Case Studies

FPT Education

FPT UNIVERSITY

Learning Objectives:

- Implement the basic building blocks of ResNets in a deep neural network using Keras.
- Train a state-of-the-art neural network for image classification.
- Implement a skip connection in your network.
- Create a dataset from a directory.
- Preprocess and augment data using the Keras Sequential API.
- Adapt a pretrained model to new data and train a classifier using the Functional API and MobileNet.
- Fine-tine a classifier's final layers to improve accuracy

Case Studies

FPT Education

FPT UNIVERSITY

FPT Education

FPT UNIVERSITY

Case Studies

Why look at case studies?

Why look at case studies?

- The past few years of computer vision research have been focused on how to put together the basic building blocks of convolutional layers, pooling layers, and fully connected layers to form effective convolutional neural networks. By looking at case studies of effective networks, one can gain insights and ideas that could be useful for their own work, even if they are not working on computer vision tasks.
- Several classic networks that have been successful in computer vision, including LeNet-5, AlexNet, VGG, ResNet, and Inception, and explains that these networks can be used as templates or starting points for one's own work.
- The ideas from these networks are cross-fertilizing and making their way into other disciplines.

Outline

- Classic networks:
- LeNet-5
- AlexNet
- VGG
- ResNet
- Inception

FPT Education

FPT UNIVERSITY

Case Studies

Classic networks

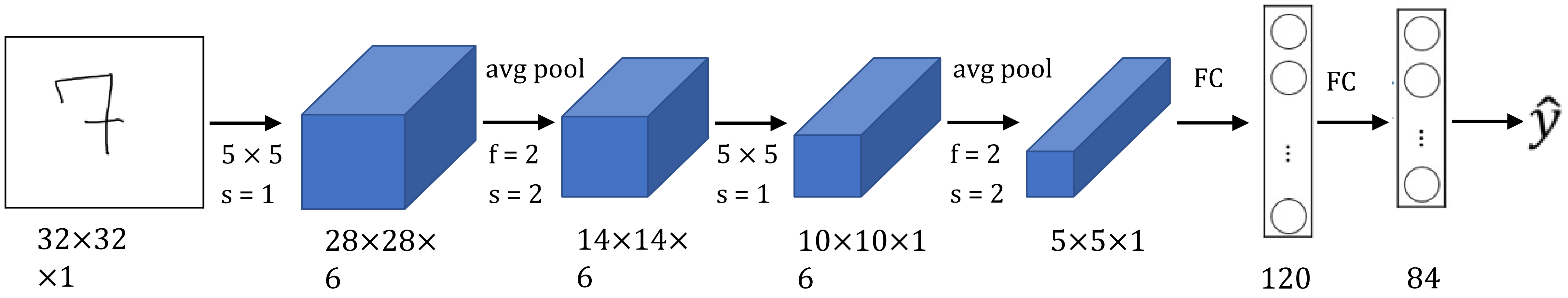
LeNet - 5

- Lenet-5 is one of the earliest pre-trained models proposed by Yann LeCun and others in the year 1998, in the research paper Gradient-Based Learning Applied to Document Recognition. They used this architecture for recognizing the handwritten and machine-printed characters.
- The main reason behind the popularity of this model was its simple and straightforward architecture. It is a multi-layer convolution neural network for image classification
- The architecture consists of a series of convolutional and pooling layers followed by fully connected layers and a final output layer. As you go deeper into the network, the height and width of the volumes decrease, while the number of channels increases.

LeNet - 5

- It takes a grayscale image as input.
- Once we pass it through a combination of convolution and pooling layers, the output will be passed through fully connected layers and classified into corresponding classes.
- The total number of parameters in LeNet-5 are:
  - Parameters: 60k
  - Layers flow: Conv -> Pool -> Conv -> Pool -> FC -> FC -> Output
  - Activation functions: Sigmoid/tanh and ReLu

LeNet - 5



Architecture of the Lenet-5

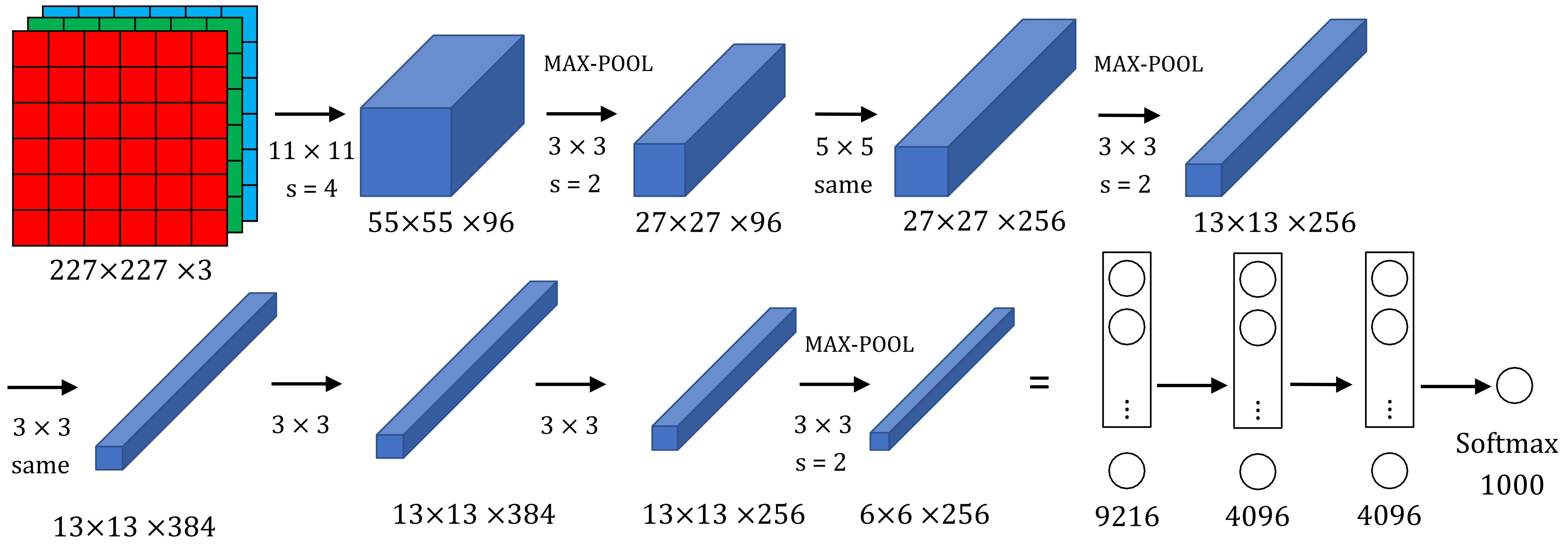
AlexNet

- Alexnet was proposed in 2012 in the research paper named Imagenet Classification with Deep Convolution Neural Network by Alex Krizhevsky and his colleagues.
- In this model, the depth of the network was increased in comparison to Lenet-5.
- It was designed for image classification tasks and had a similar architecture to LeNet-5, with a series of convolutional and pooling layers followed by fully connected layers and a final output layer:
  - Parameters: 60 million
  - Activation function: ReLu
- AlexNet used the value activation function, which was a major improvement over the sigmoid and tanh nonlinearities used at the time.

AlexNet

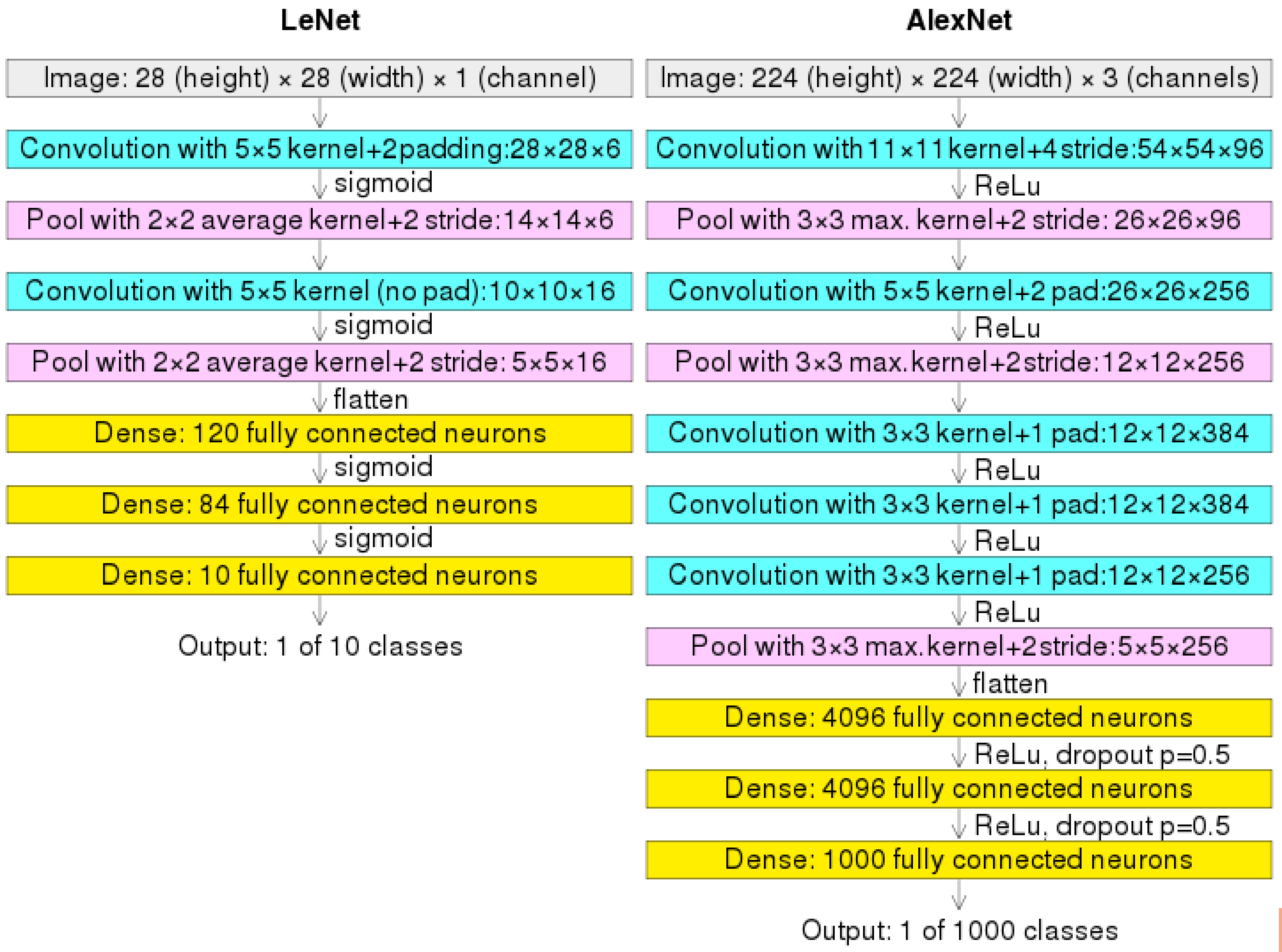
- AlexNet architecture:
  - It has 8 layers with learnable parameters.
  - The input to the Model is RGB images.
  - It has 5 convolution layers with a combination of max-pooling layers.
  - Then it has 3 fully connected layers.
  - The activation function used in all layers is Relu.
  - It used two Dropout layers.
  - The activation function used in the output layer is Softmax.
  - The total number of parameters in this architecture is 62.3 million.

AlexNet



Architecture of the AlexNet

AlexNet

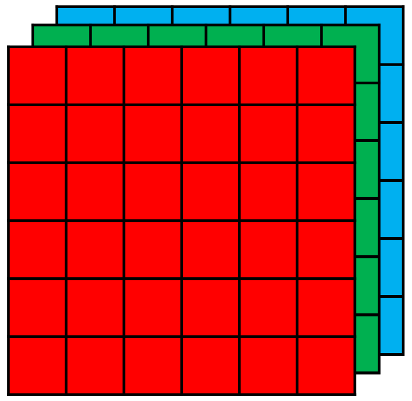

LeNet
Image: 28 (height) × 28 (width) × 1 (channel)
Convolution with 5×5 kernel+2padding:28×28×6
sigmoid
Pool with 2×2 average kernel+2 stride:14×14×6
Convolution with 5×5 kernel (no pad):10×10×16
sigmoid
Pool with 2×2 average kernel+2 stride: 5×5×16
flatten
Dense: 120 fully connected neurons
sigmoid
Dense: 84 fully connected neurons
sigmoid
Dense: 10 fully connected neurons
Output: 1 of 10 classes
AlexNet
Image: 224 (height) × 224 (width) × 3 (channels)
Convolution with 11×11 kernel+4 stride:54×54×96
ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
Convolution with 5×5 kernel+2 pad:26×26×256
ReLu
Pool with 3×3 max.kernel+2stride:12×12×256
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×256
ReLu
Pool with 3×3 max.kernel+2stride:5×5×256
flatten
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 1000 fully connected neurons
Output: 1 of 1000 classes

VGG - 16

- VGG -16 was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”.
- It was designed for the ImageNet Large Scale Visual Recognition Challenge, which is an annual competition to recognize objects in images from a thousand different categories.
- VGG-16 is a deep CNN architecture that consists of 16 layers, including 13 convolutional layers, 5 max pooling layers, and 3 fully connected layers. The input to the network is an RGB image of size 224x224.

VGG - 16

MAX-POOL = 2×2 , s = 2 CONV = 3×3 filter, s = 1, same



$$224\times224\times3 \xrightarrow[\text{[CONV 64]}\ \times2]{} 224\times224\times64 \xrightarrow[\text{POOL}]{} 112\times112\times64 \xrightarrow[\text{[CONV 128]}\ \times2]{} 112\times112\times128 \xrightarrow[\text{POOL}]{} 56\times56\times128$$

$$\xrightarrow[\text{[CONV 256]}\ \times3]{} 56\times56\times256 \xrightarrow[\text{POOL}]{} 28\times28\times256 \xrightarrow[\text{[CONV 512]}\ \times3]{} 28\times28\times512 \xrightarrow[\text{POOL}]{} 14\times14\times512$$

$$\xrightarrow[\text{[CONV 512]}\ \times3]{} 14\times14\times512 \xrightarrow[\text{POOL}]{} 7\times7\times512 \xrightarrow{} \text{FC } 4096 \xrightarrow{} \text{FC } 4096 \xrightarrow{} \text{Softmax } 1000$$

138m parameters

VGG - 16

- VGG-16 achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge, and its architecture has been widely adopted and used as a basis for many other CNNs.
- Its deep architecture and small filter sizes have been shown to be effective for extracting meaningful features from images, even in cases where the images are highly varied or complex.

FPT Education

FPT UNIVERSITY

Case Studies

Residual Networks (ResNets)

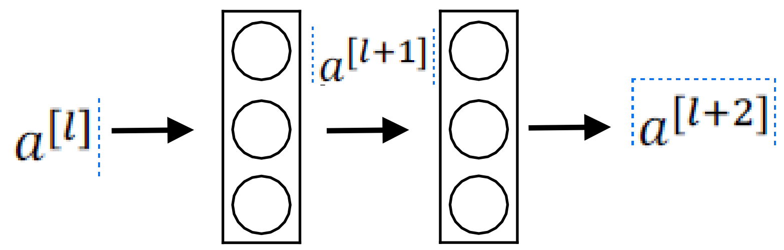
Residual Networks (ResNets)

- The ResNet architecture was proposed in 2015 by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, who were researchers at Microsoft at the time. They introduced the architecture in their paper titled "Deep Residual Learning for Image Recognition," which was published in the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
- ResNets were designed to address the problem of vanishing gradients in very deep CNNs.
- The key innovation in ResNets is the use of residual connections, which allow the network to learn residual functions rather than directly learning the underlying mapping. A residual connection is simply a shortcut connection that skips one or more layers and adds the output of the skipped layers to the output of the subsequent layers.

