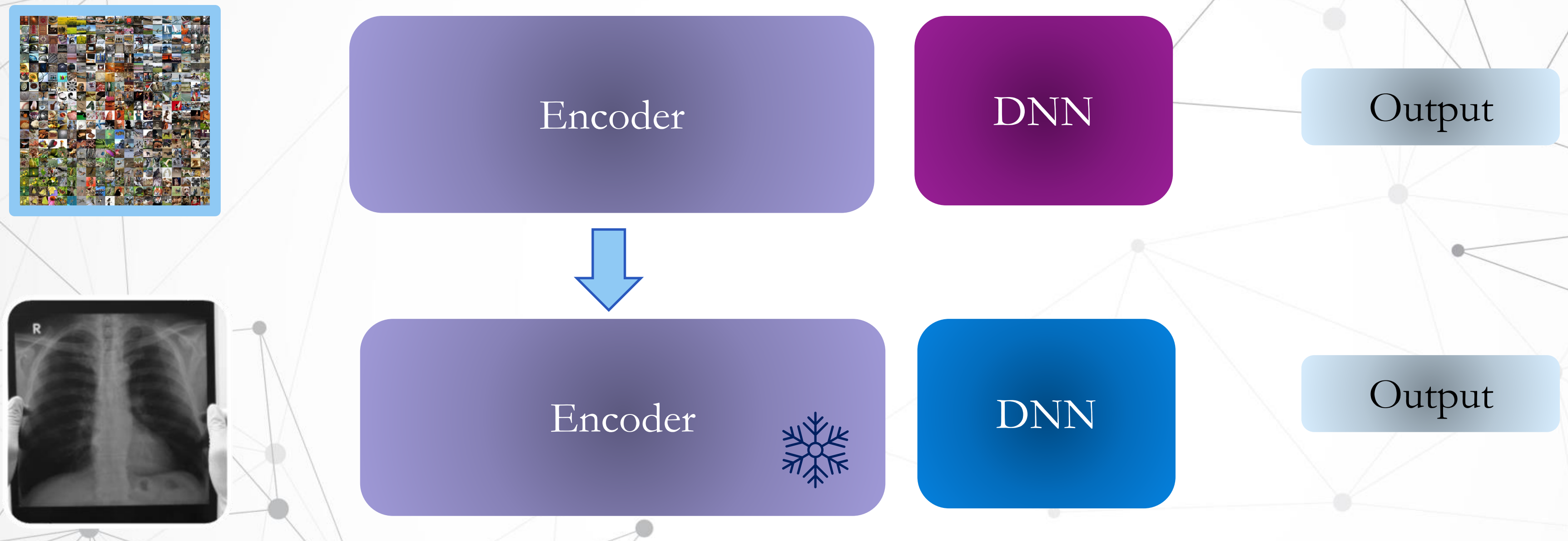

Horizontal Edge
Vertical Edge
Changes in Value
Angular Edge
Circular Edge
Sharp Turns
Eye
Beak
ImageNet
DNN
0.1
0.2
0.7


Horizontal Edge
Vertical Edge
Changes in Value
Angular Edge
Circular Edge
Sharp Turns
Eye
Beak
ImageNet
Gradient
Calculation Off
DNN
0.1
0.2
0.7

Transfer Learning

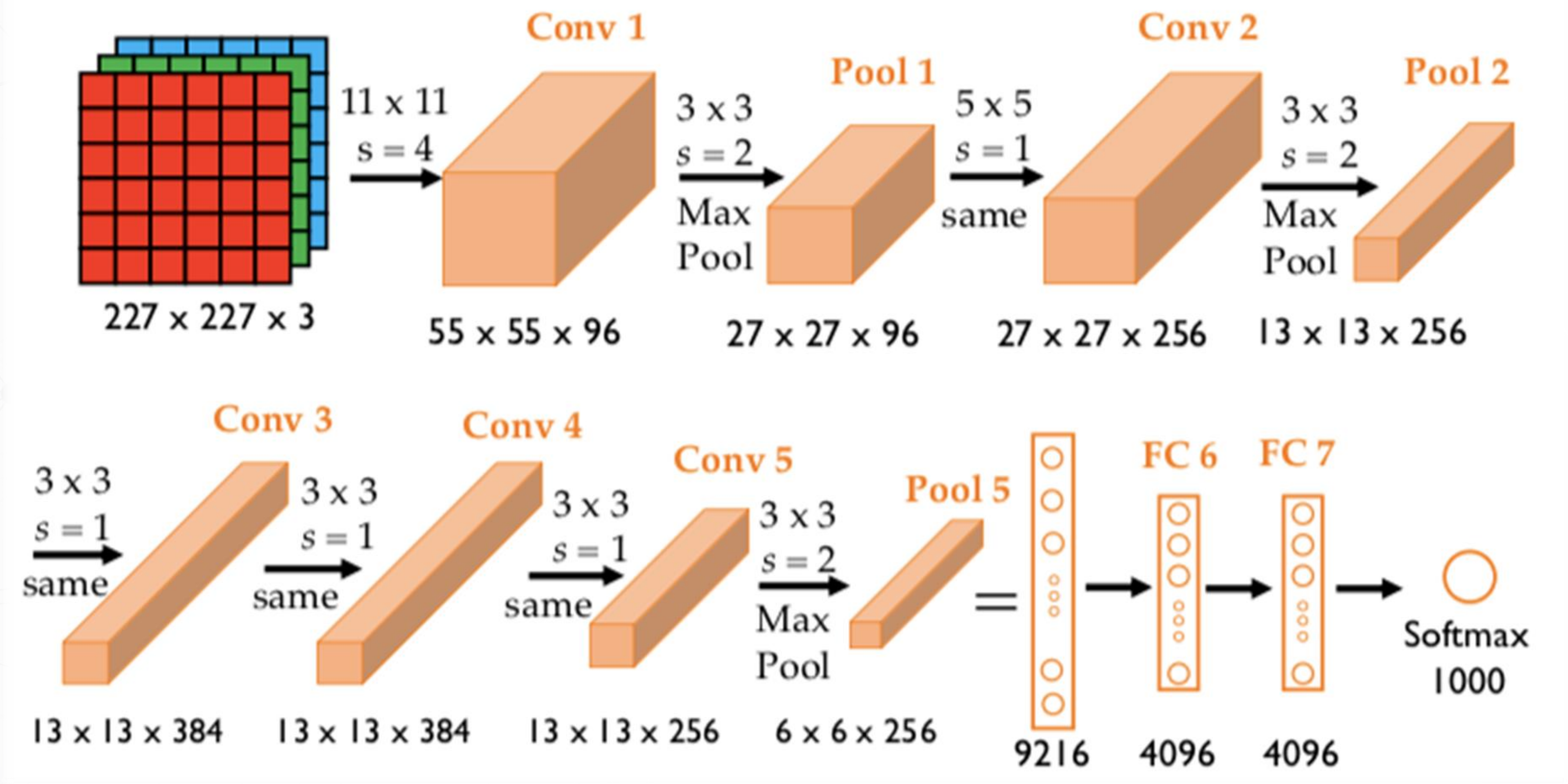


Transfer Learning



CNN Architectures

AlexNet

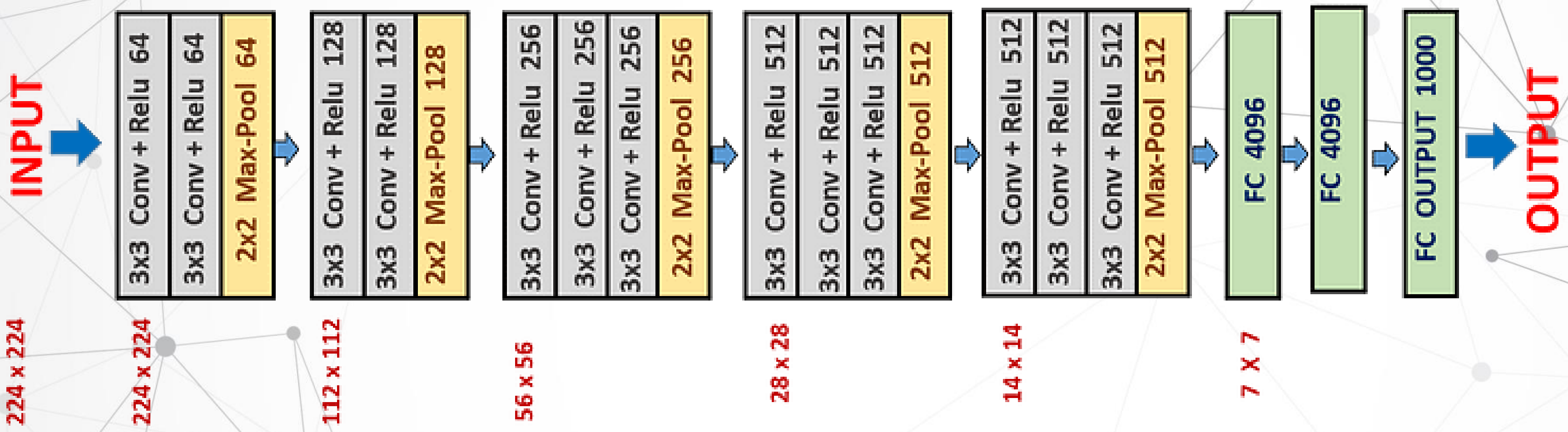


- Used ReLU
- Around 60 M Param
- Used 2 GTX 580
- (VRAM 6 GB Total)
- Overlapping Pooling
- Used Dropout

2012 ImageNet Classification with Deep Convolutional Neural Networks

Link

VGG



- Simplified Architecture
- Consists of 3x3 Conv ,2x2 MaxPool
- Resolution down by scale of 2, Channel up by scale of 2