Residual block

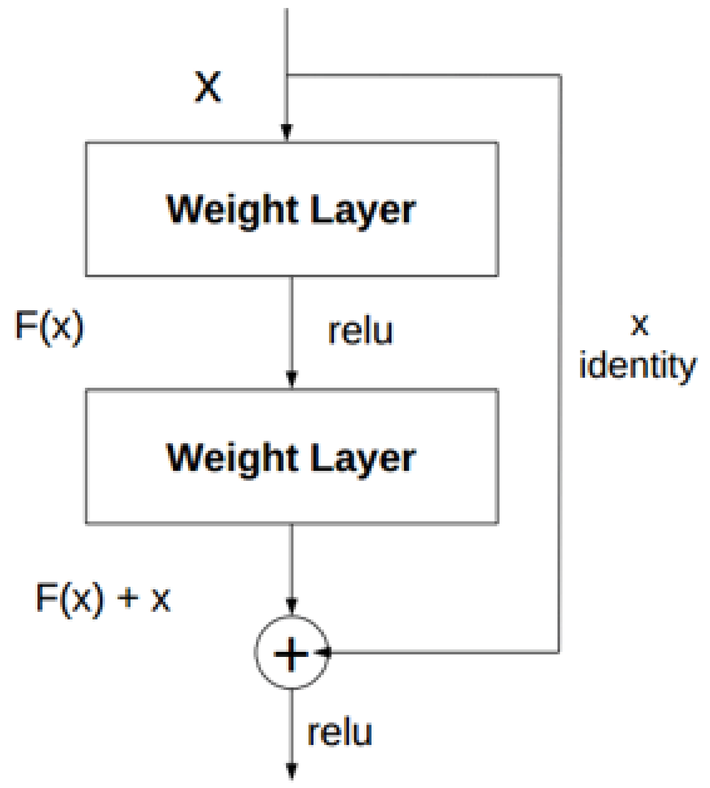
- The ResNet architecture consists of several blocks, each of which contains multiple residual units. A residual unit consists of two convolutional layers with batch normalization and ReLU activation, followed by a residual connection that adds the output of the first convolutional layer to the output of the second convolutional layer.
- Many residual blocks can be stacked together to form a deep network. When compared to plain networks, which do not have skip connections, ResNets enable the training of much deeper neural networks without any appreciable loss in performance. In plain networks, as the number of layers increases, the training error tends to decrease initially but then starts increasing.
- On the other hand, ResNets enable the performance of the training error to keep decreasing even as the number of layers gets deeper.

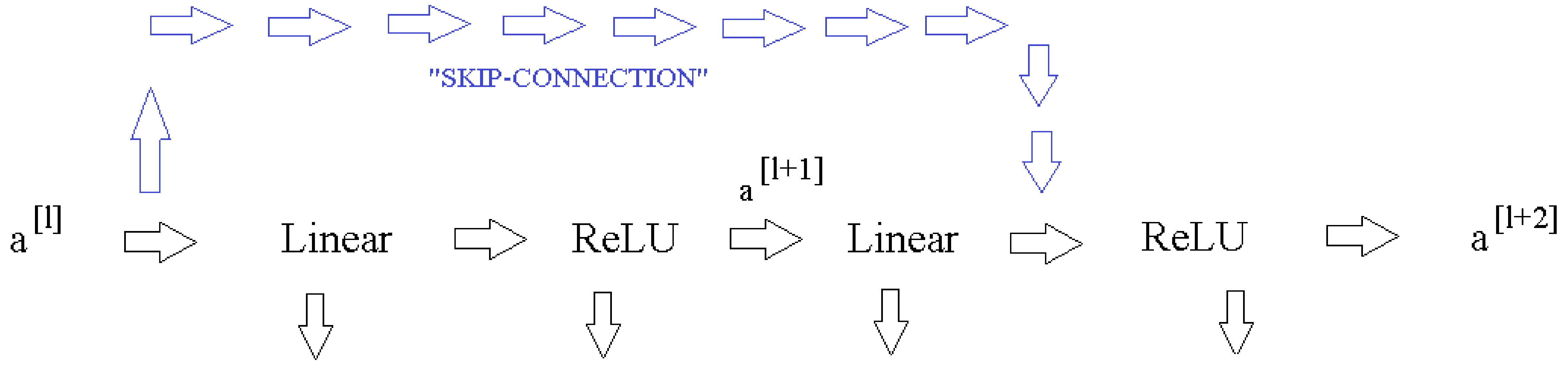
Residual block



$$\begin{tikzcd}
a^{[l]} \arrow[r] & \text{Linear} \arrow[r] \arrow[d] & \text{ReLU} \arrow[r, "a^{[l+1]}"] \arrow[d] & \text{Linear} \arrow[r] \arrow[d] & \text{ReLU} \arrow[r] \arrow[d] & a^{[l+2]} \\
 & z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]} & a^{[l+1]} = g(z^{[l+1]}) & z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} & a^{[l+2]} = g(z^{[l+2]}) &
\end{tikzcd}$$

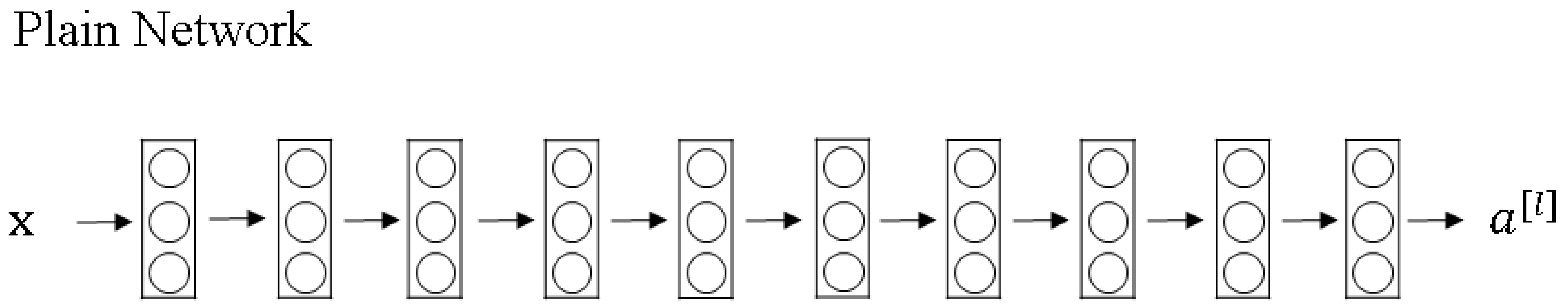
Residual block

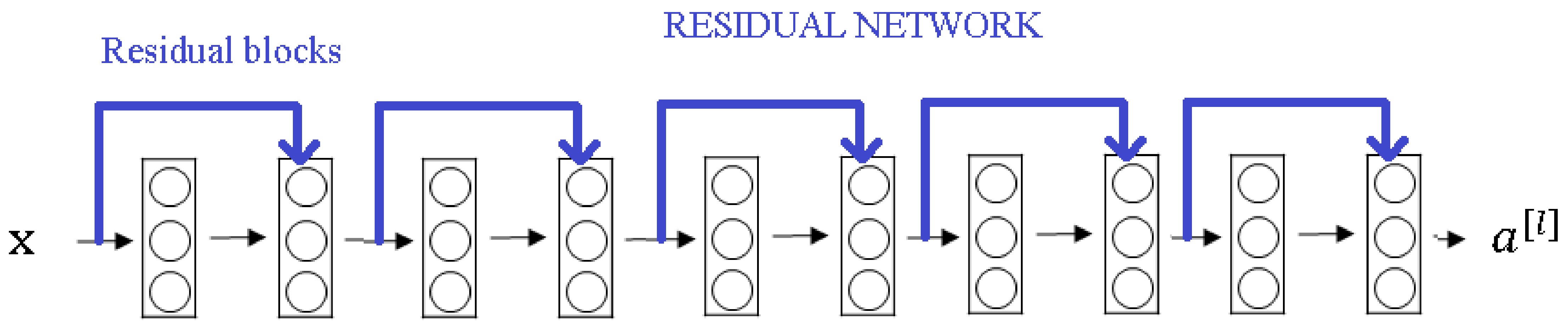




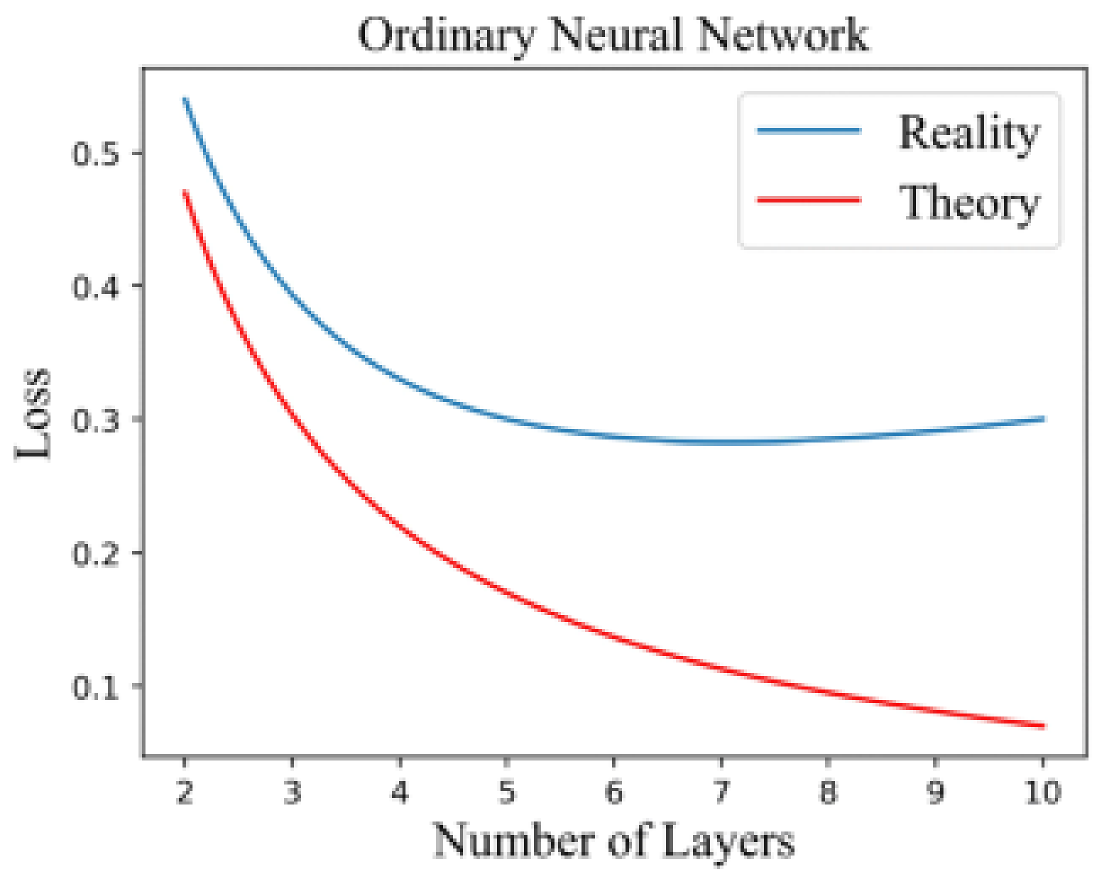
$$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]}) \qquad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

Residual block

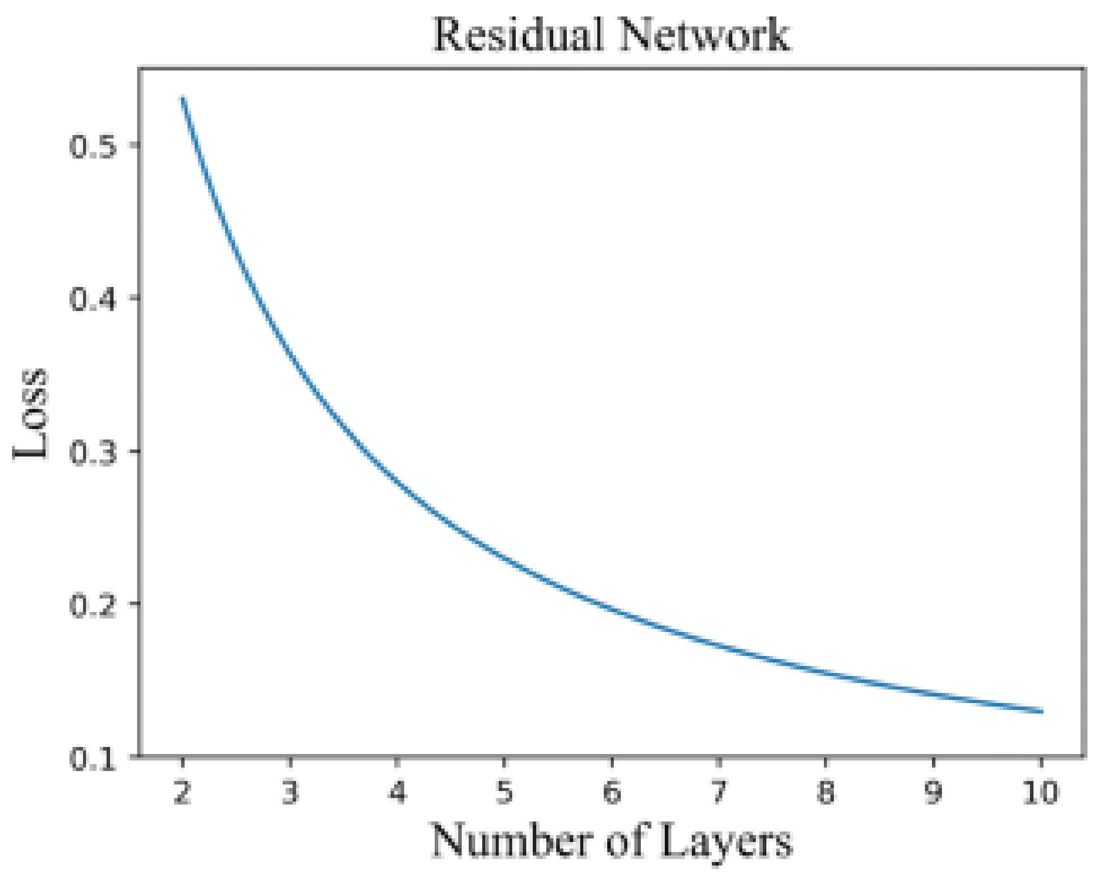




Residual Network



(a) Ordinary neural network



(b) Residual network

Residual Networks (ResNets)

- ResNets work so well because they help with the vanishing and exploding gradient problems in very deep neural networks.
- The ResNet architecture has been shown to achieve state-of-the-art performance on many computer vision tasks, including image classification, object detection, and semantic segmentation. It has also been used as a backbone for other architectures, such as Faster R-CNN and Mask R-CNN, which are used for object detection and instance segmentation tasks, respectively.
- Overall, ResNets have been an important development in the field of deep learning, allowing researchers to train deeper and more powerful neural networks for a wide range of computer vision tasks.
- Although ResNets are effective at training very deep networks, it is unclear whether they will keep helping as the number of layers increases further.

FPT Education

FPT UNIVERSITY

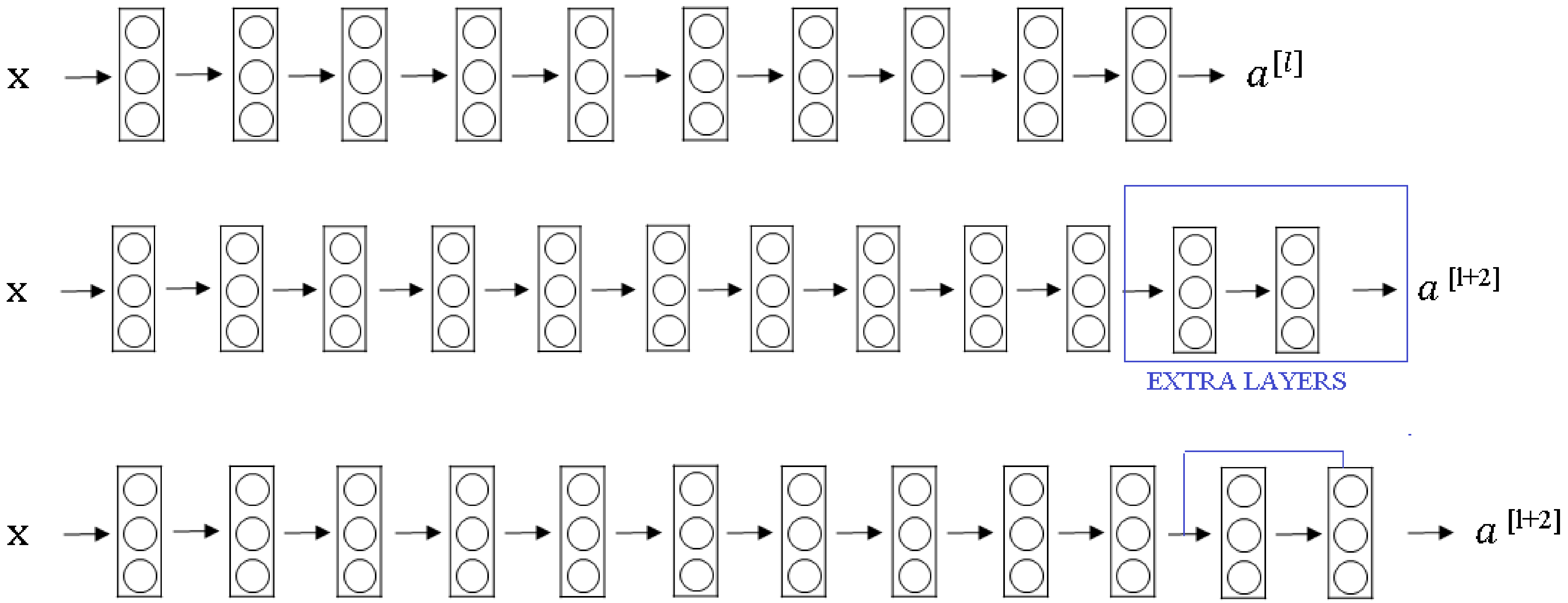
Case Studies

Why ResNets work

Why ResNets work

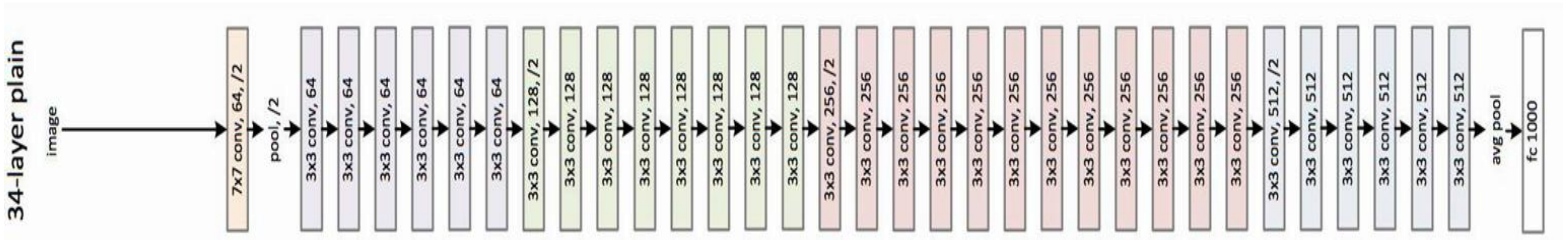
- Residual Networks, or ResNets, work so well because it is easy for the extra layers to learn the identity function, allowing the network to make use of more layers without hurting performance.
- The addition of a skip connection in a residual block allows the activation output to be equal to the activation input, which is necessary for the identity function to be learned. This addition also requires same convolutions to preserve the dimensionality of the output and input vectors. The ResNet architecture involves using multiple convolutional layers, occasional pooling layers, and a fully connected layer at the end for prediction. One key idea behind ResNets is using one by one convolutions, which will be discussed in the next section.

Why do residual networks work?

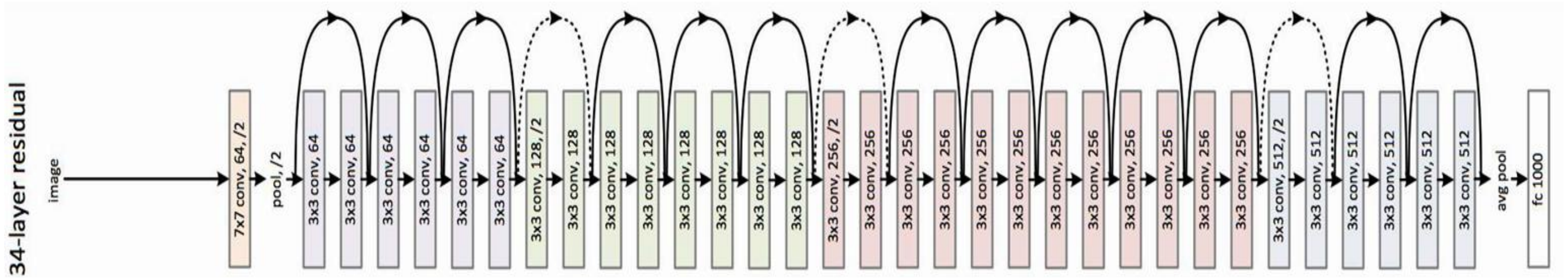


ResNet

Plain



ResNet



FPT Education

FPT UNIVERSITY

Case Studies

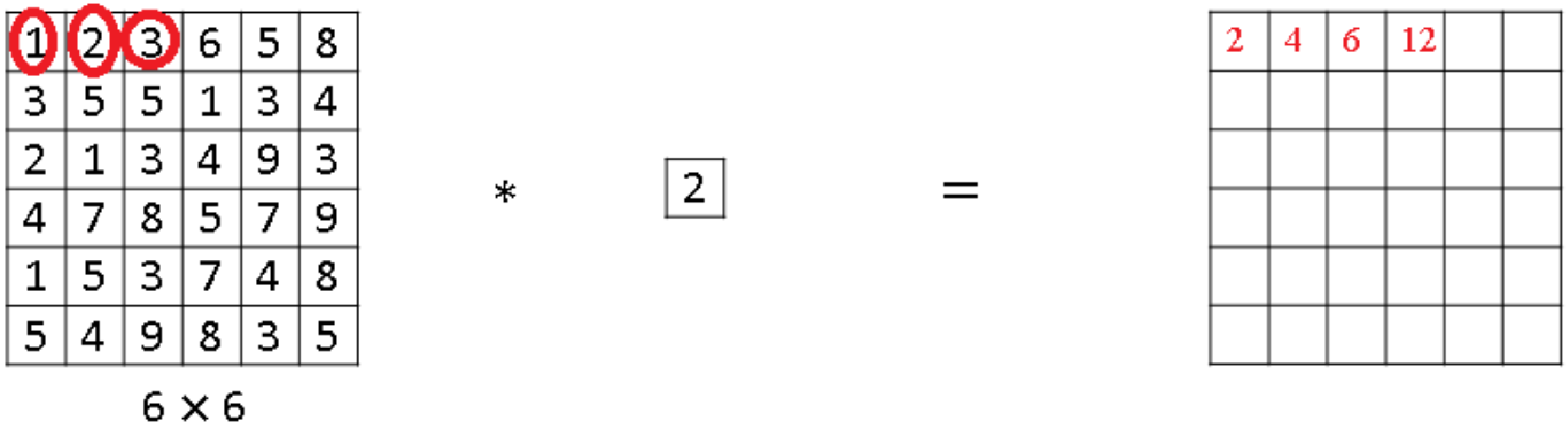
Network in Network and 1×1 convolutions

Why does a 1×1 convolution do?

- A one-by-one convolution is a type of convolutional layer that uses a filter with dimensions 1x1xN, where N is the number of channels in the input volume.
- While it may seem trivial, it applies a fully connected neural network to each position in the input volume, outputting a single value.
- This can perform a non-trivial computation and the output volume has the same height and width as the input volume but may have a different number of channels.

Why does a 1 × 1 convolution do?

- We have a 1x1 filter, we'll put in number two there, and if you take 6x6x1 input and convolve it with this filter, you end up just taking the image and multiplying it by two.



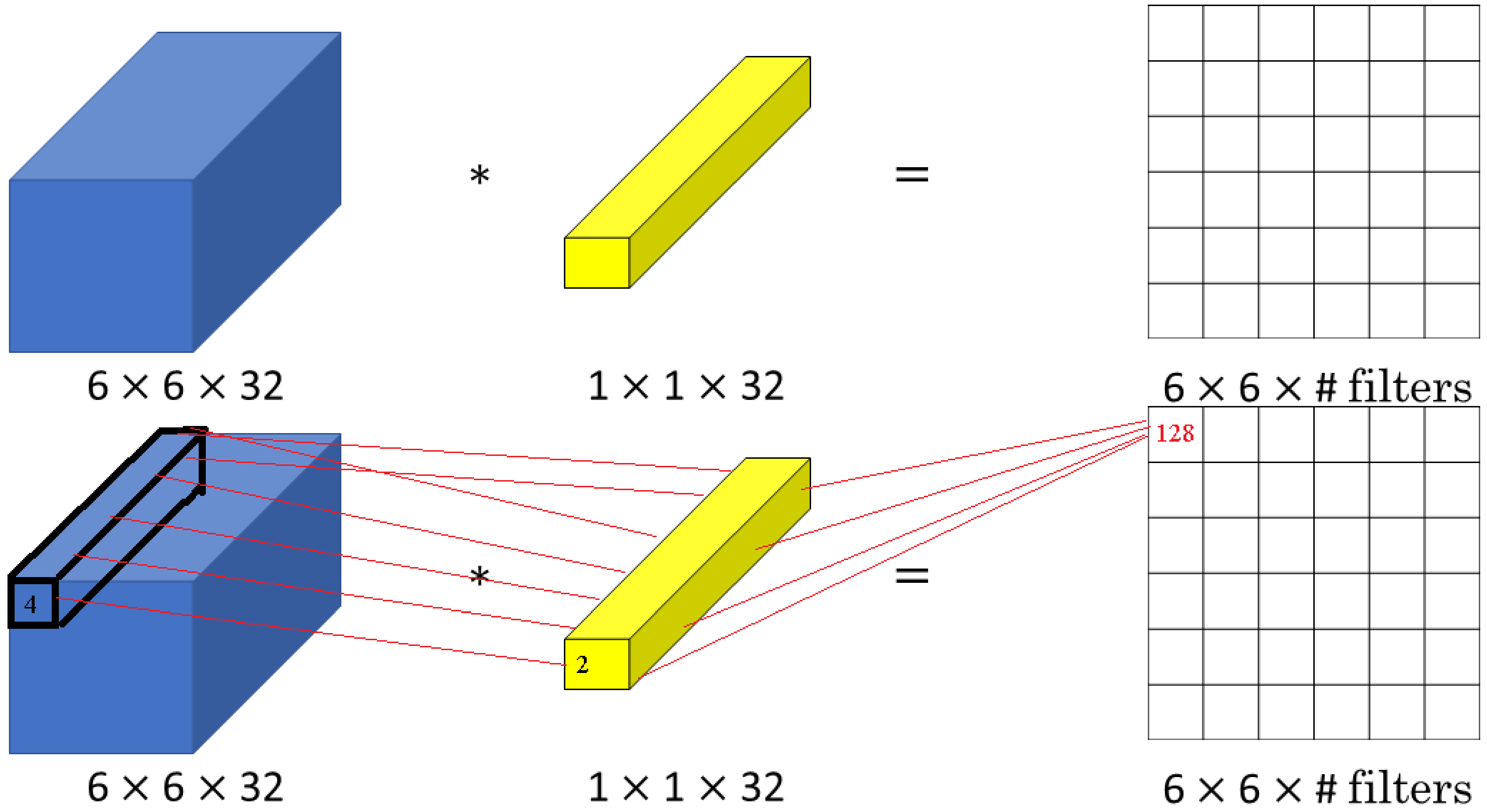
1	2	3	6	5	8
3	5	5	1	3	4
2	1	3	4	9	3
4	7	8	5	7	9
1	5	3	7	4	8
5	4	9	8	3	5

6 × 6

$*$ $\boxed{2}$ $=$

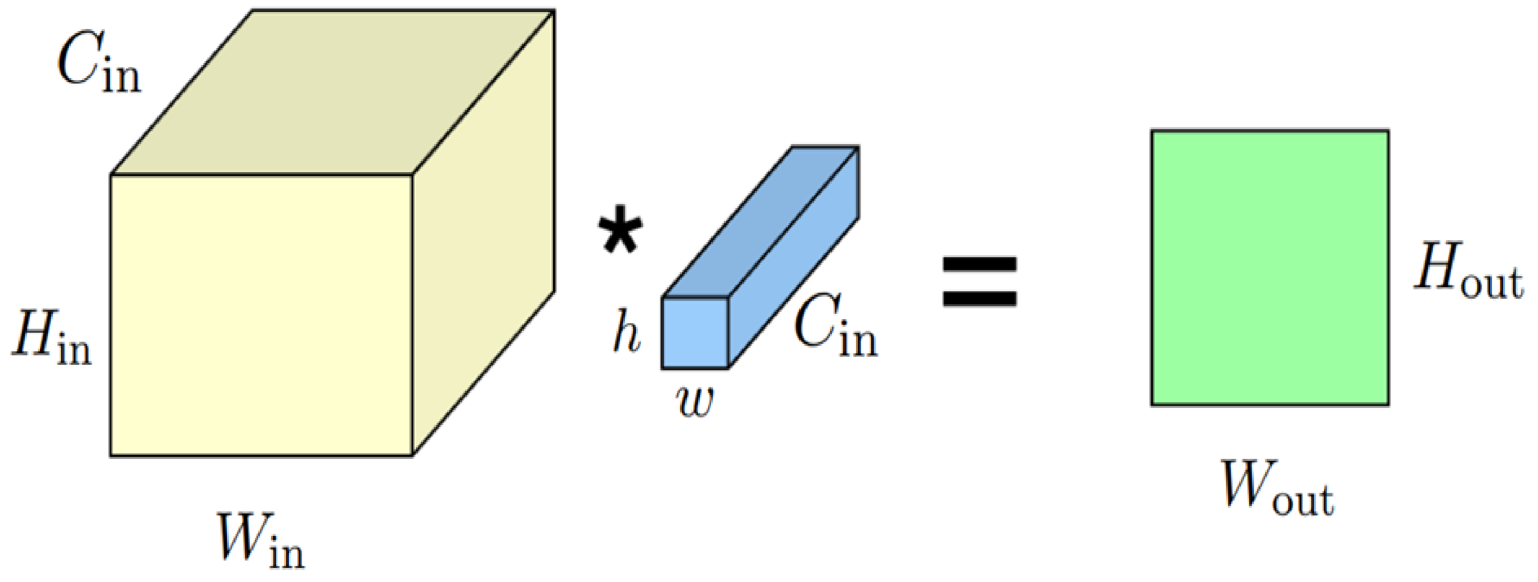
2	4	6	12		

Why does a 1 × 1 convolution do?



Why does a 1 × 1 convolution do?

height
x
width
x
channels



typically, h=w ~= 3

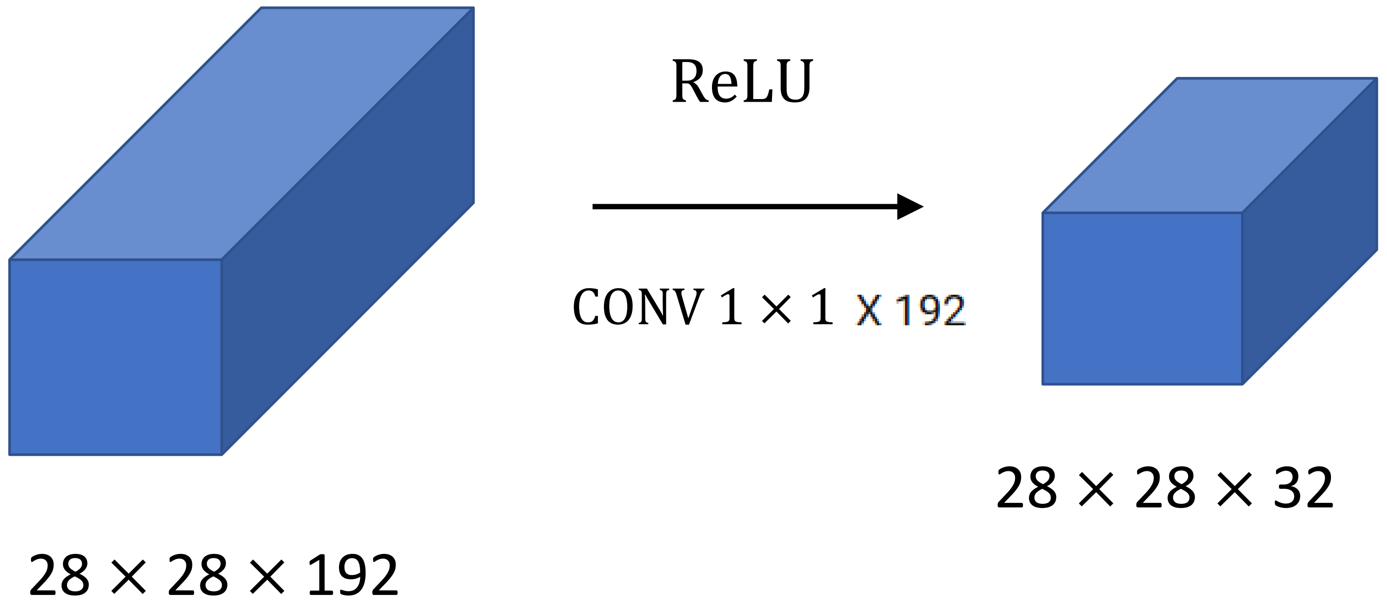
$x[i,j,k]$ $h[i,j,k]$ $y[i,j]$

convolution is done with no padding in the depth dimension, so at each "shift" a single output pixel is generated

Using 1×1 convolutions

- One way to use a one-by-one convolution is to shrink the number of channels in an input volume. For example, if you have a 28x28x192 volume and you want to shrink it to a 28x28x32 volume, you can use 32 one-by-one filters to compute the output volume.
- The one-by-one convolution operation has non-linearity, which allows it to learn a more complex function of the input volume by adding another layer. One-by-one convolutions have been used in many neural network architectures, including the inception network, which we'll see in the next video.