2015 Very deep convolutional networks for large-scale image recognition

Link

2	1	1	0	0
0	1	0	1	0
3	0	-1	1	1
0	0	1	1	0
1	1	1	0	0

Convolutional Kernel

1	1	1	0	0
0	1	0	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

Convolutional Kernel

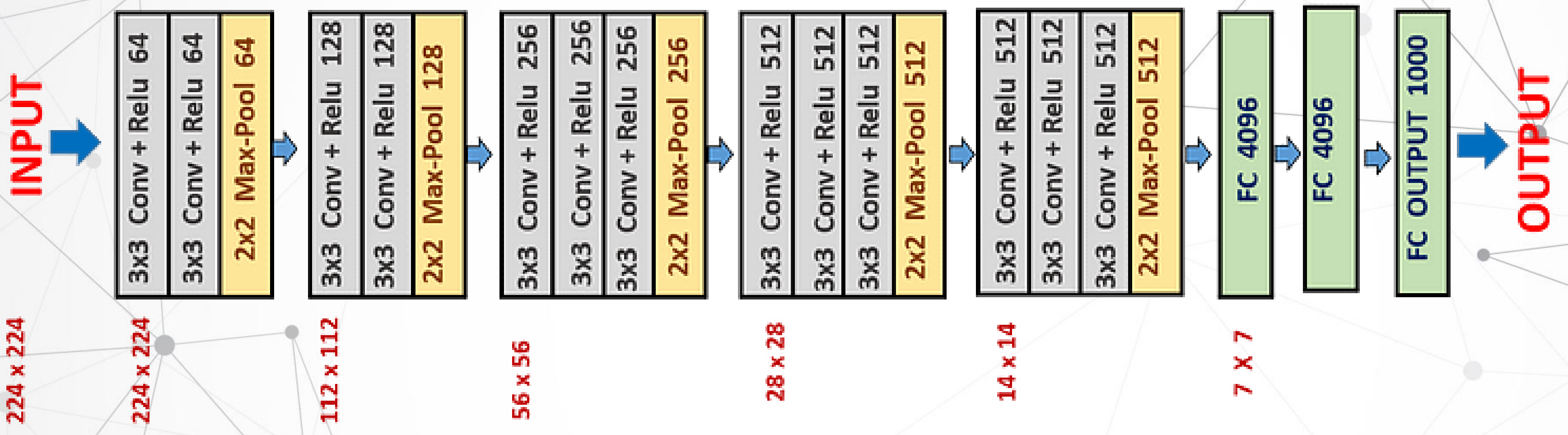
1	1	1	0	0
0	1	0	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

4	3	4
2	4	3
2	3	4

18

Applying a series of two 3x3 kernel is giving similar output to applying one 5x5 kernel, but
5x5 – 2x(3x3) = 7 parameters less per kernel

VGG



2015 Very deep convolutional networks for large-scale image recognition

[Link]

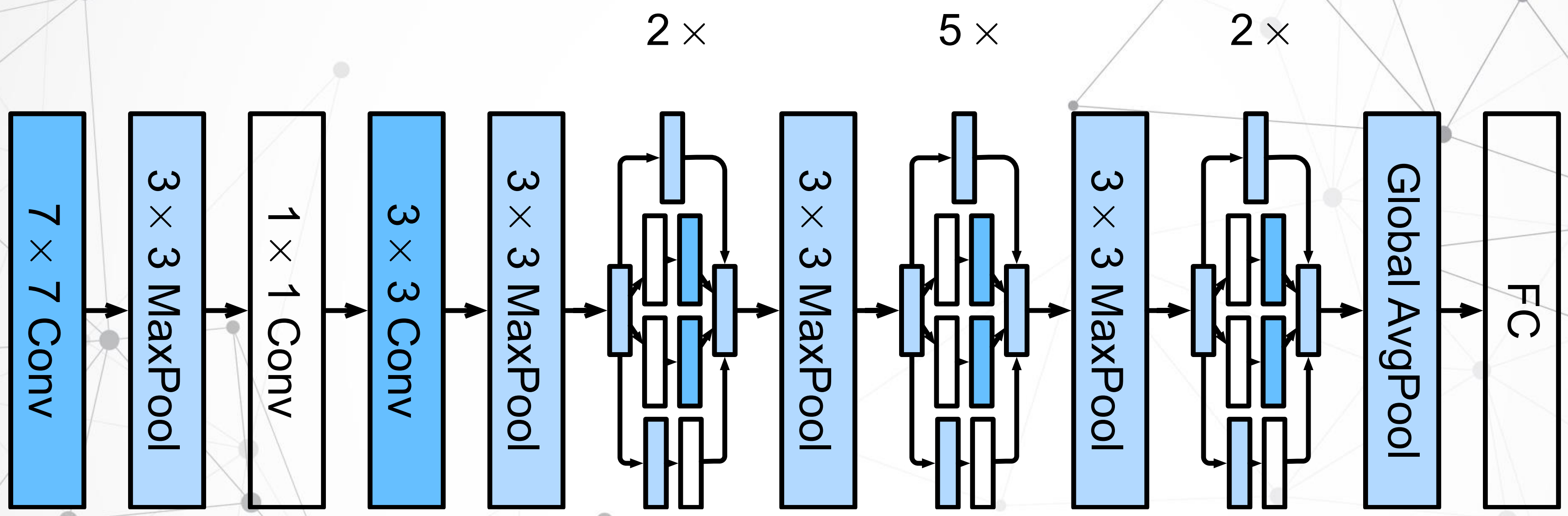
such layers have a 7×7 effective receptive field. So what have we gained by using, for instance, a stack of three 3×3 conv. layers instead of a single 7×7 layer? First, we incorporate three non-linear rectification layers instead of a single one, which makes the decision function more discriminative. Second, we decrease the number of parameters: assuming that both the input and the output of a three-layer 3×3 convolution stack has C channels, the stack is parametrised by $3\left(3^2C^2\right) = 27C^2$ weights; at the same time, a single 7×7 conv. layer would require $7^2C^2 = 49C^2$ parameters, i.e. 81% more. This can be seen as imposing a regularisation on the 7×7 conv. filters, forcing them to have a decomposition through the 3×3 filters (with non-linearity injected in between).

Second, we observe that the classification error decreases with the increased ConvNet depth: from 11 layers in A to 19 layers in E. Notably, in spite of the same depth, the configuration C (which contains three 1×1 conv. layers), performs worse than the configuration D, which uses 3×3 conv. layers throughout the network. This indicates that while the additional non-linearity does help (C is better than B), it is also important to capture spatial context by using conv. filters with non-trivial receptive fields (D is better than C). The error rate of our architecture saturates when the depth reaches 19 layers, but even deeper models might be beneficial for larger datasets. We also compared the net B with a shallow net with five 5×5 conv. layers, which was derived from B by replacing each pair of 3×3 conv. layers with a single 5×5 conv. layer (which has the same receptive field as explained in Sect. 2.3). The top-1 error of the shallow net was measured to be 7% higher than that of B (on a center crop), which confirms that a deep net with small filters outperforms a shallow net with larger filters.

2015

Very deep convolutional networks for large-scale image recognition

GoogLeNet



- 22 layers
- (27 Including Pooliing)

2015 Going Deeper with Convolutions

Link

Inception

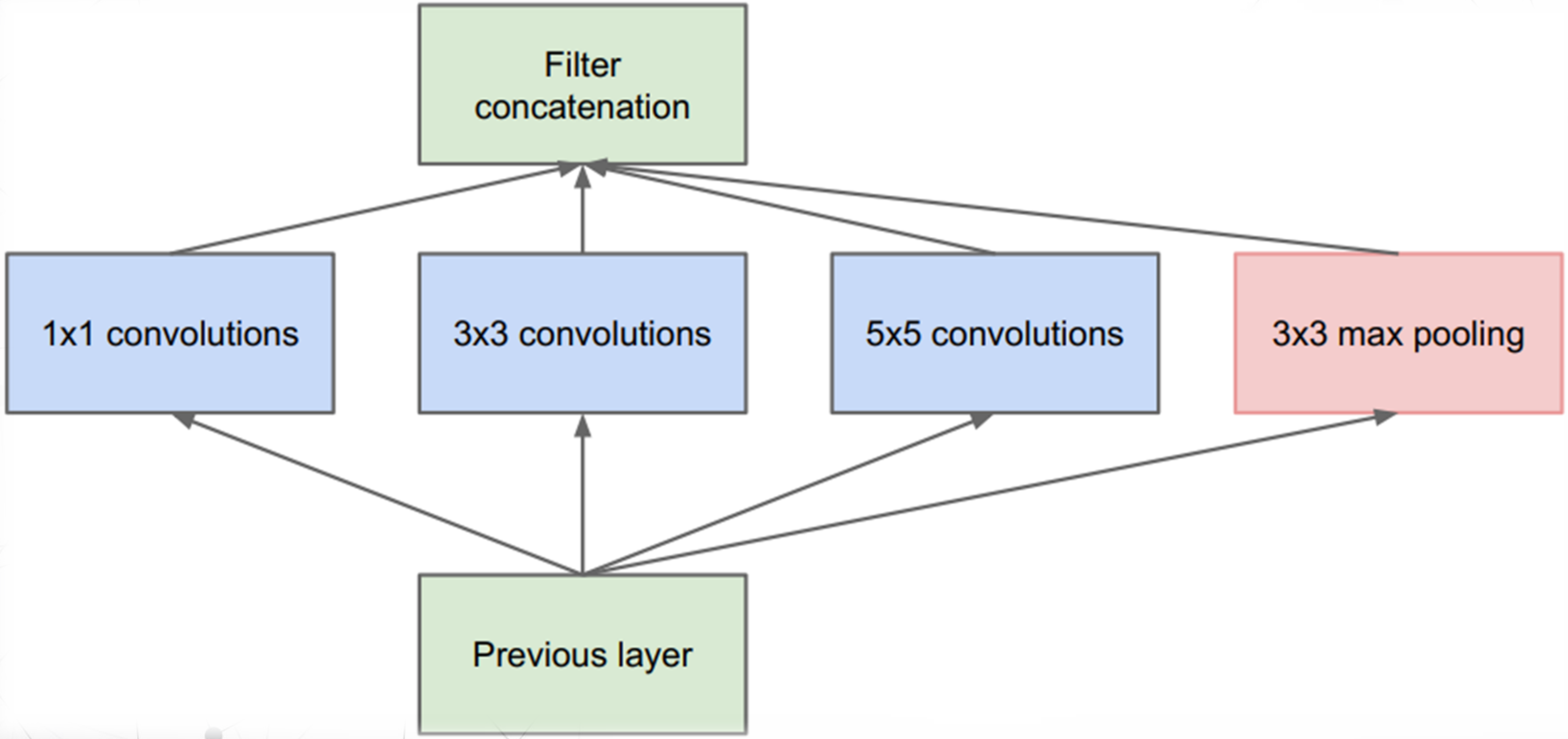
References

[1] Know your meme: We need to go deeper. http://knowyourmeme.com/memes/we-need-to-go-deeper. Accessed: 2014-09-15.