Using 1×1 convolutions



FPT Education

FPT UNIVERSITY

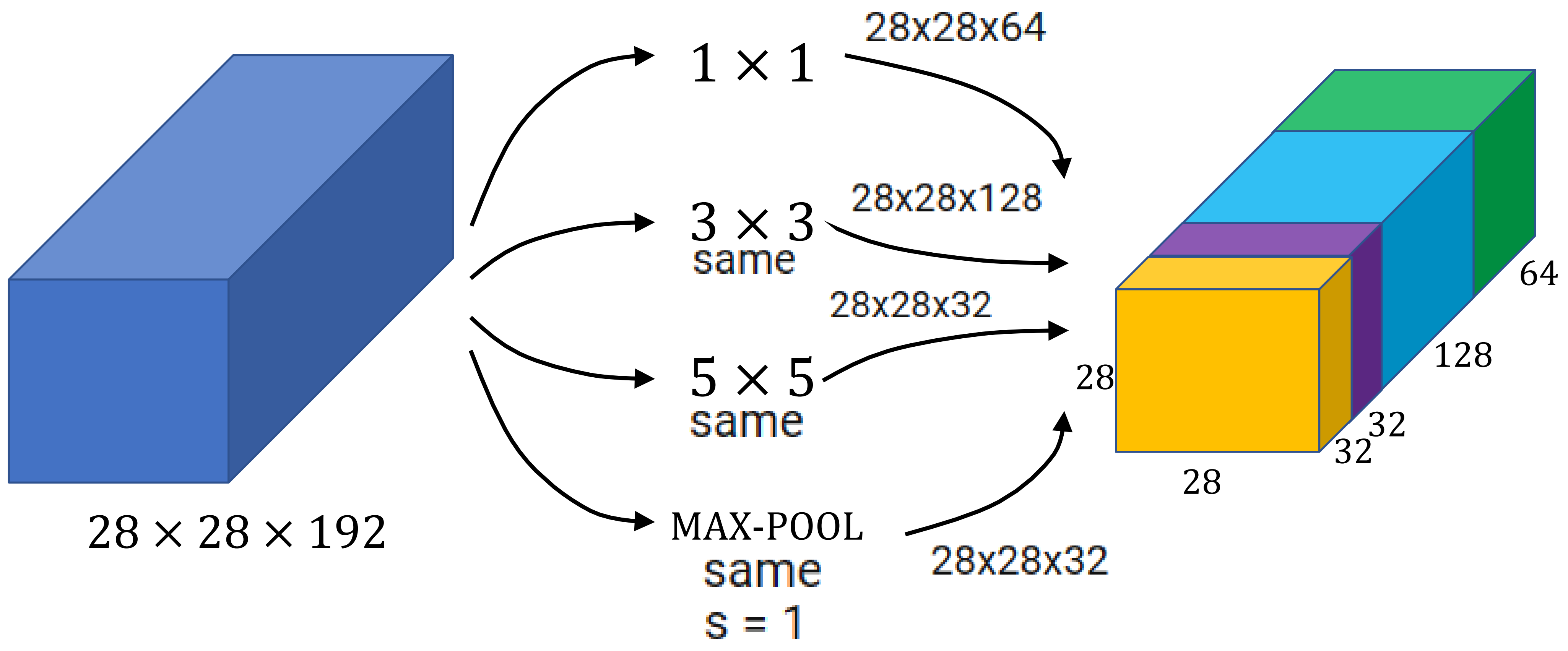
Case Studies

Inception network motivation

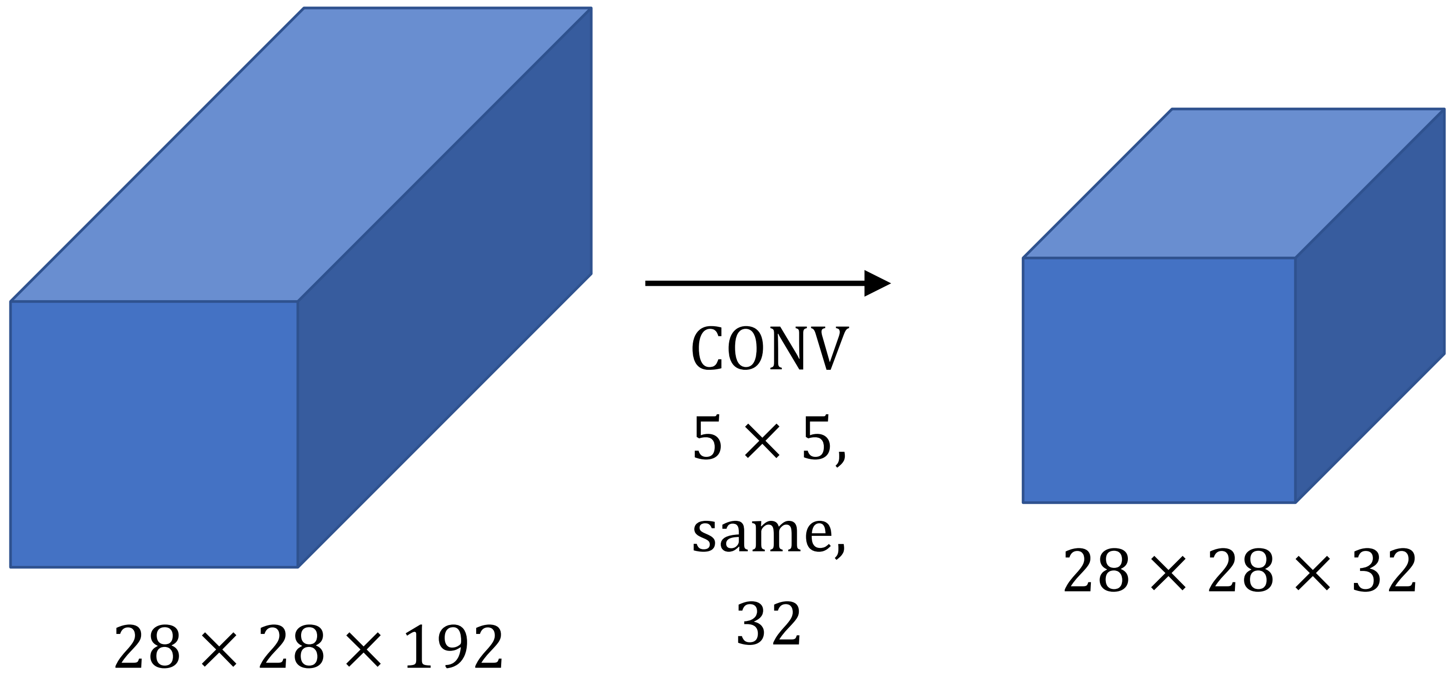
Motivation for inception network

- The Inception network is a convolutional neural network that uses multiple filter sizes and pooling layers in a single layer, allowing the network to learn the most effective combinations for each layer.
- The bottleneck layer is introduced to reduce the dimensionality of the input before applying a larger filter size, reducing computational cost while maintaining performance.
- It was proposed by Christian Szegedy at Google and has achieved state-of-the-art performance on image recognition tasks, influencing many other CNN architectures.

Motivation for inception network

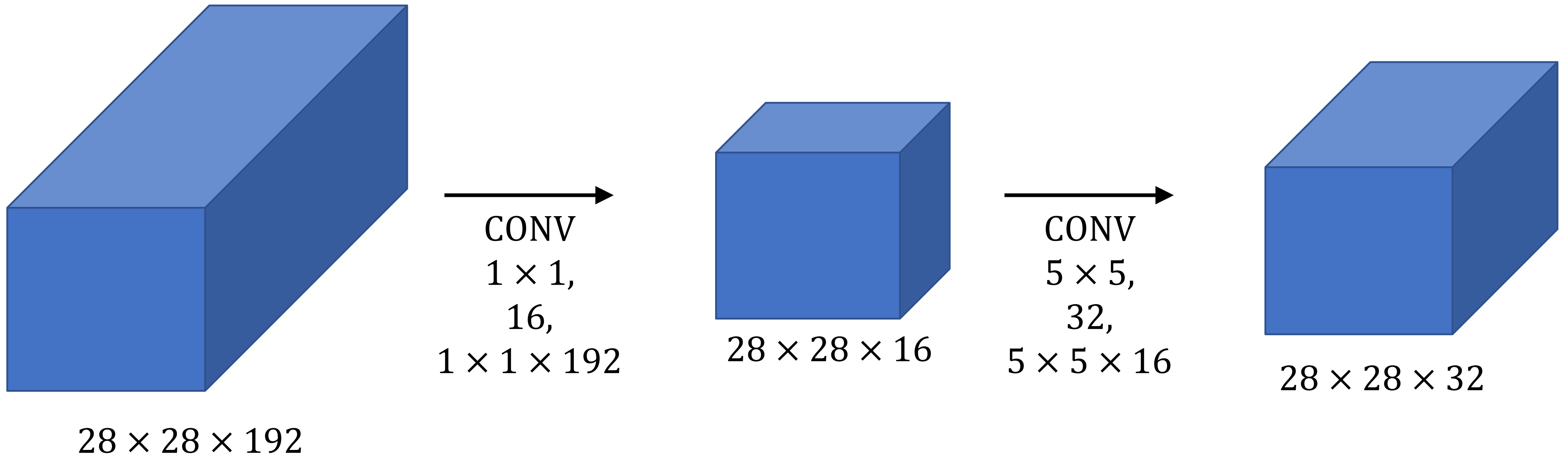


The problem of computational cost



28 x 28 x32 x 5 x 5 x 192 = 120M

Using 1×1 convolution



$28 \times 28 \times 192$
CONV 1×1, 16, $1 \times 1 \times 192$
$28 \times 28 \times 16$
CONV 5×5, 32, $5 \times 5 \times 16$
$28 \times 28 \times 32$

FPT Education

FPT UNIVERSITY

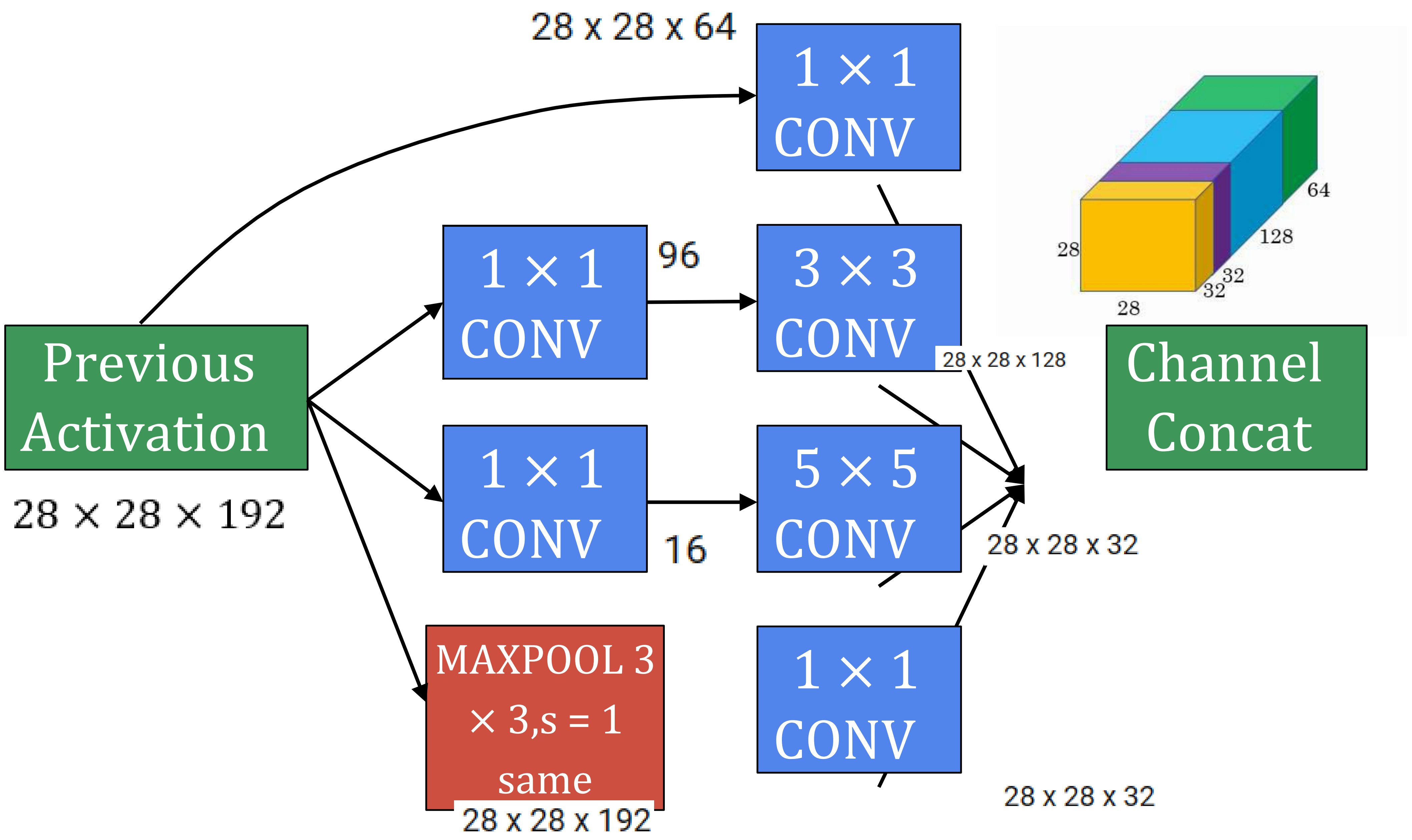
Case Studies

Inception network

Inception module

- The Inception module is a building block used in the Inception network, a convolutional neural network architecture designed for image recognition tasks.
- The module takes as input the activation output from some previous layer and applies a combination of 1x1, 3x3, and 5x5 convolutions, as well as pooling layers, to the input.
- The output of each convolutional layer is then concatenated along the channel dimension to form the output of the Inception module. The Inception module allows the network to learn the most effective combinations of filter sizes and pooling layers for each layer, and has been shown to achieve state-of-the-art performance on image recognition tasks.

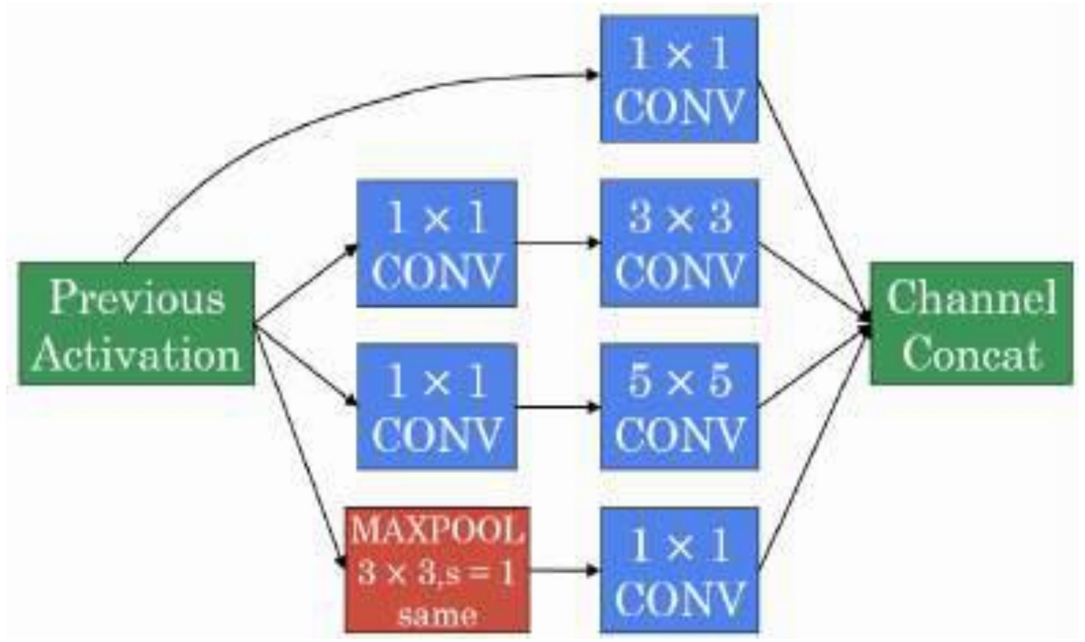
Inception module

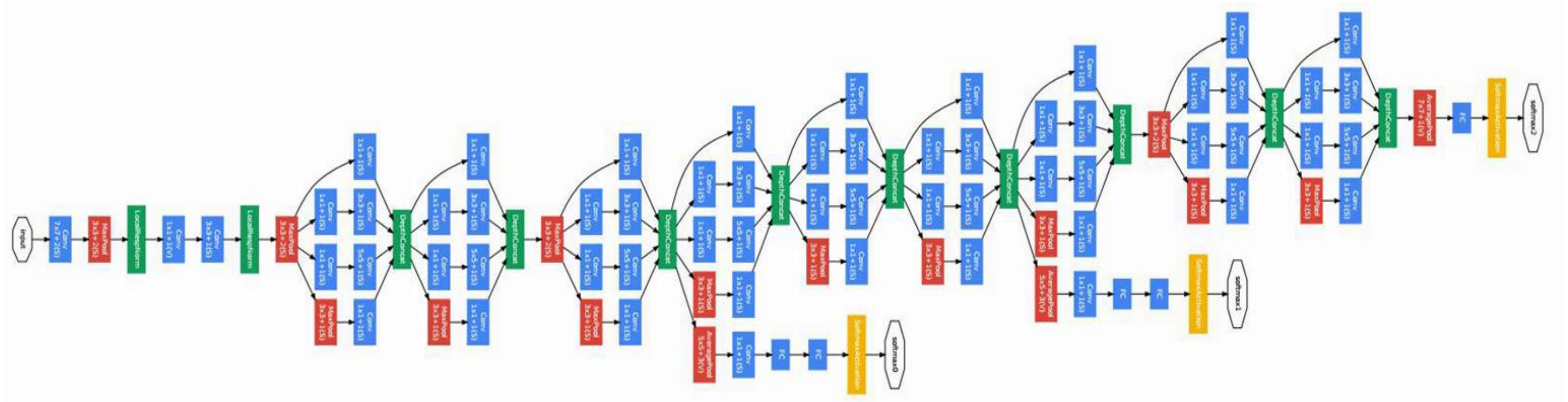


Inception network

- The Inception network was developed by Google and was named after a meme from the movie Inception. There are newer versions of the Inception algorithm, such as Inception v2, v3, and v4, and these are built on the same basic idea of the Inception module.
- The Inception network is composed of a lot of repeated blocks, which are basically the inception modules repeated in different positions of the network.
- The last few layers of the network consist of a fully connected layer followed by a softmax layer to make a prediction. The additional side-branches take some hidden layer and use it to make a prediction, which helps ensure that the features computed in the intermediate layers are not too bad for predicting the output. This has a regularizing effect on the Inception network and helps prevent it from overfitting.

Inception network





WE NEED TO GO
DEEPER

FPT Education

FPT UNIVERSITY

Convolutional Neural Networks

MobileNet

Motivation for MobileNets

- MobileNet is a convolutional neural network architecture designed for efficient processing on mobile devices and embedded systems. It was developed by Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam from Google.

- MobileNet uses depthwise separable convolutions, which split a standard convolution into two separate operations: a depthwise convolution that applies a single filter to each input channel, and a pointwise convolution that applies a 1x1 filter to the output of the depthwise convolution to combine the channels. This reduces the number of computations required and results in a more efficient network with fewer parameters, while still achieving good accuracy on image recognition tasks.

- MobileNet has been used in a variety of applications, including object detection, face recognition, and image segmentation, and has become a popular choice for computer vision tasks on mobile devices and other resource-constrained environments.

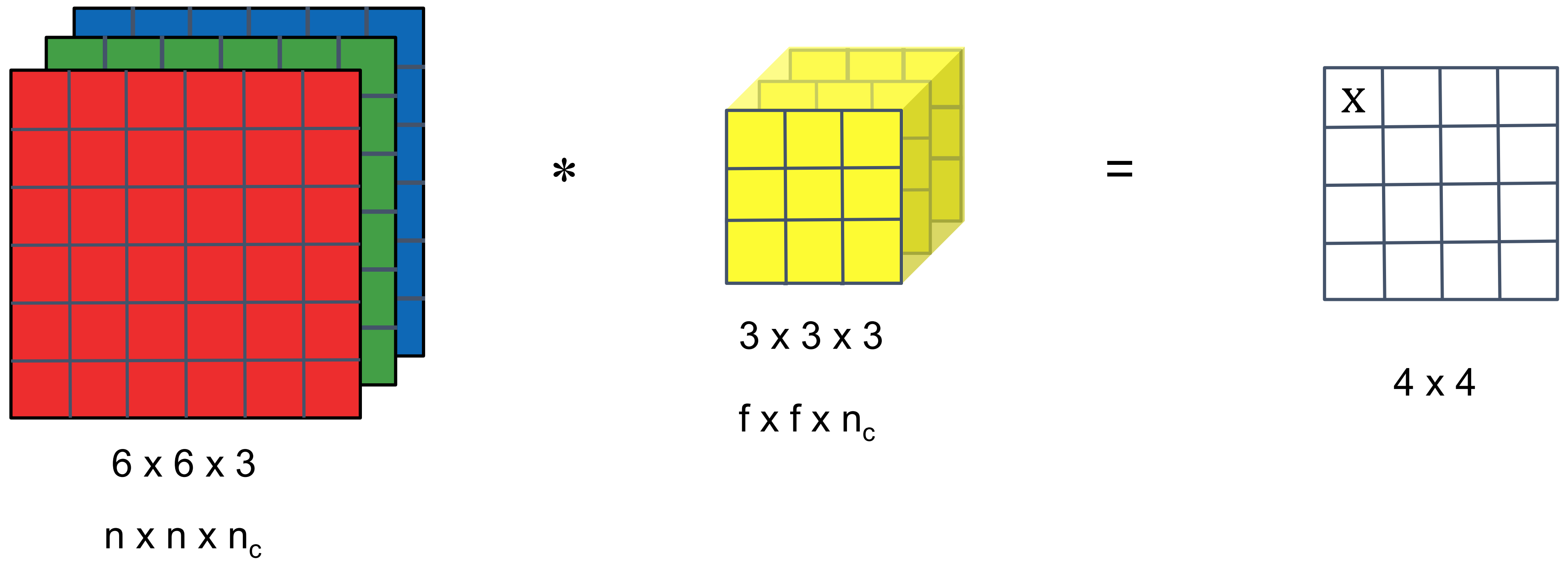
Motivation for MobileNets

- Low computational cost at deployment
- Useful for mobile and embedded vision applications
- Key idea: Normal vs. depthwise- separable convolutio

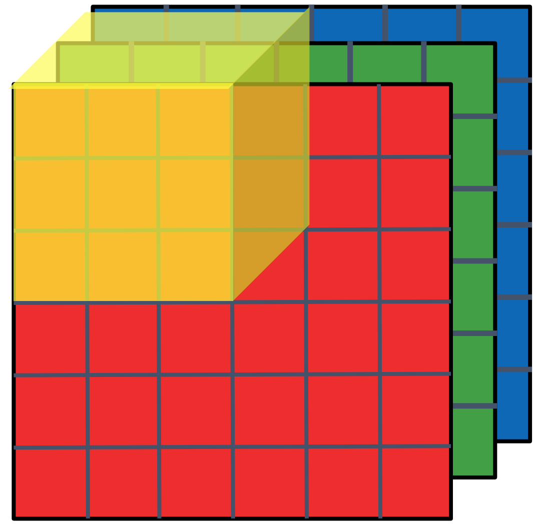
Normal Convolution

- Where an input image is convolved with a filter to produce an output feature map. The computational cost of this operation is proportional to the number of filter parameters, the number of filter positions, and the number of filters.