In this paper, we will focus on an efficient deep neural network architecture for computer vision, codenamed Inception, which derives its name from the Network in network paper by Lin et al [12] in conjunction with the famous "we need to go deeper" internet meme [1]. In our case, the

2015 Going Deeper with Convolutions

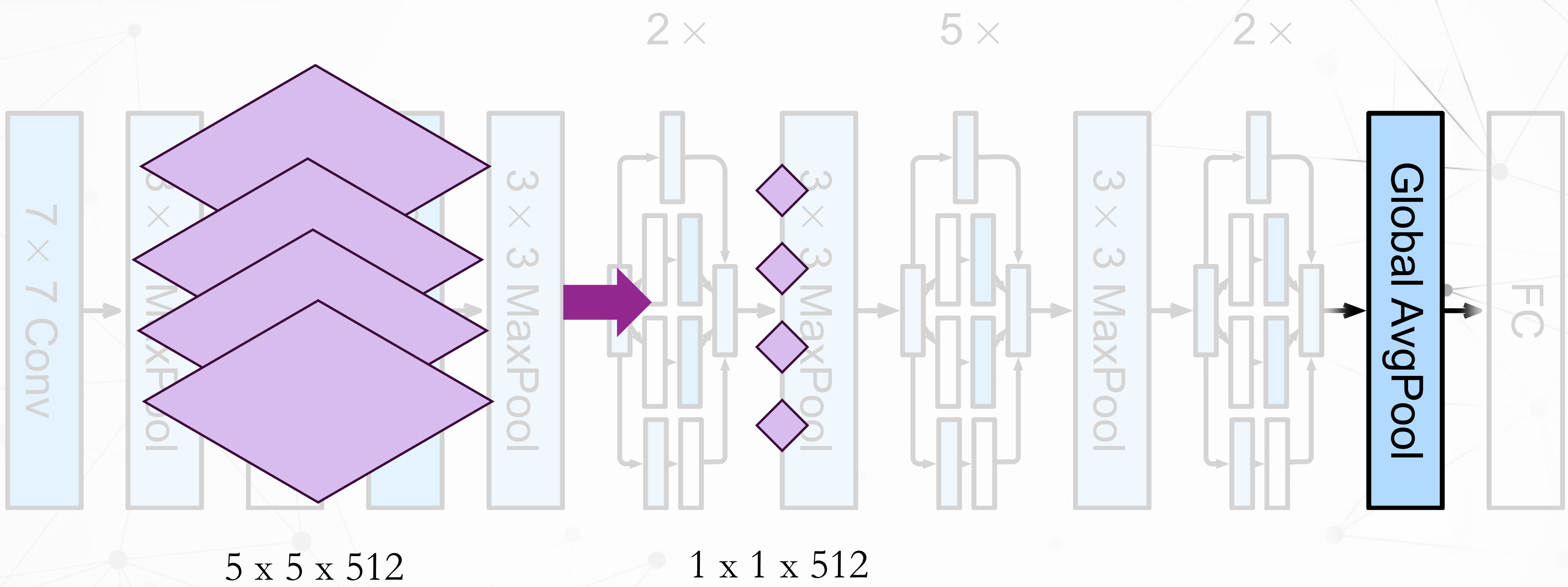
Inception



2015 Going Deeper with Convolutions

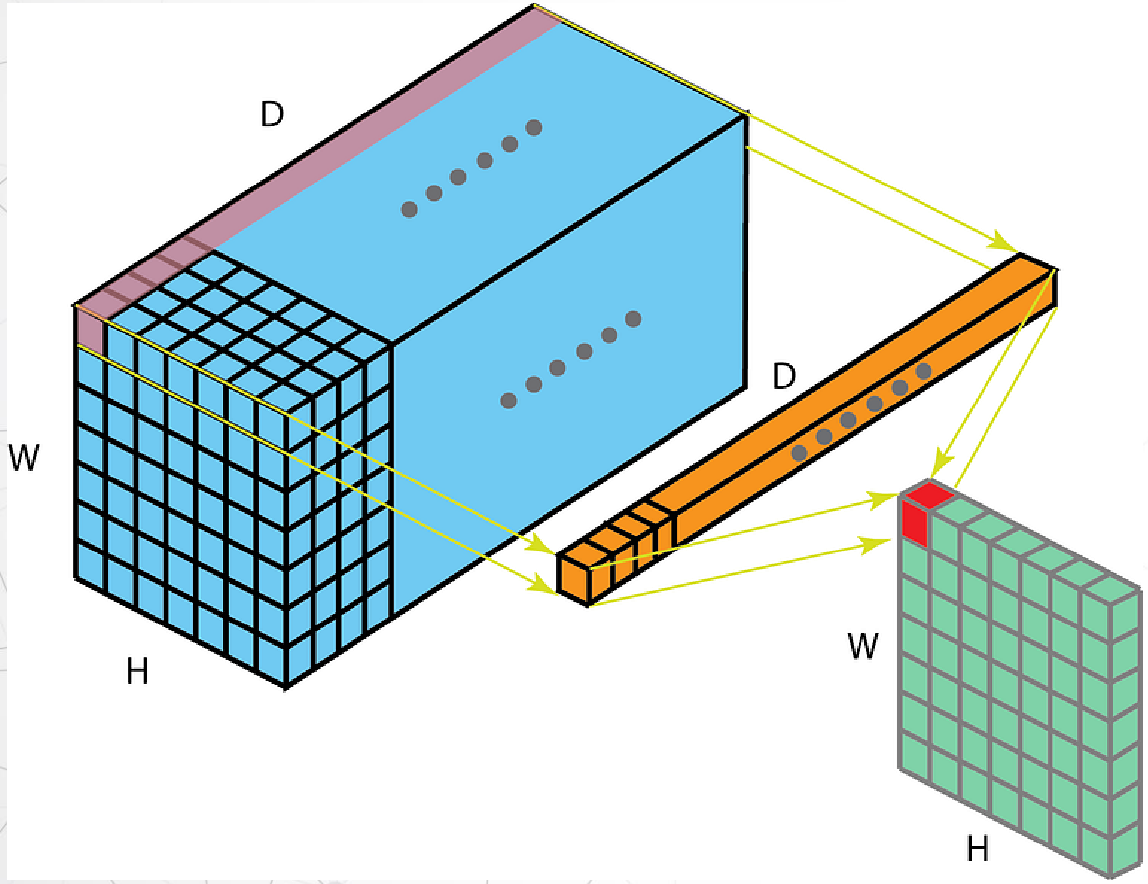
Link

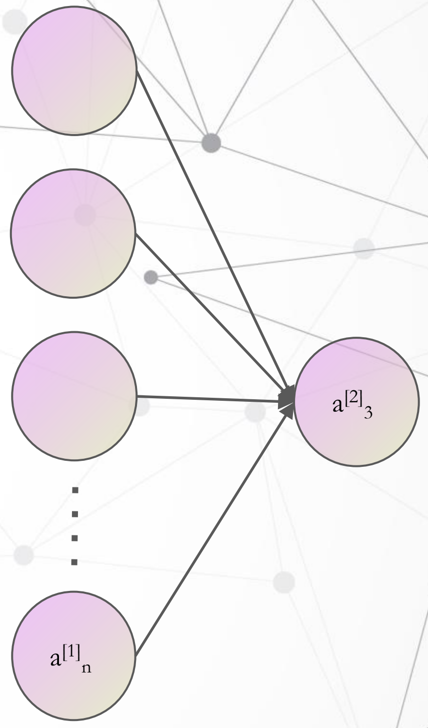
Global Average Pooling



a major effect. We found that a move from fully connected layers to average pooling improved the top-1 accuracy by about 0.6%, however the use of dropout remained essential even after removing the fully connected layers.

1x1 Convolutions

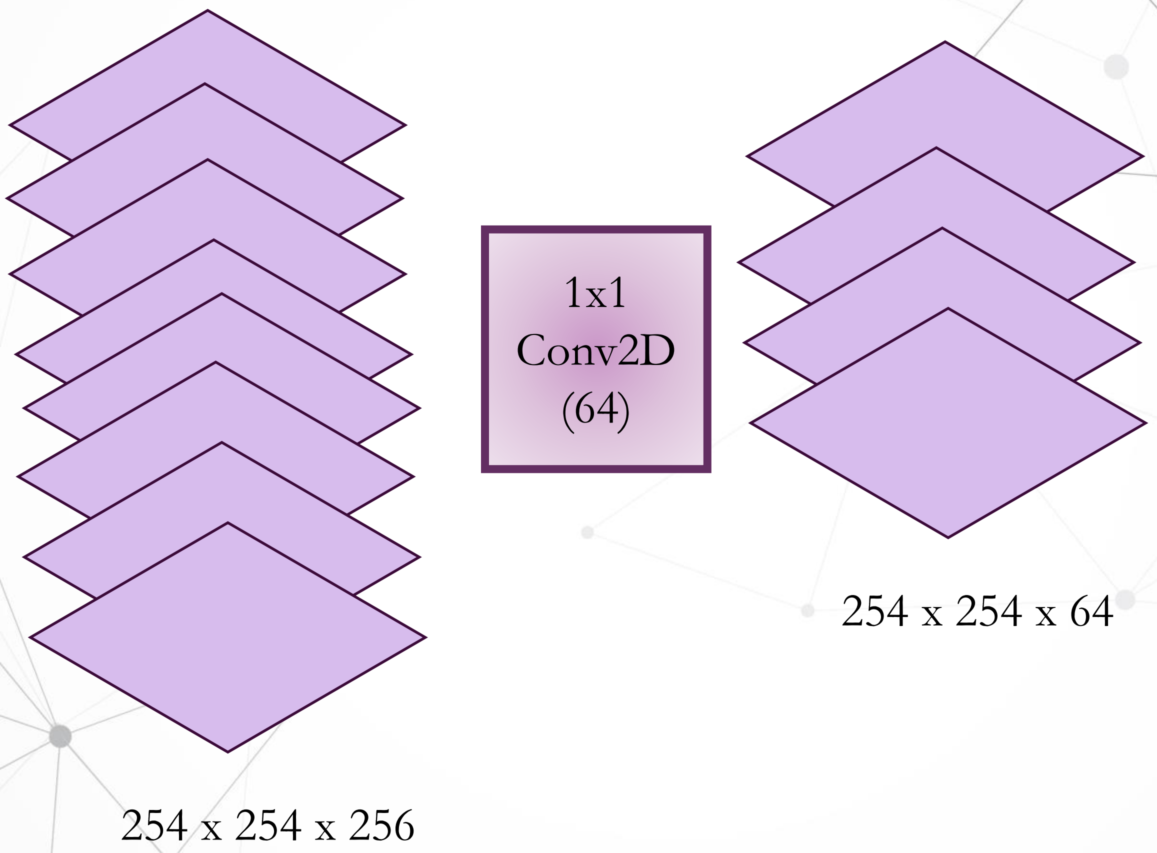




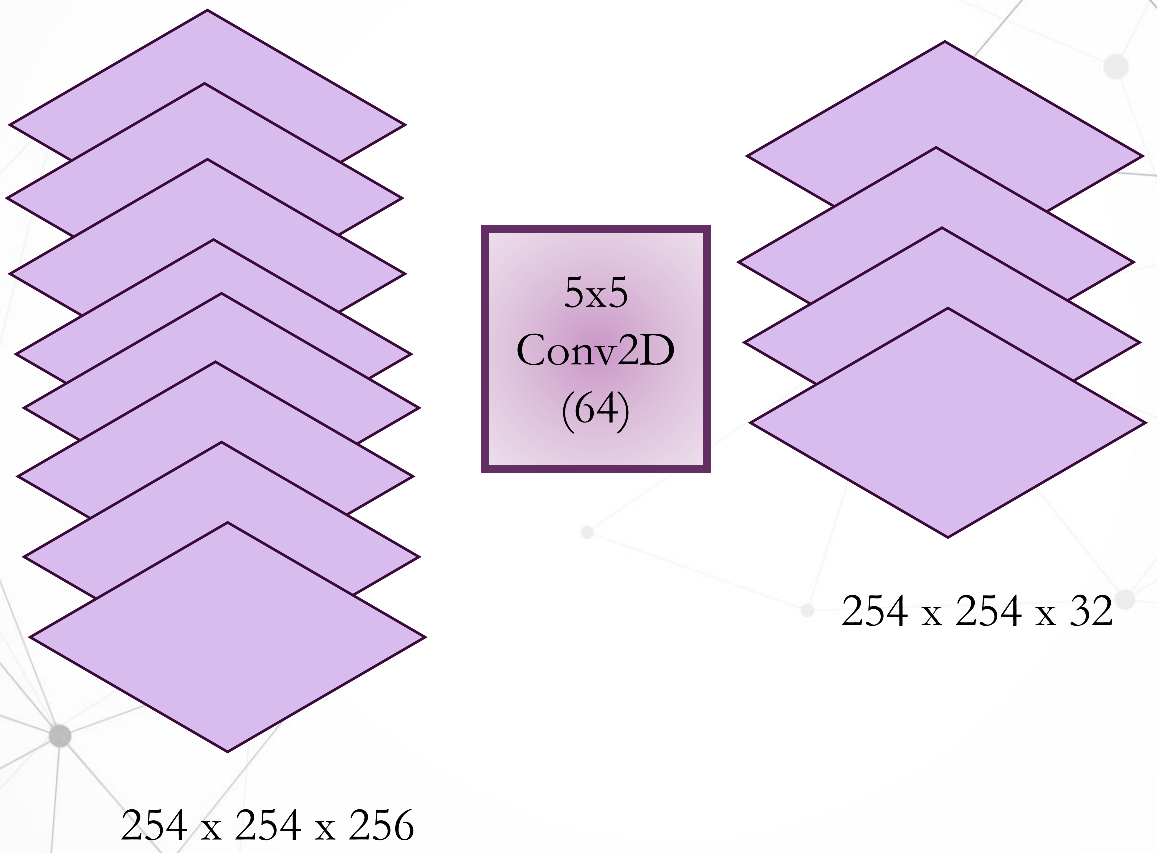
1x1 Convolutions can estimate the information across all the input channels and represent them in a smaller or larger number of channels, depending on the filter size specified