Normal Convolution



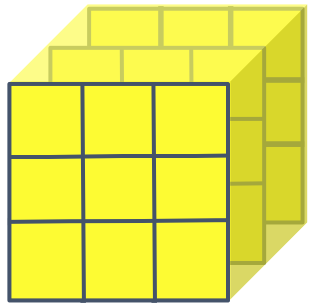
Normal Convolution



6 x 6 x 3

$n \times n \times n_c$

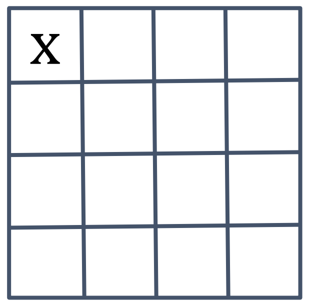
$*$



3 x 3 x 3

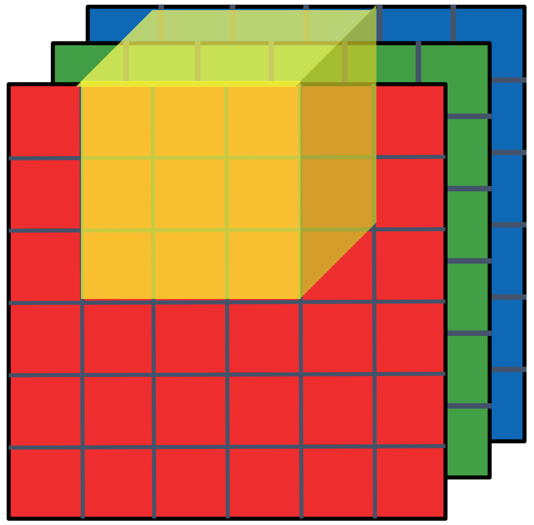
$f \times f \times n_c$

$=$



4 x 4

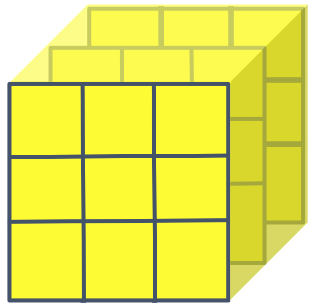
Normal Convolution



6 x 6 x 3

n x n x n_c

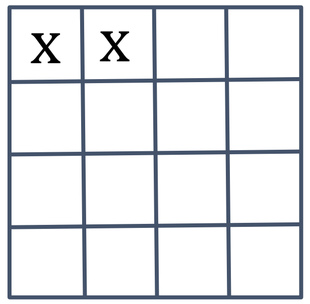
*



3 x 3 x 3

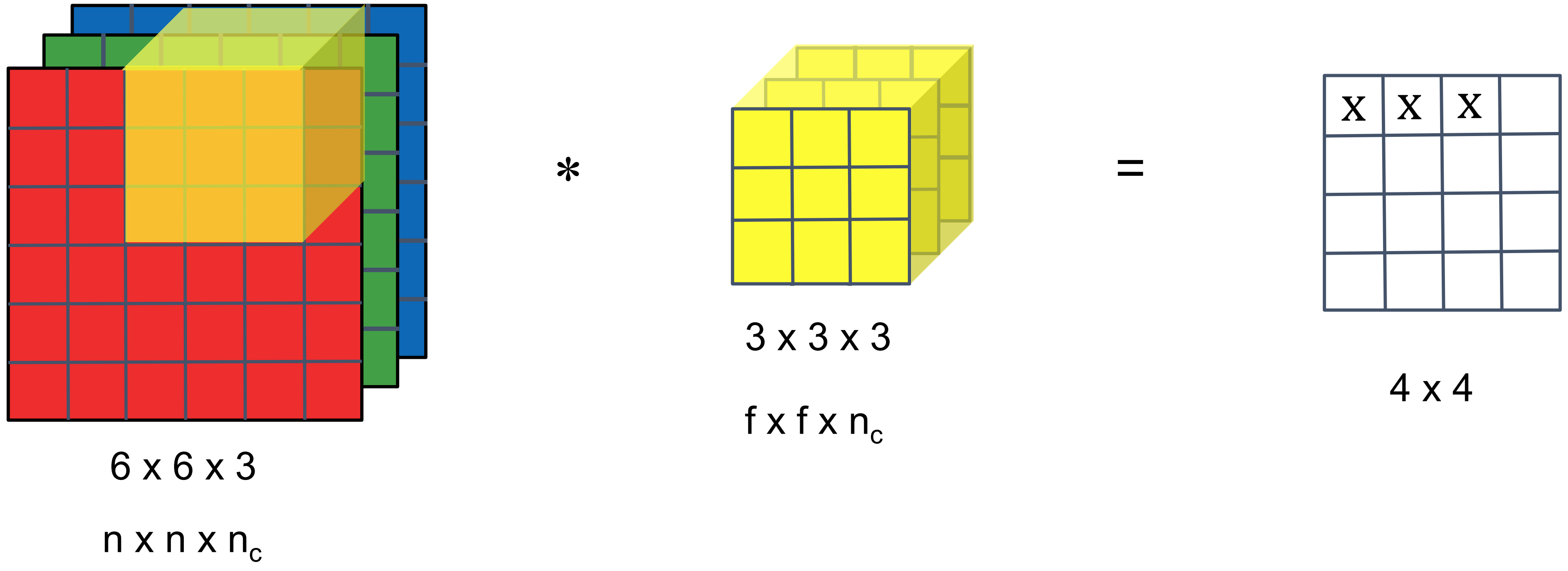
f x f x n_c

=

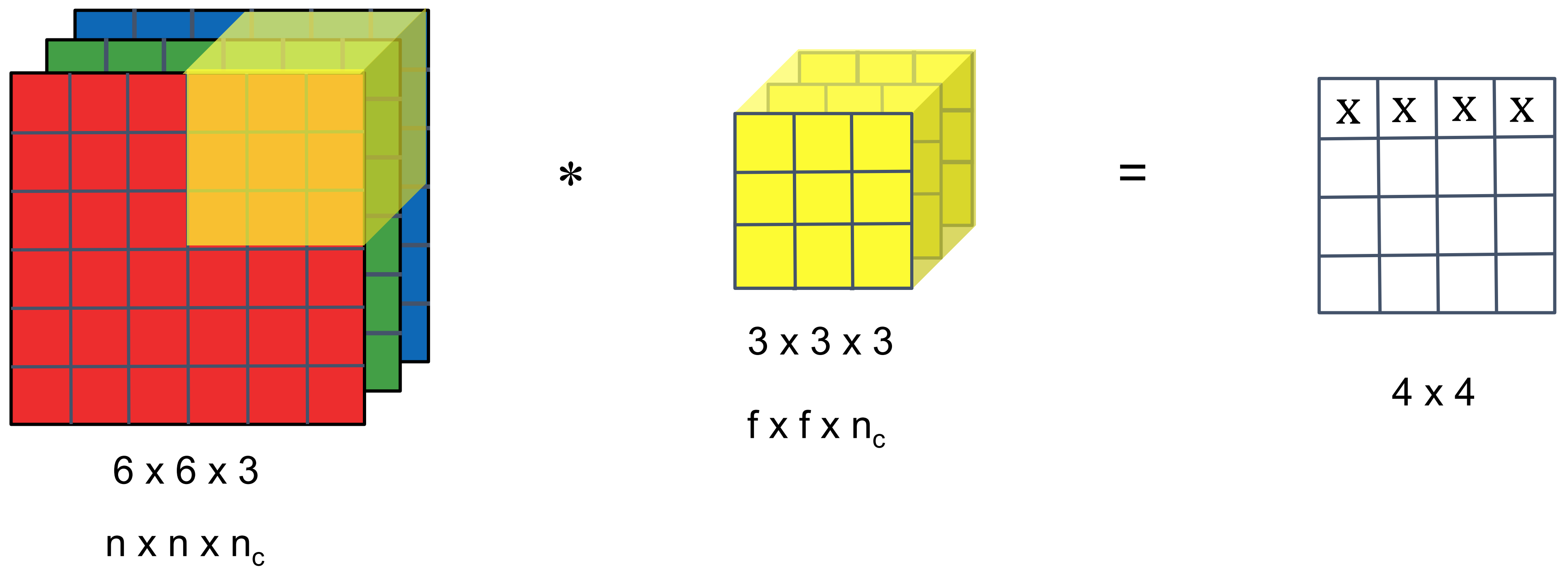


4 x 4

Normal Convolution



Normal Convolution



6 x 6 x 3

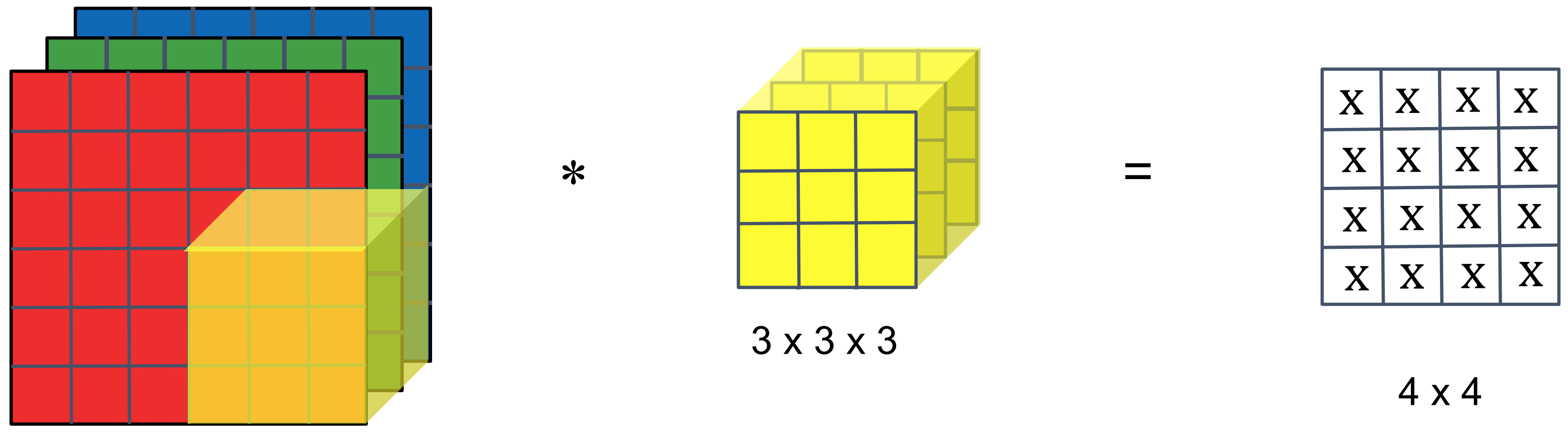
n x n x n_c

3 x 3 x 3

f x f x n_c

4 x 4

Normal Convolution



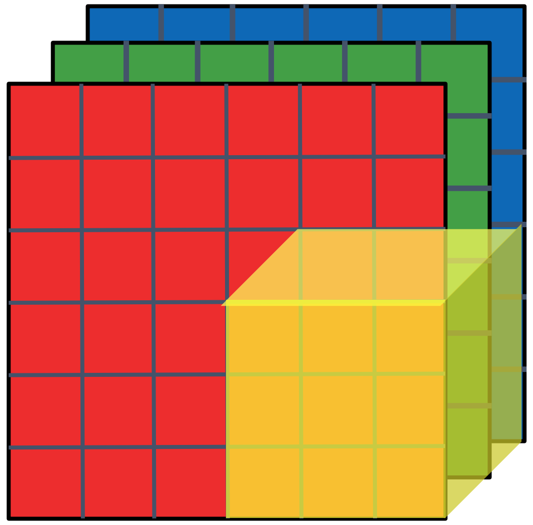
6 x 6 x 3

$n \times n \times n_c$

3 x 3 x 3

4 x 4

Normal Convolution

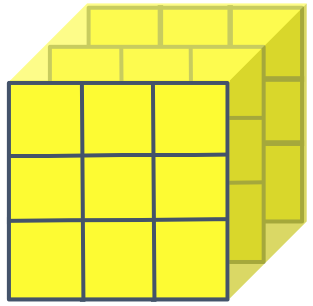


6 x 6 x 3

n x n x n_c

*

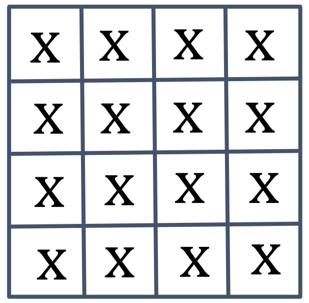
n_c' filter
5



3 x 3 x 3

f x f x n_c

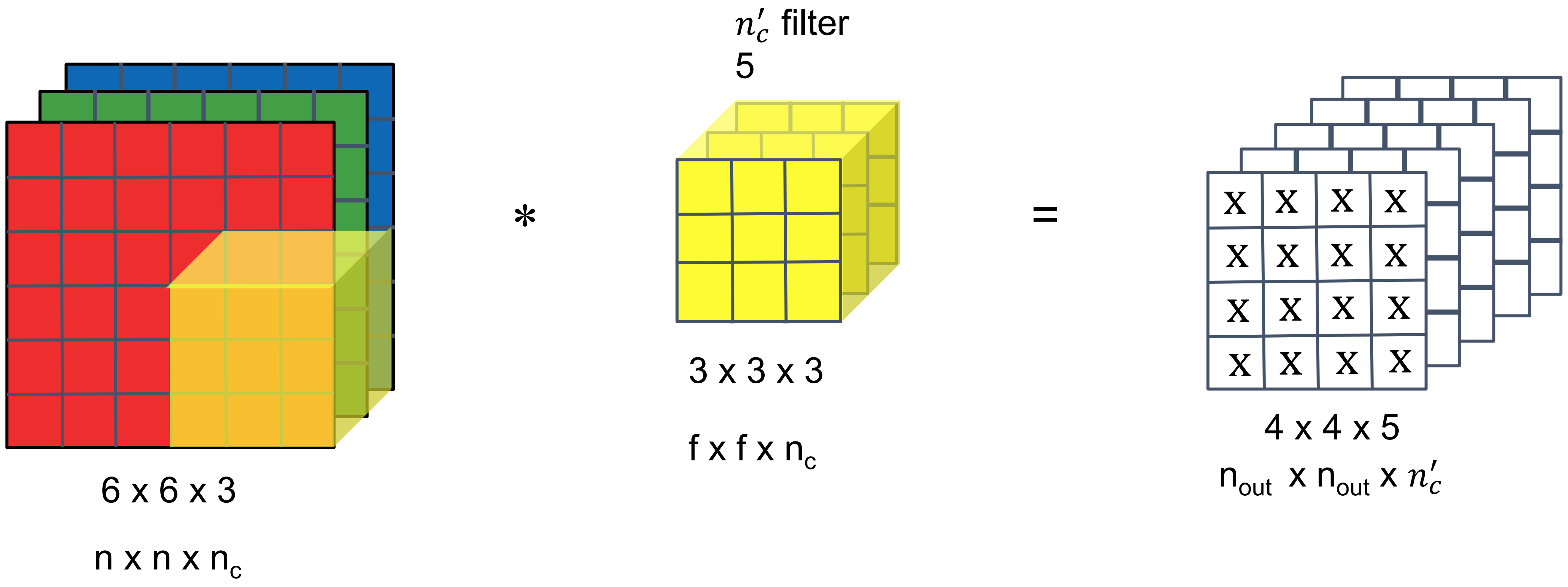
=



4 x 4

n_{out} x n_{out}

Normal Convolution



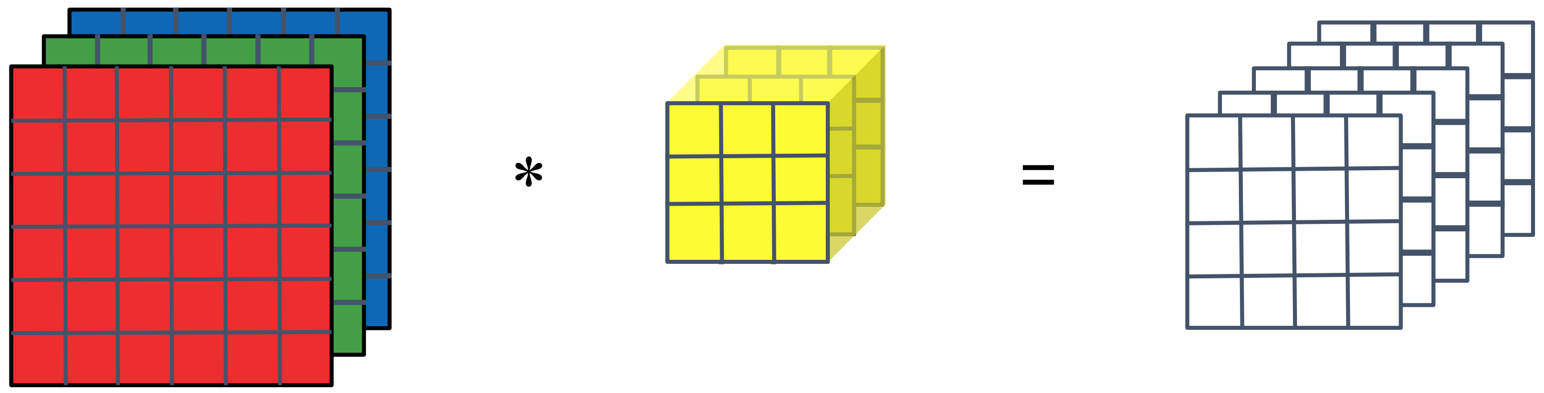
Computational cost = #filter params x # filter positions x # of filters

2160 **3 x 3 x 3** **4 x 4** **x** **5**

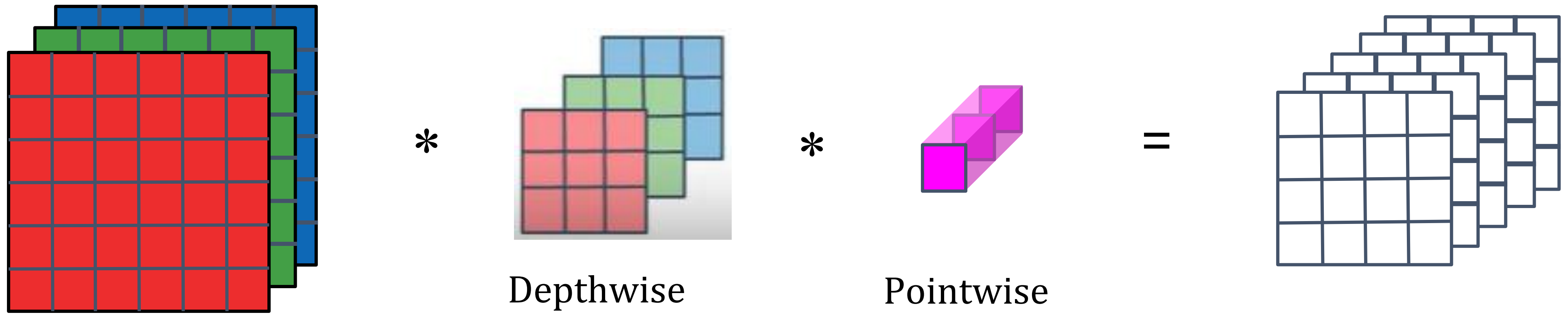
Depthwise Separable Convolution

- The depthwise separable convolution can be used as a building block for the MobileNet architecture, which can achieve significant computational efficiency while maintaining high accuracy in image classification tasks.