Link

1x1 Convolutions

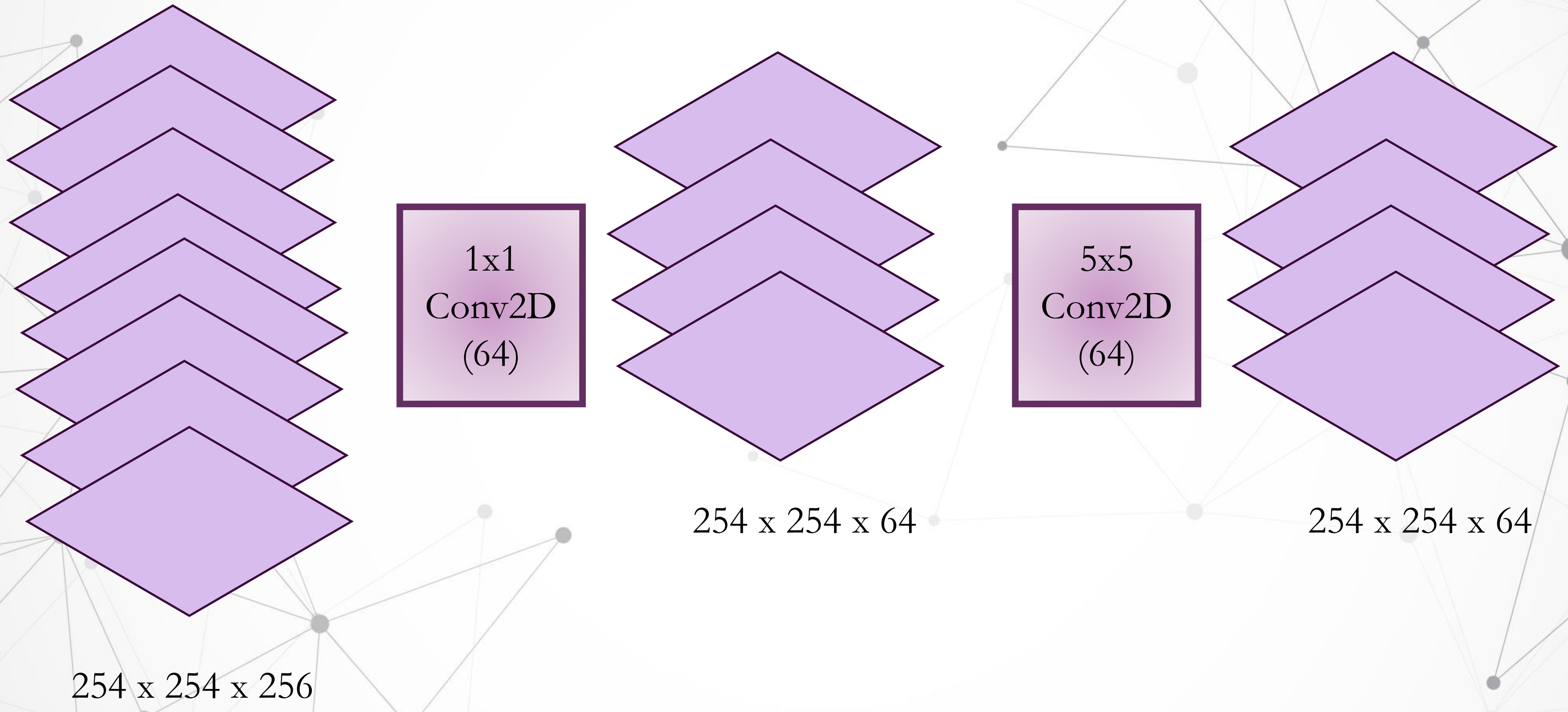


(256x1)x64 = 16,384



$(5x5x256) \times 64 = 409{,}600$

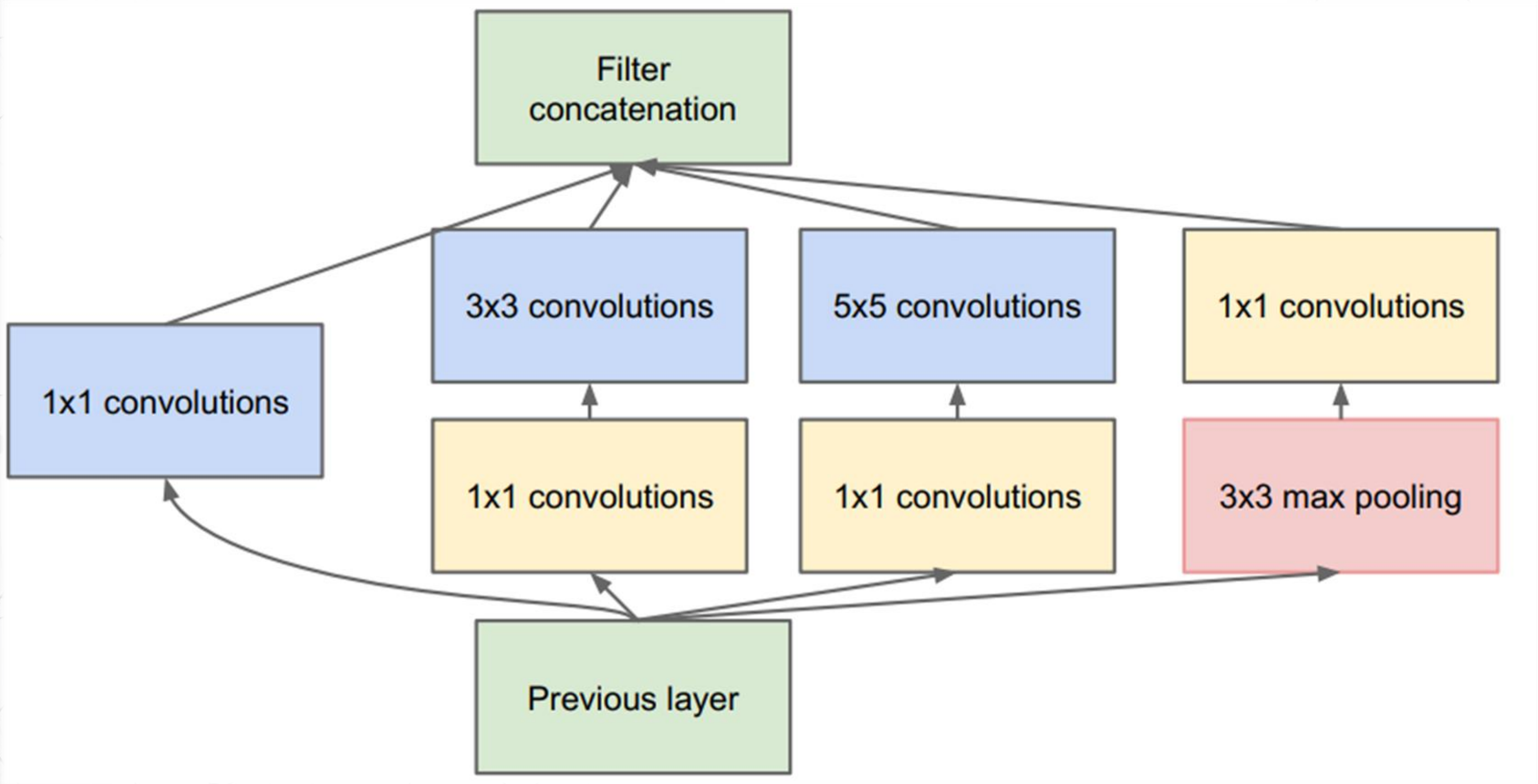
Bottleneck



(256x1)x64 + (5x5x64) x 64 = 118,784

(5x5x256) x 64 = 409,600

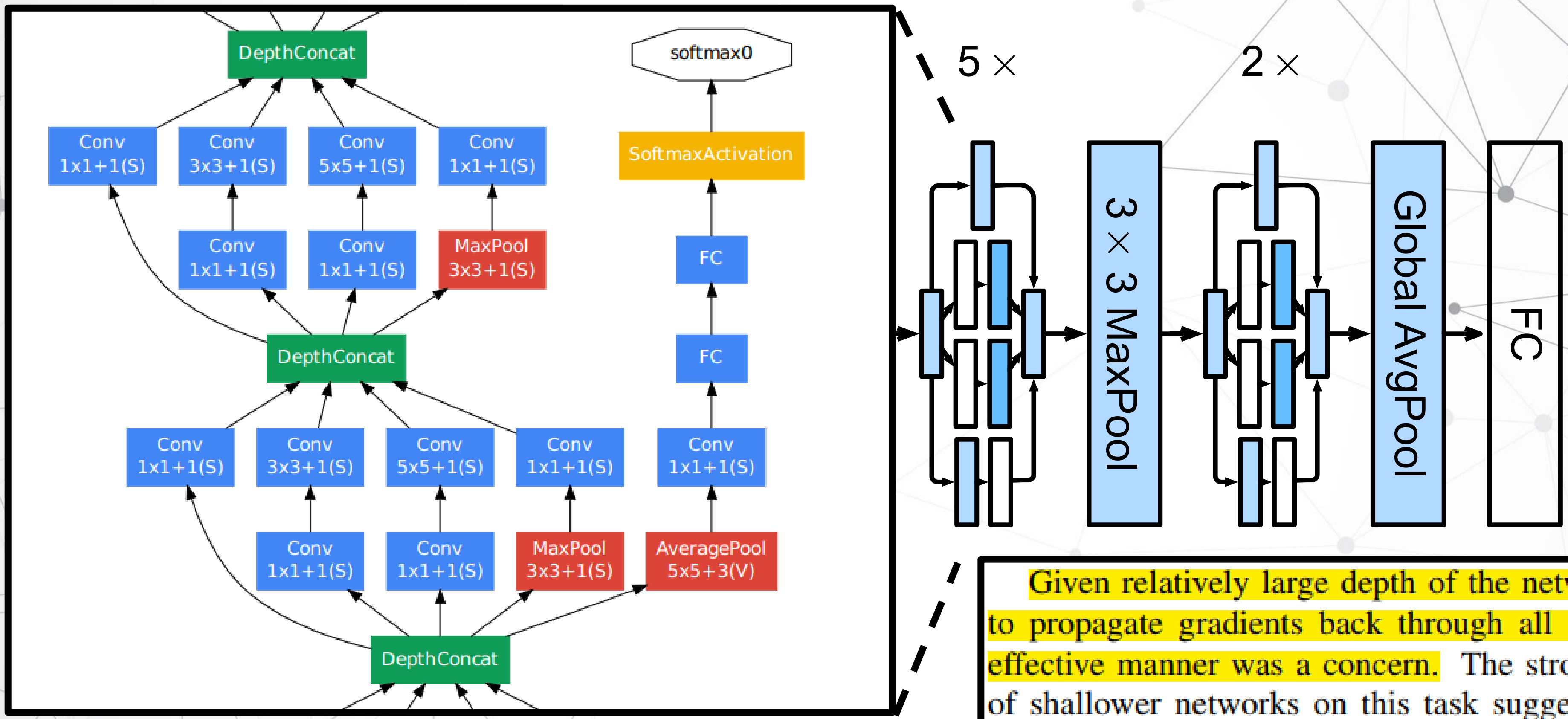
GoogLeNet



2015 Going Deeper with Convolutions

[Link]

GoogLeNet

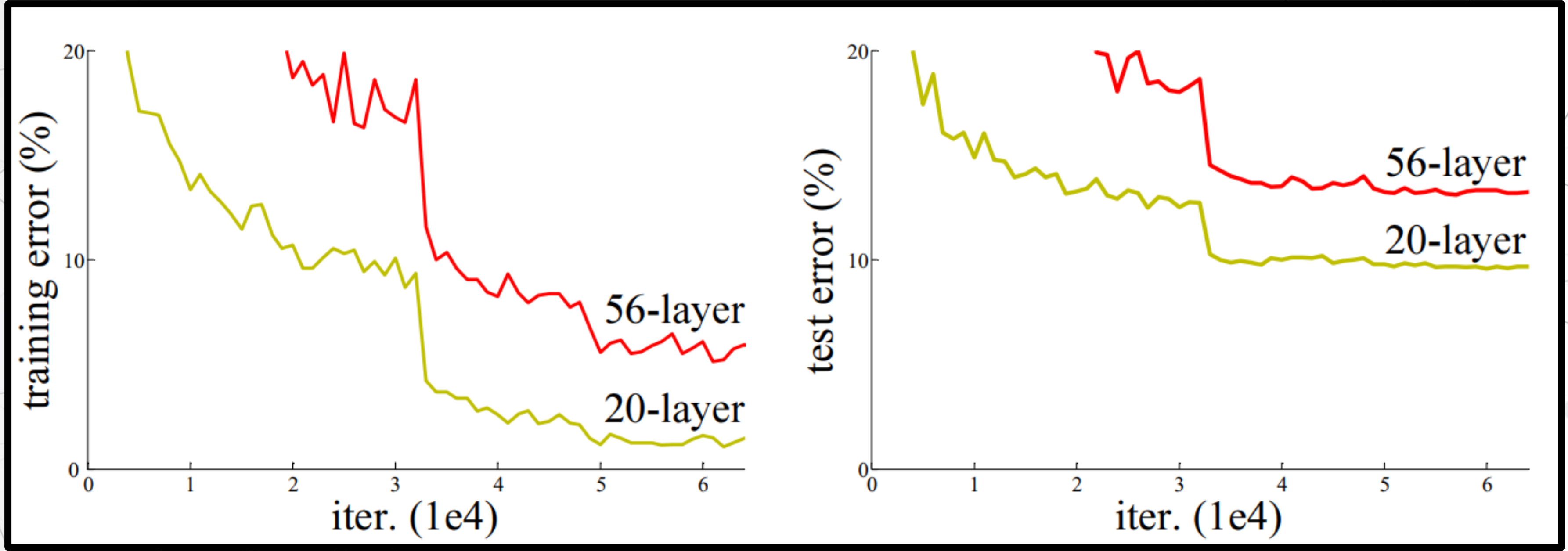


Given relatively large depth of the network, the ability to propagate gradients back through all the layers in an effective manner was a concern. The strong performance of shallower networks on this task suggests that the features produced by the layers in the middle of the network should be very discriminative. By adding auxiliary classifiers connected to these intermediate layers, discrimination in the lower stages in the classifier was expected. This was thought to combat the vanishing gradient problem while

Link

2015 Going Deeper with Convolutions

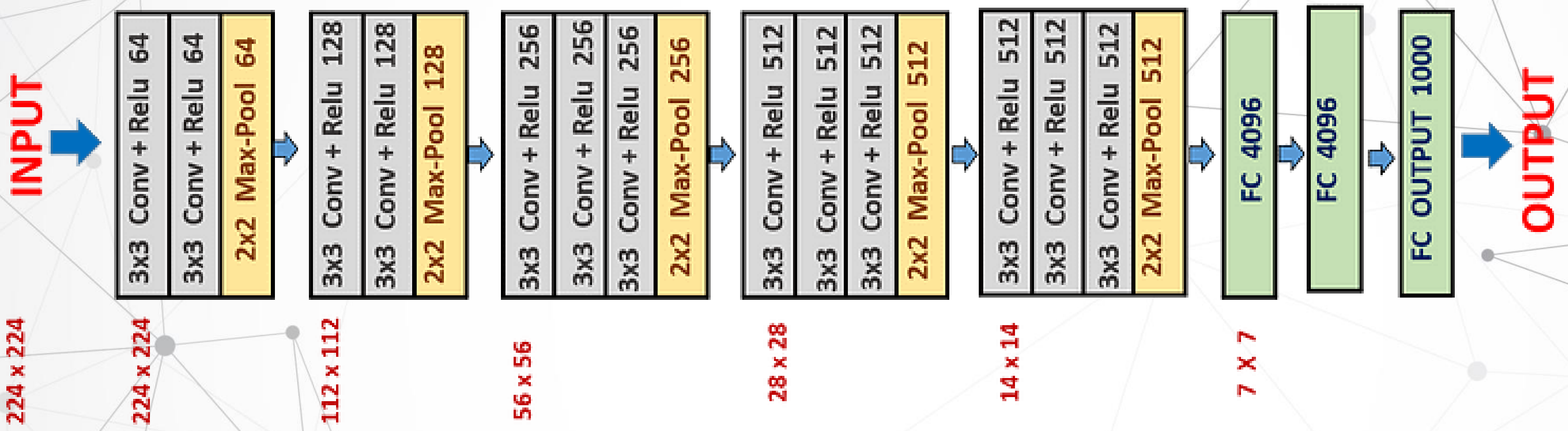
ResNet



When deeper networks are able to start converging, a *degradation* problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is *not caused by overfitting*, and adding more layers to a suitably deep model leads to *higher training error*, as reported in [10, 41] and thoroughly verified by our experiments. Fig. 1 shows a typical example.

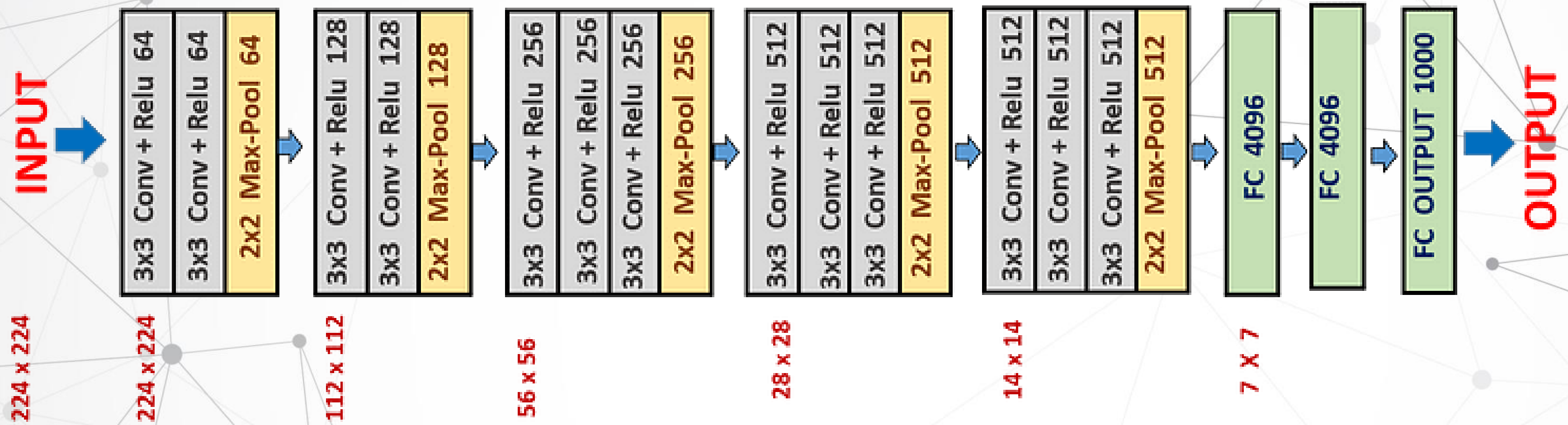
2016 Deep Residual Learning for Image Recognition