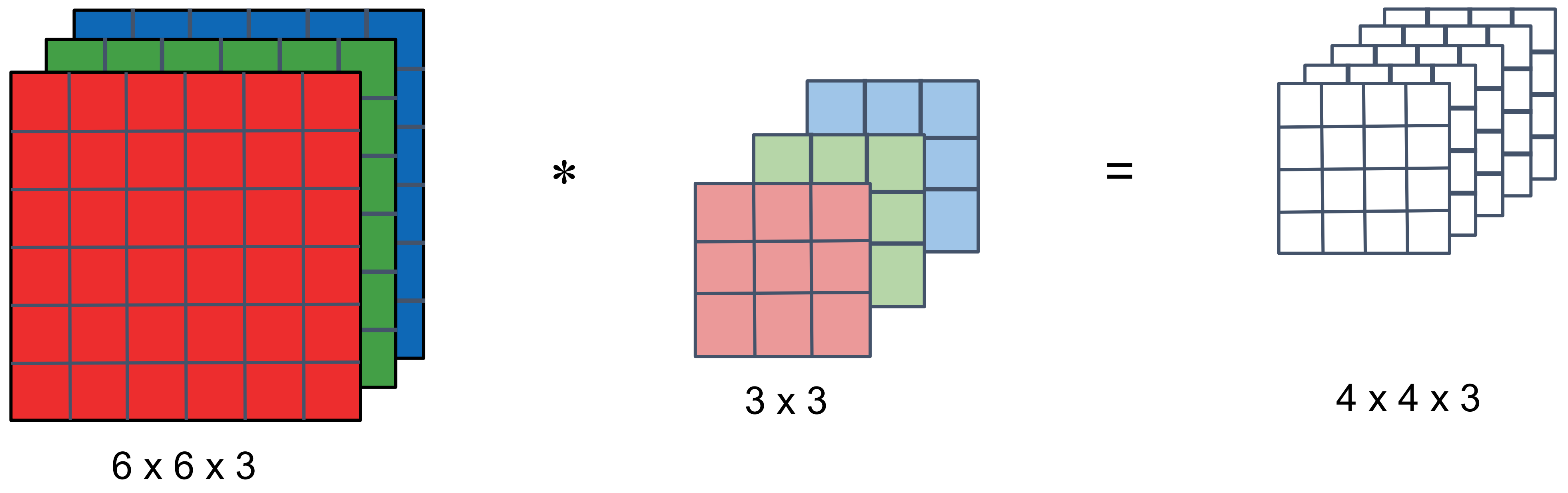
Depthwise Separable Convolution



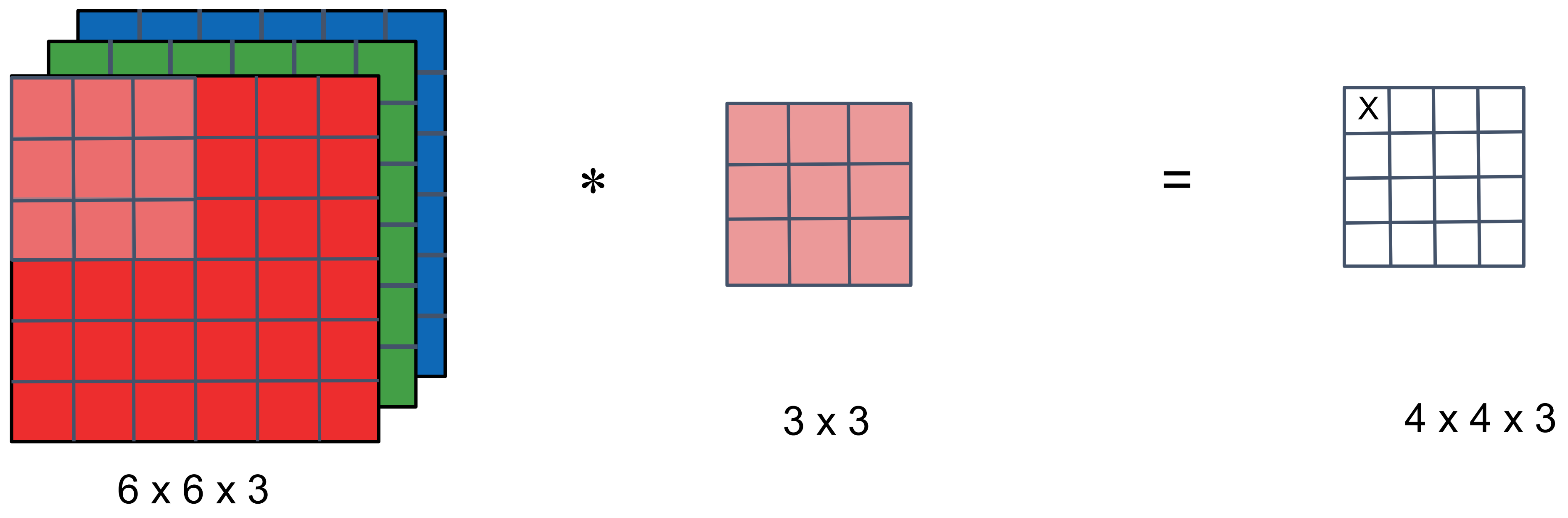
Depthwise Separable Convolution



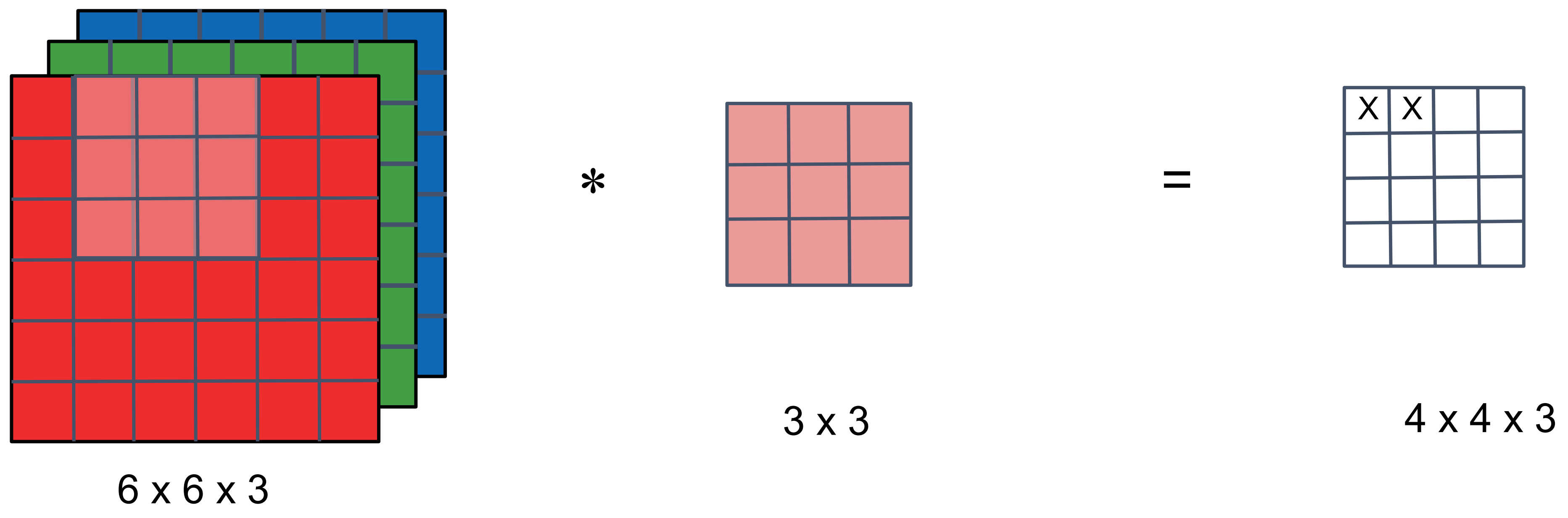
Depthwise Convolution



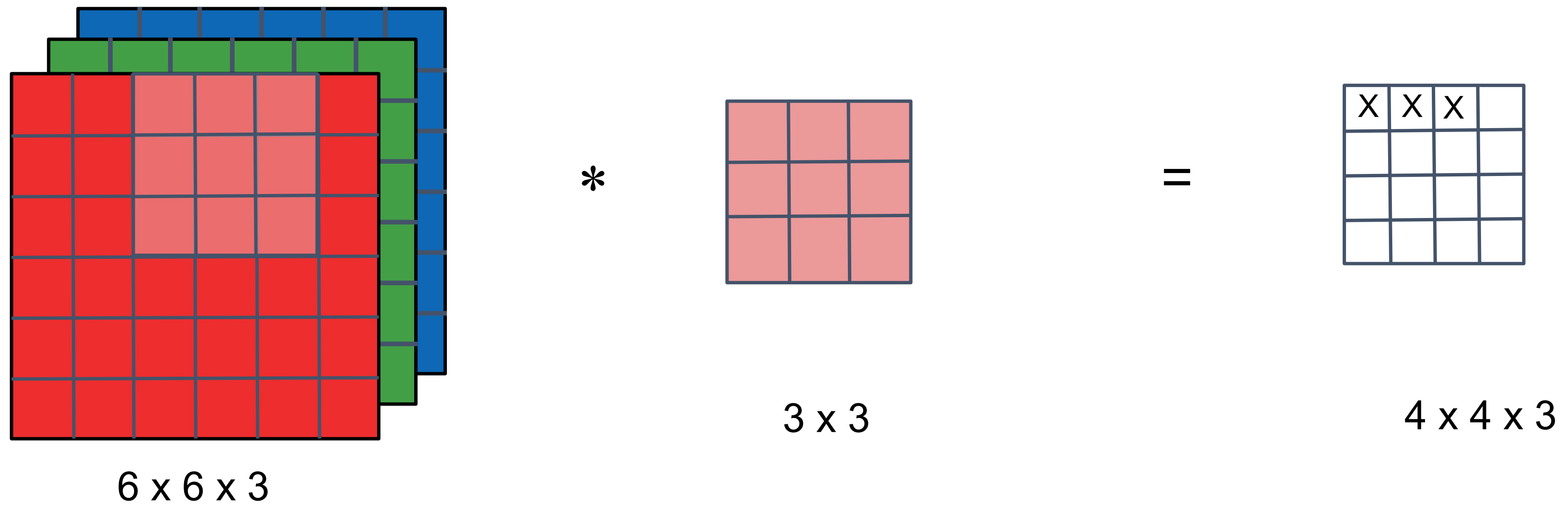
Depthwise Convolution



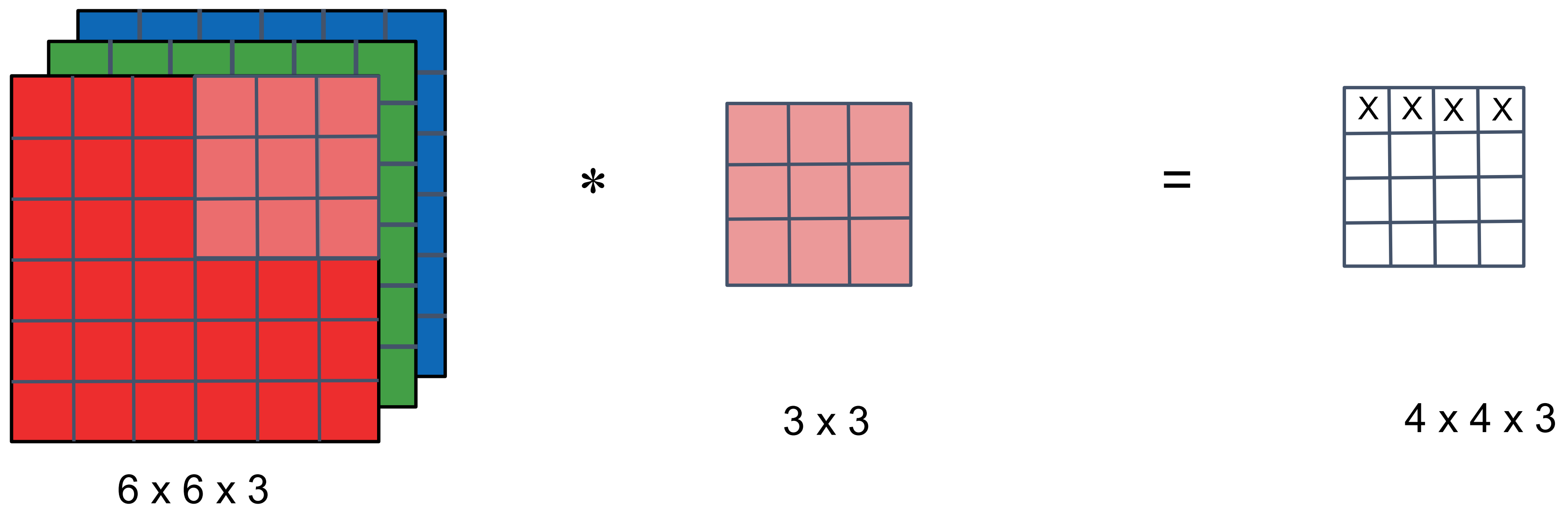
Depthwise Convolution



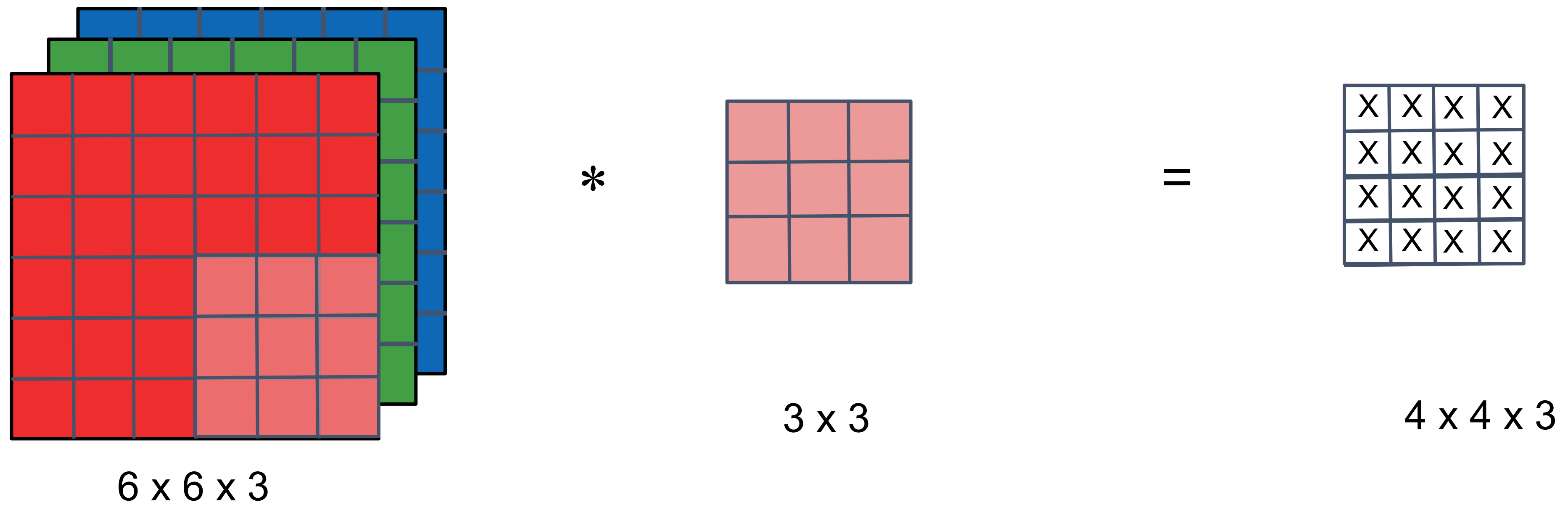
Depthwise Convolution



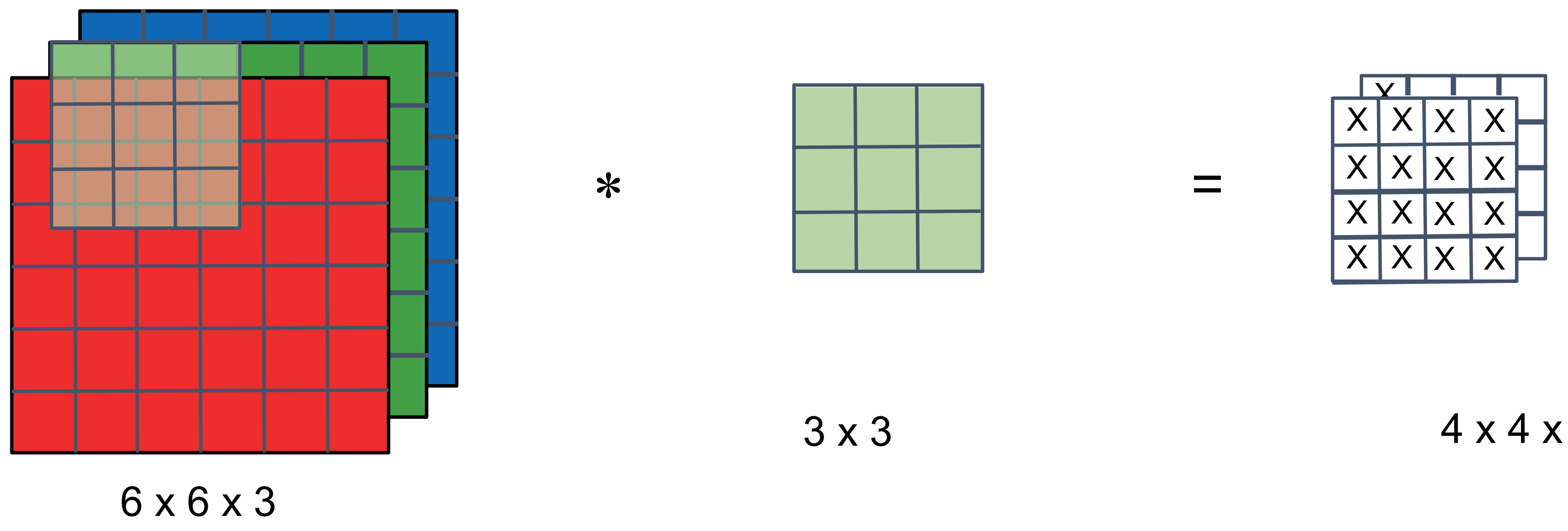
Depthwise Convolution



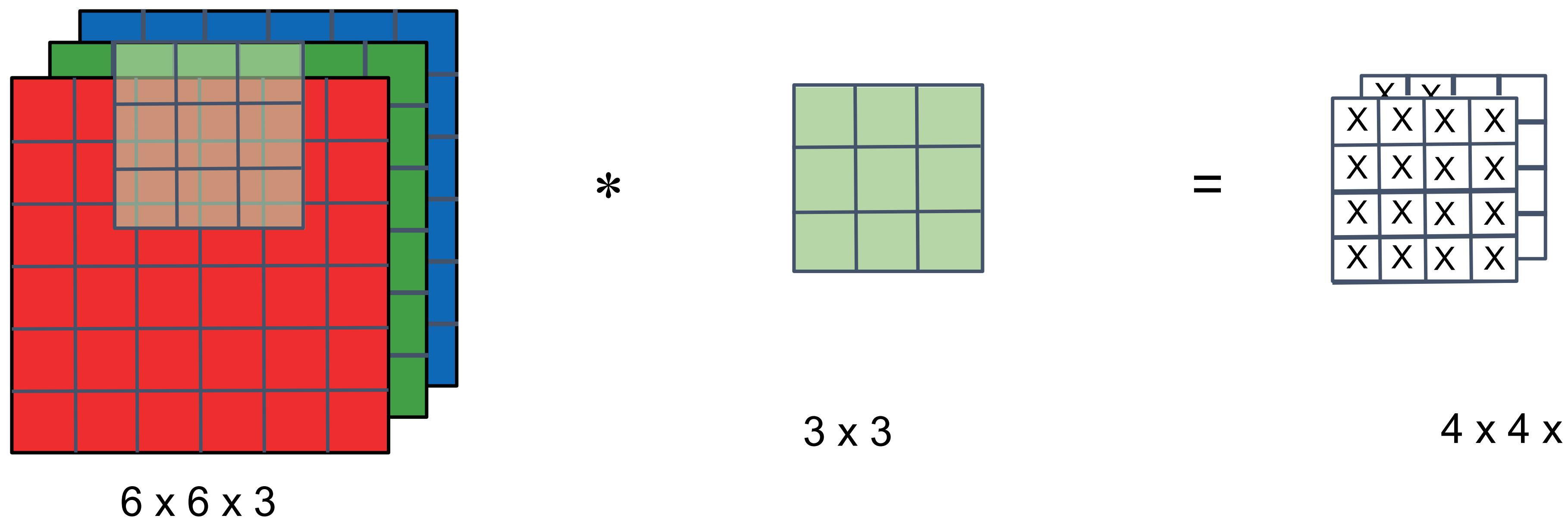
Depthwise Convolution



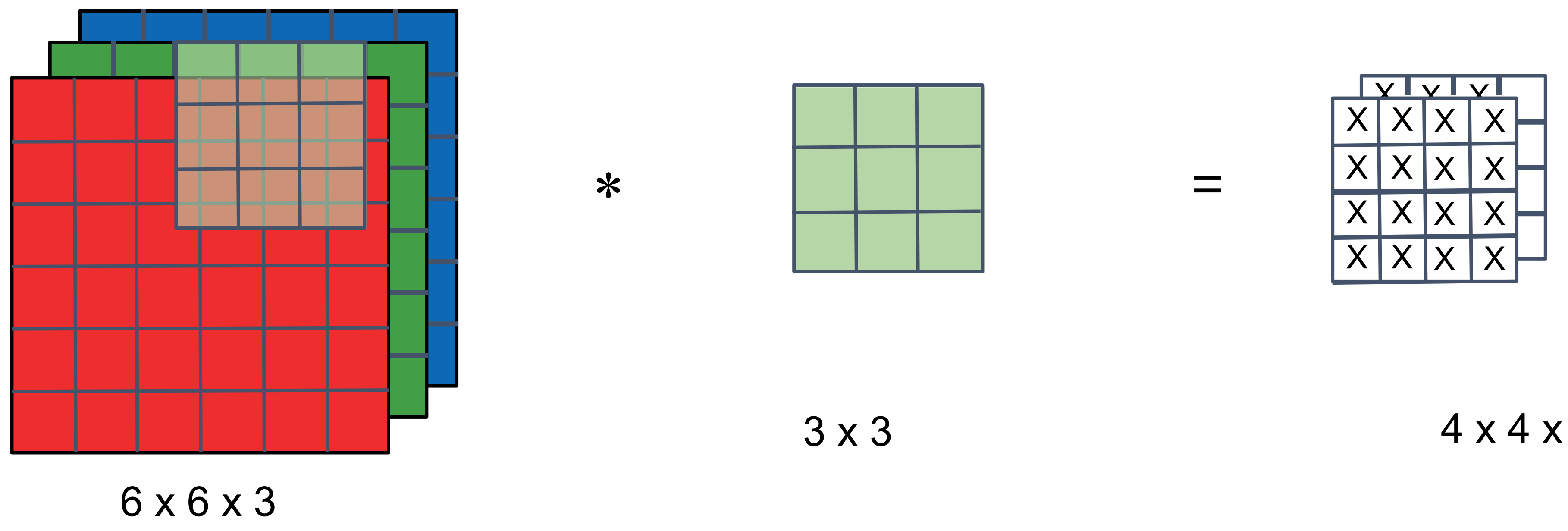
Depthwise Convolution



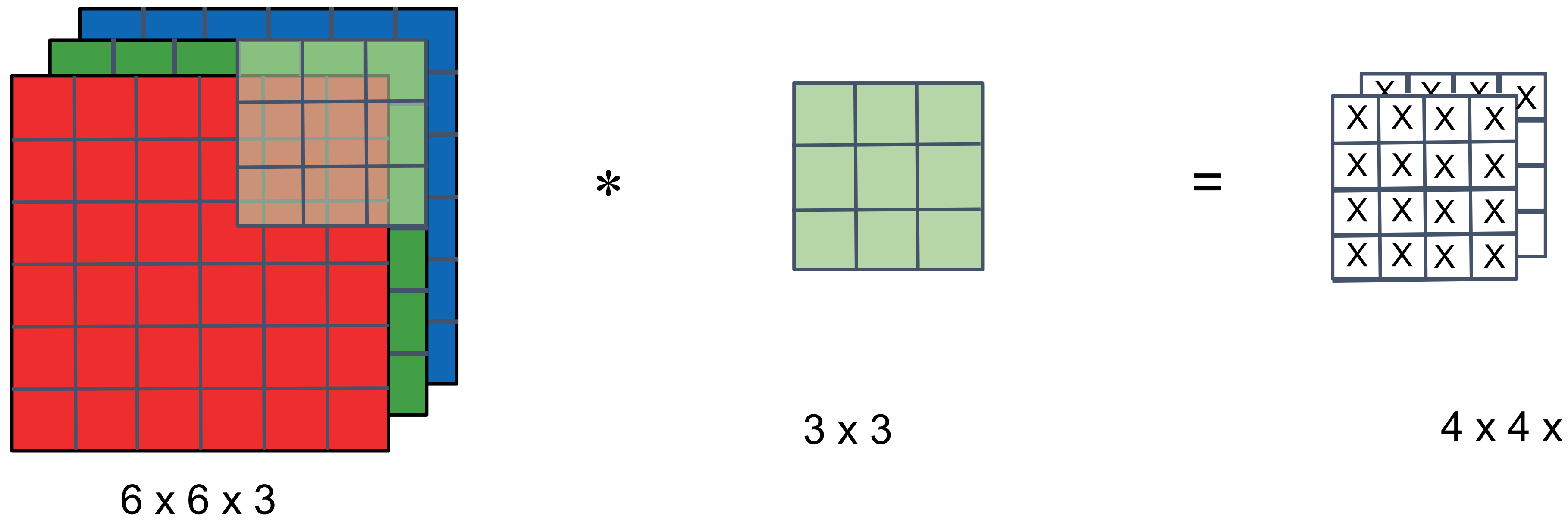