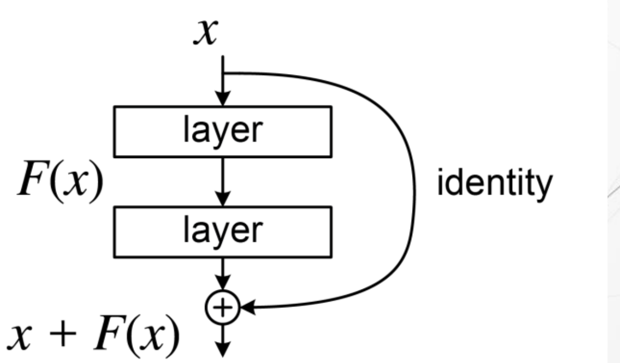
ResNet



2016 Deep Residual Learning for Image Recognition

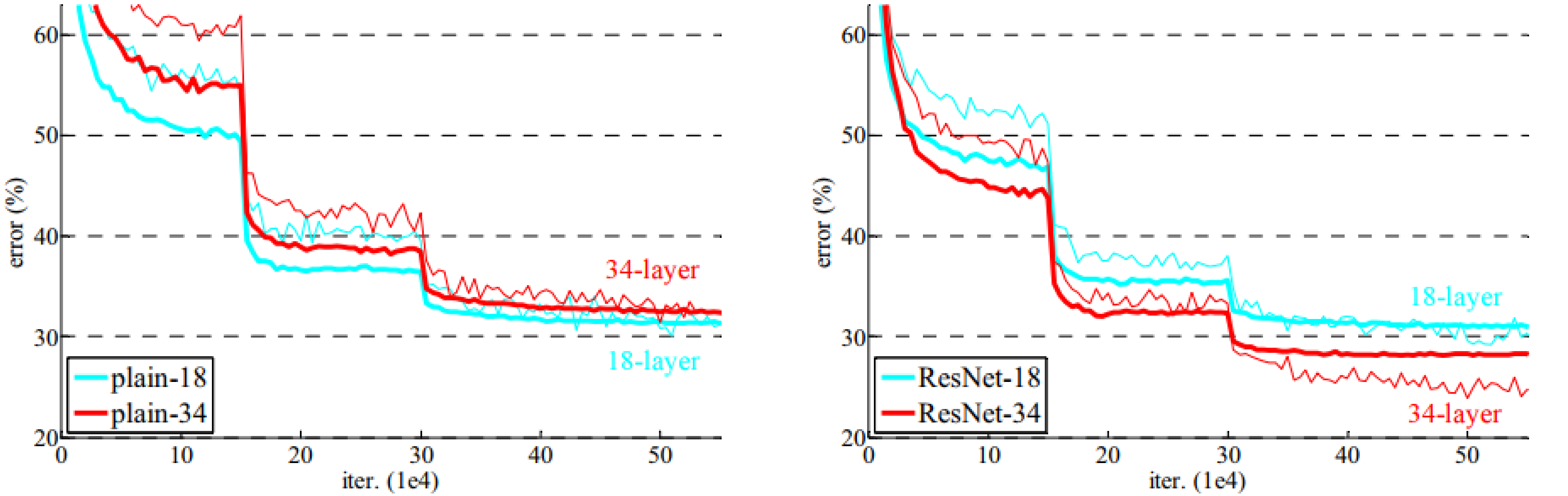
ResNet



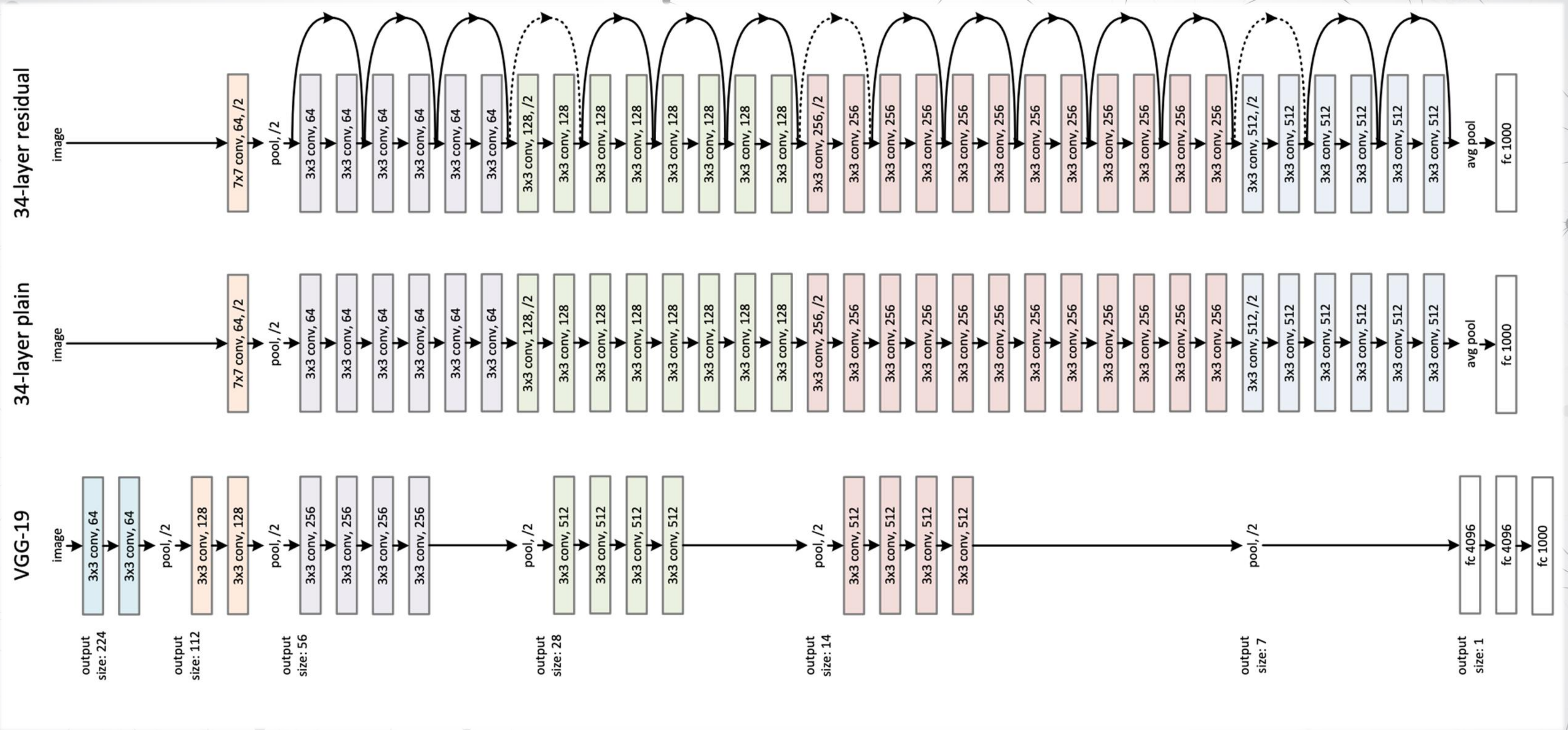


2016 Deep Residual Learning for Image Recognition

ResNet



2016 Deep Residual Learning for Image Recognition



2016 Deep Residual Learning for Image Recognition

2012 ImageNet Classification with Deep Convolutional Neural Networks

2015 Very deep convolutional networks for large-scale image recognition

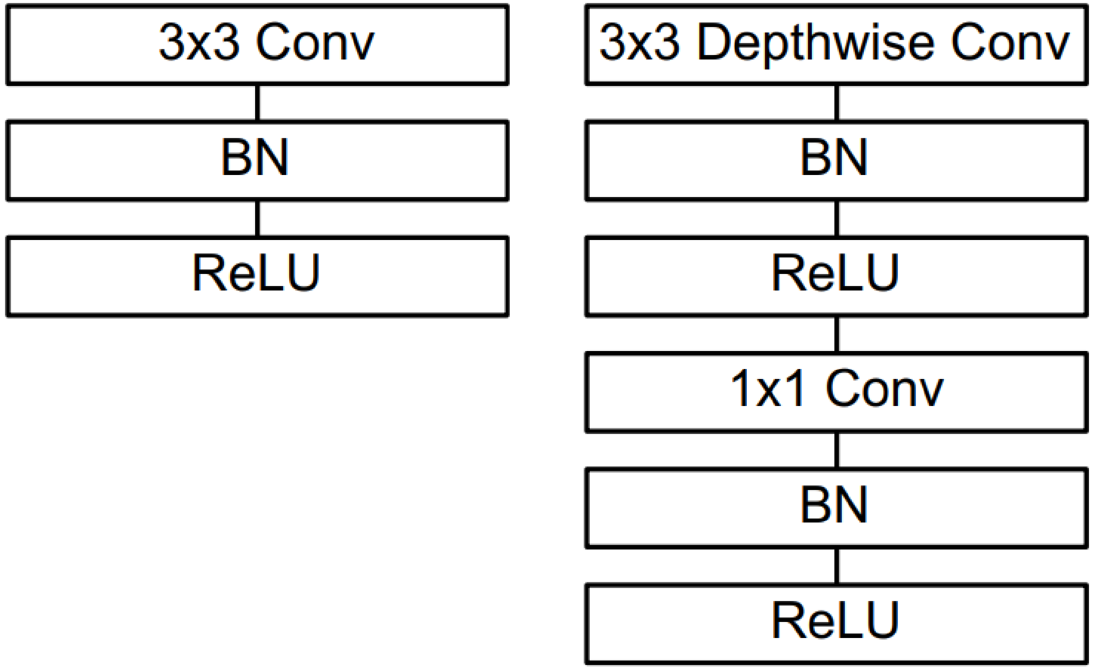
2015 Going Deeper with Convolutions

2016 Deep Residual Learning for Image Recognition

MobileNet

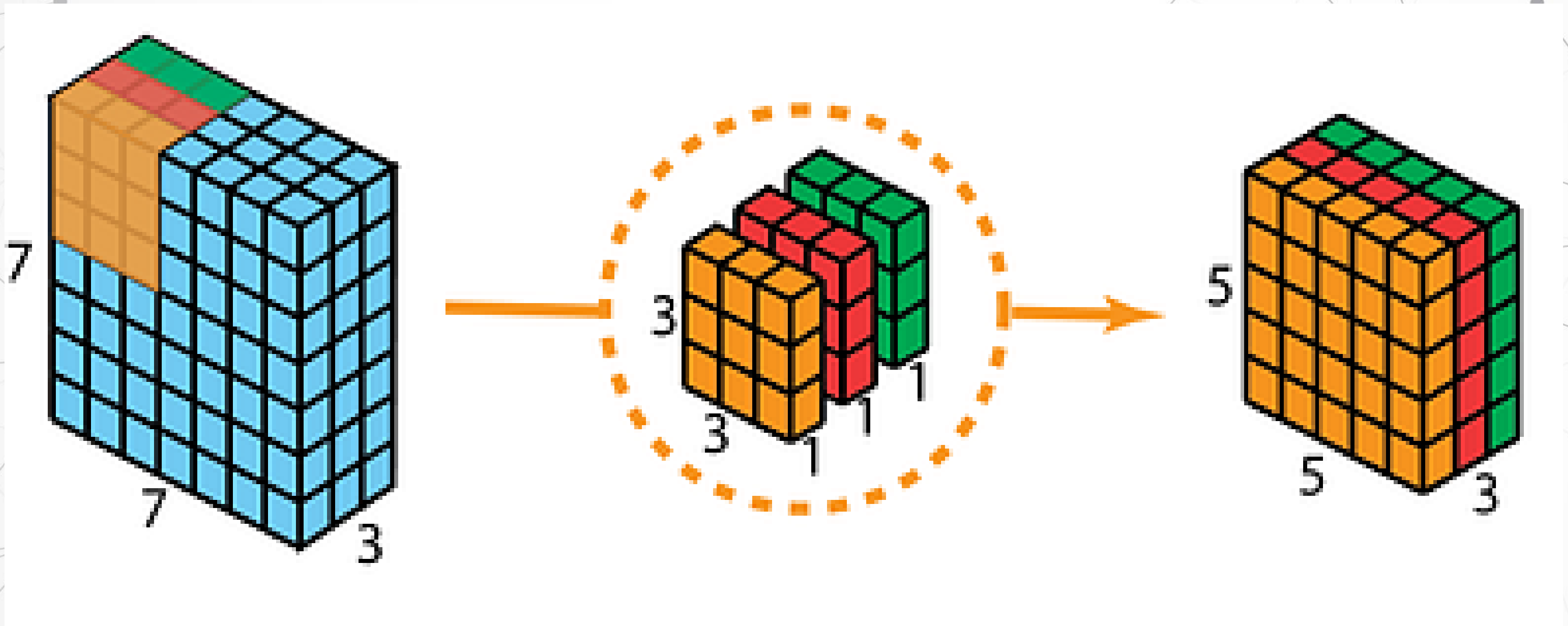
Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

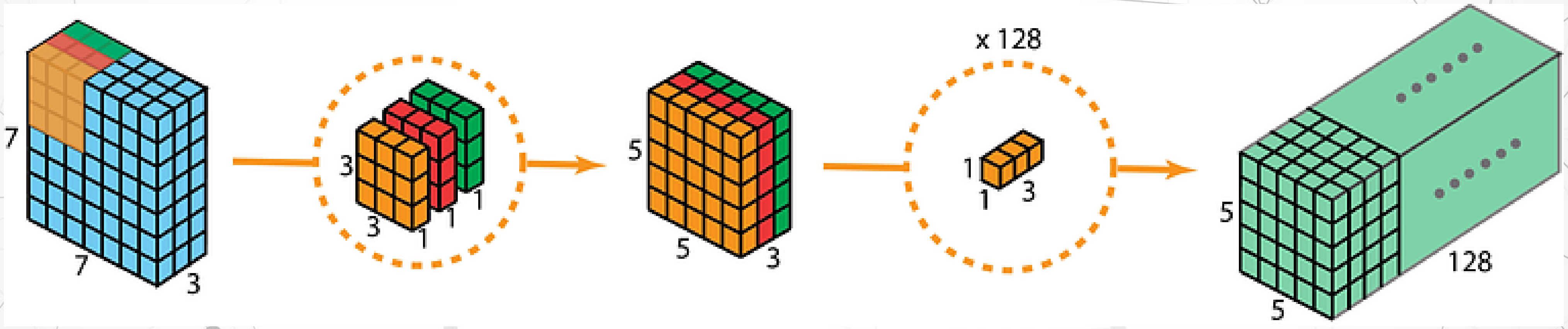


2017

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

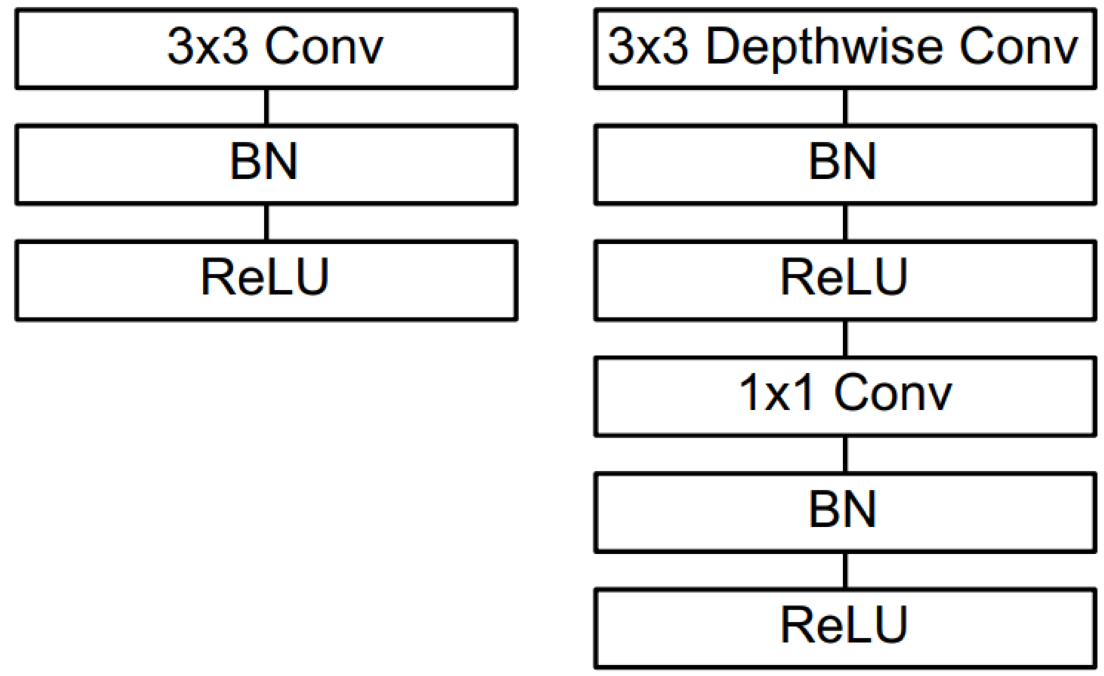


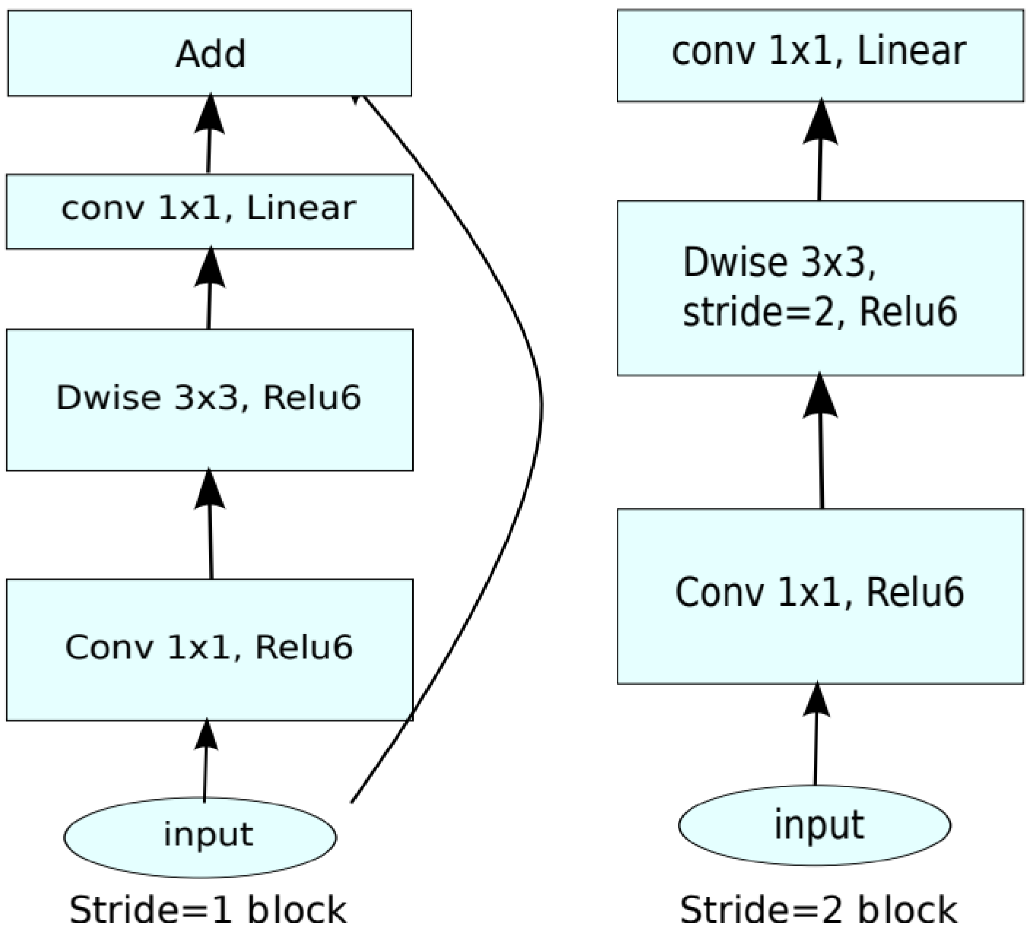
[Link]



Link

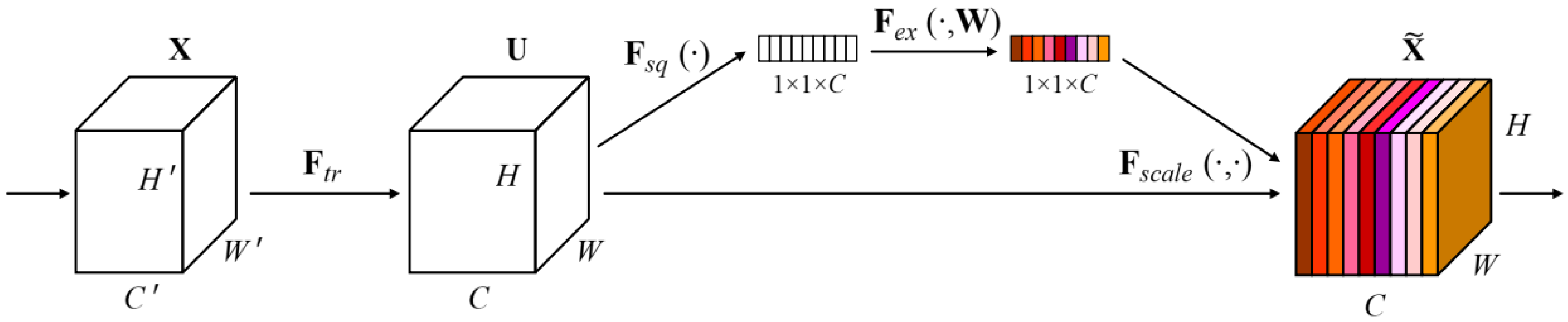
MobileNetV2





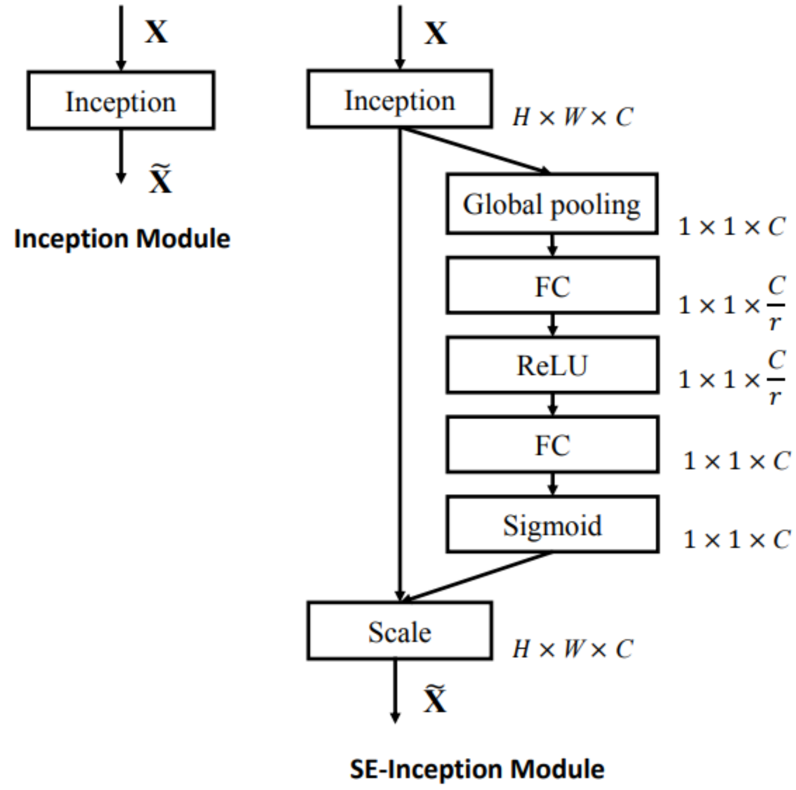
2018 MobileNetV2: Inverted Residuals and Linear Bottlenecks

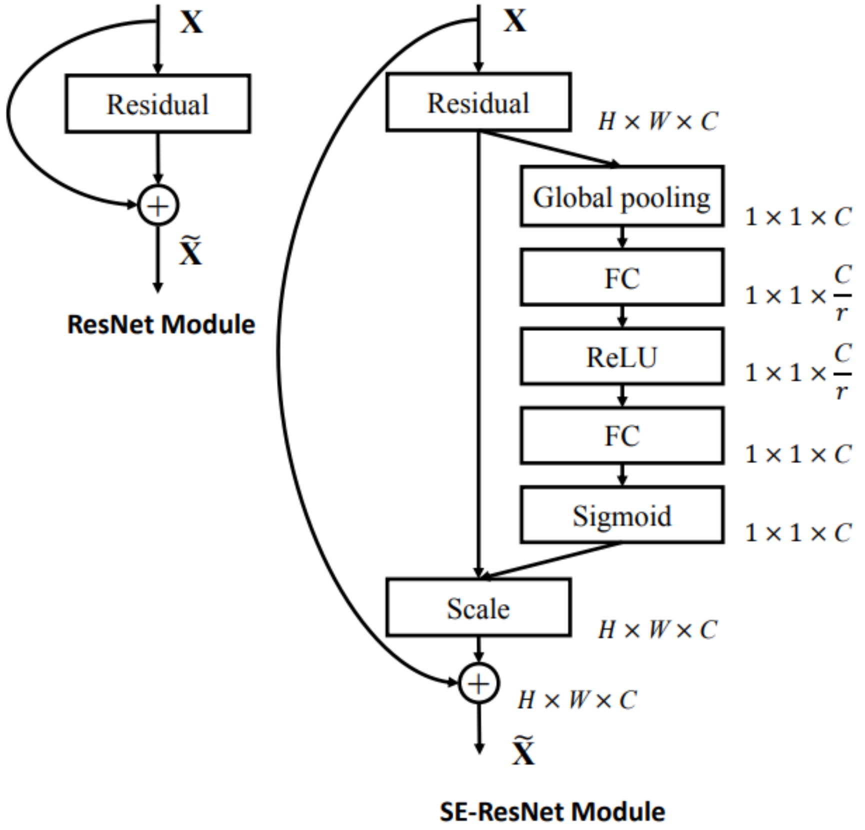
SENet



2018 Squeeze-and-Excitation Networks

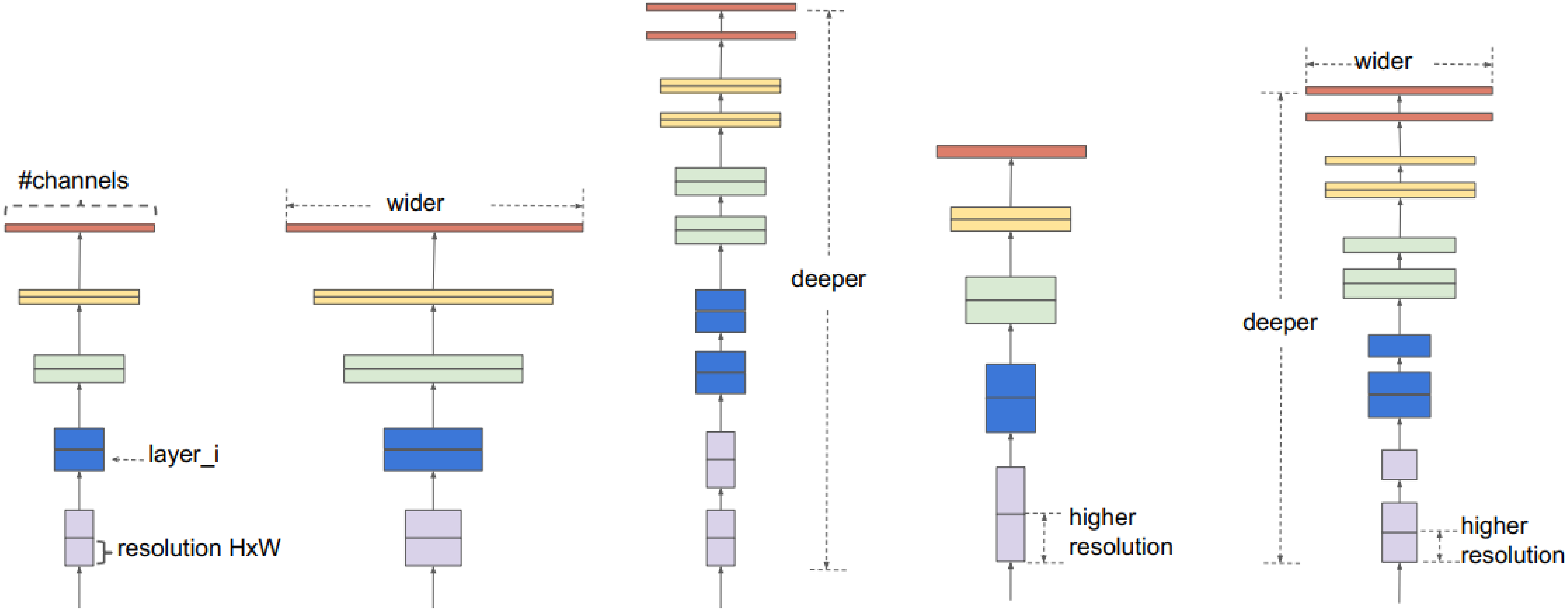
SENet





2018 Squeeze-and-Excitation Networks

EfficientNet



(a) baseline (b) width scaling (c) depth scaling (d) resolution scaling (e) compound scaling

2019

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

EfficientNet

In this paper, we propose a new **compound scaling method**, which use a compound coefficient ϕ to uniformly scales network width, depth, and resolution in a principled way:

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{3}$$

- STEP 1: we first fix $\phi = 1$, assuming twice more resources available, and do a small grid search of α, β, γ based on Equation 2 and 3. In particular, we find the best values for EfficientNet-B0 are $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$, under constraint of $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$.
- STEP 2: we then fix α, β, γ as constants and scale up baseline network with different ϕ using Equation 3, to obtain EfficientNet-B1 to B7 (Details in Table 2).

Any Question?