Depthwise Convolution



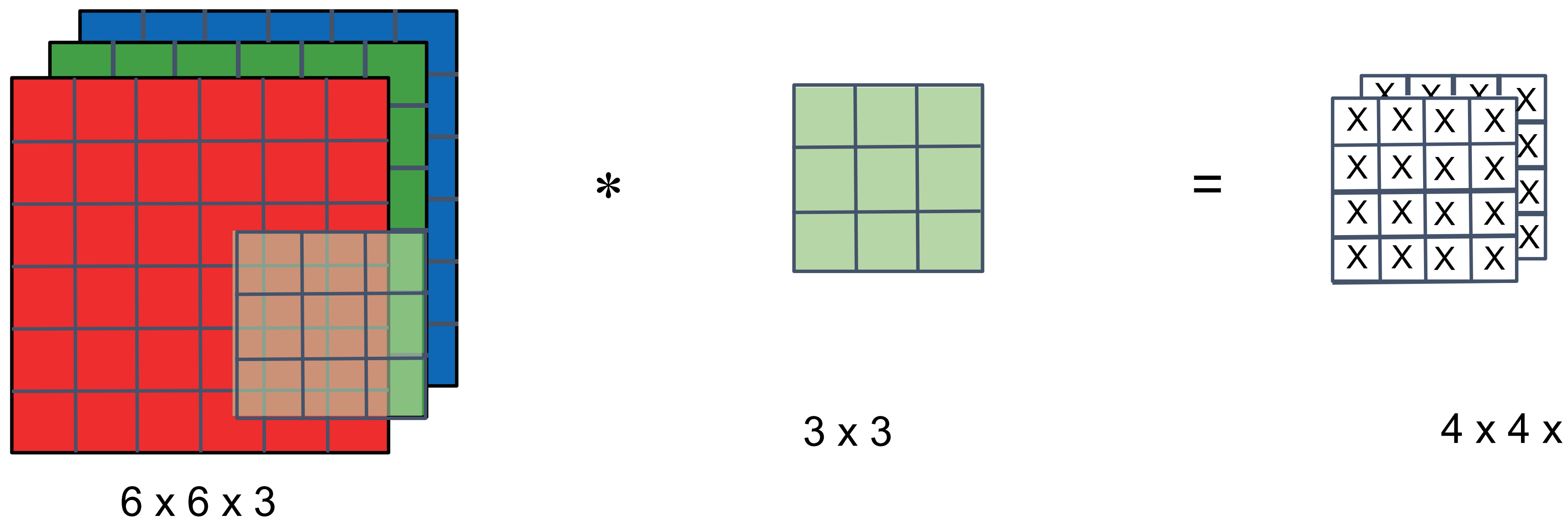
Depthwise Convolution



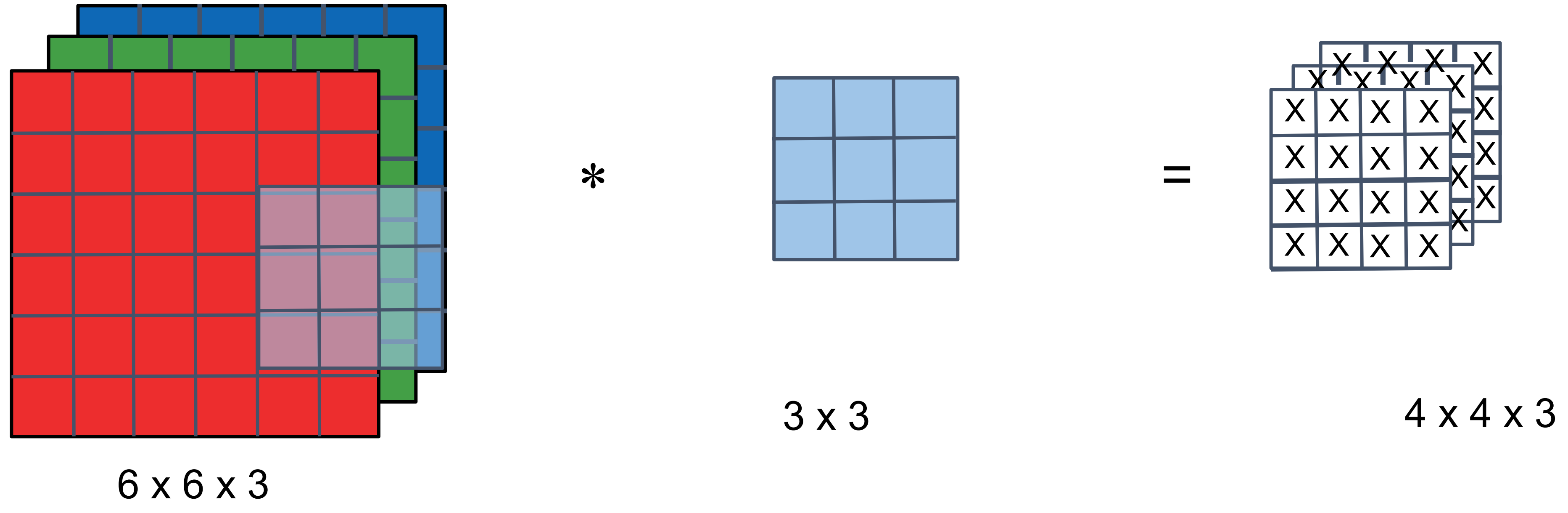
Depthwise Convolution



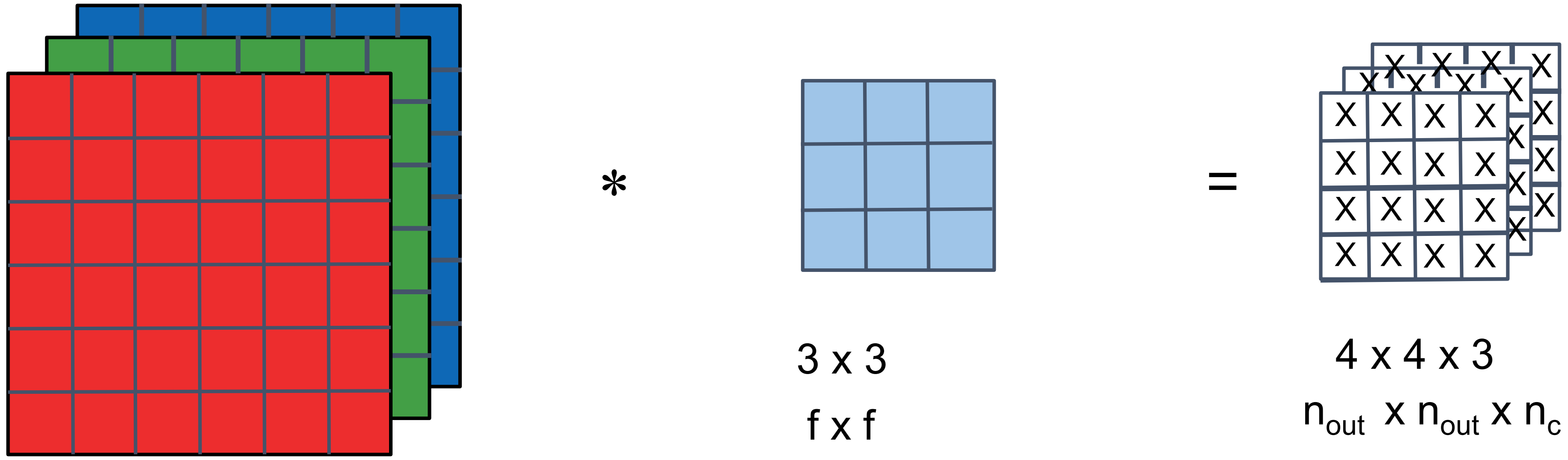
Depthwise Convolution



Depthwise Convolution



Depthwise Convolution



6 x 6 x 3

$n \times n \times n_c$

3 x 3

f x f

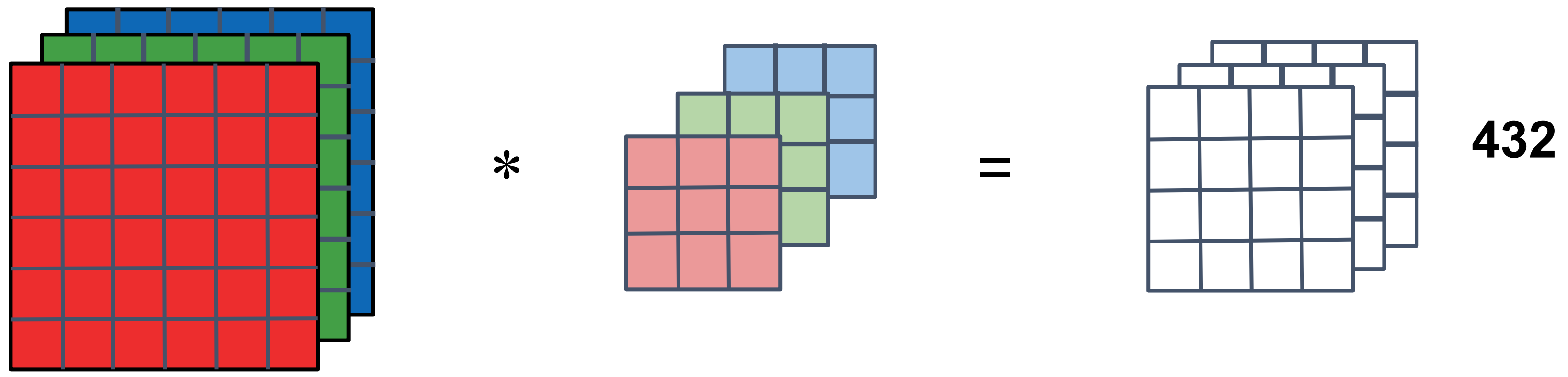
4 x 4 x 3

$n_{out} \times n_{out} \times n_c$

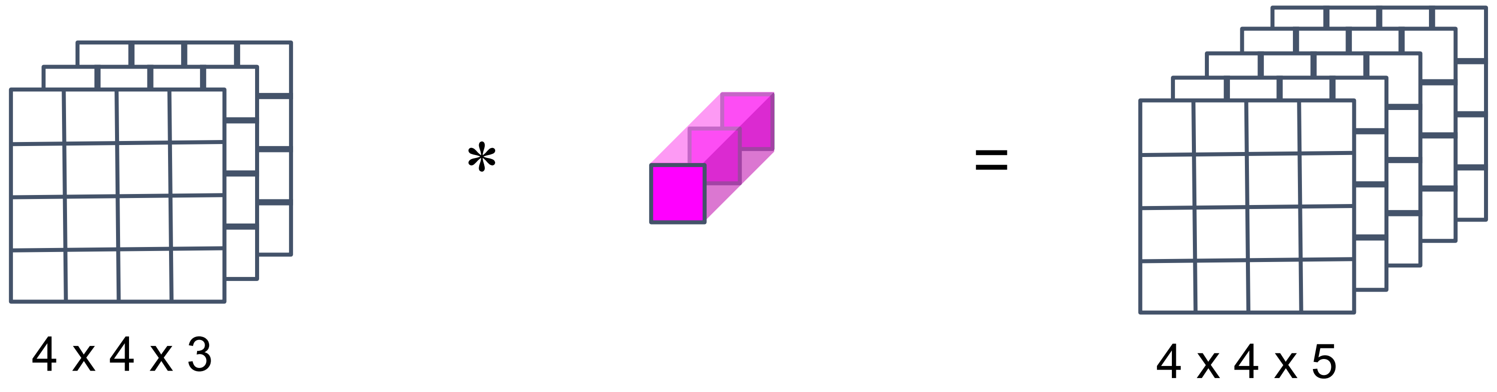
Computational cost	=	#filter params	x	# filter positions	x	# of filters
432		**3 x 3**		**4 x 4**	**x**	**3**

Depthwise Separable Convolution

- Depthwise Convolution



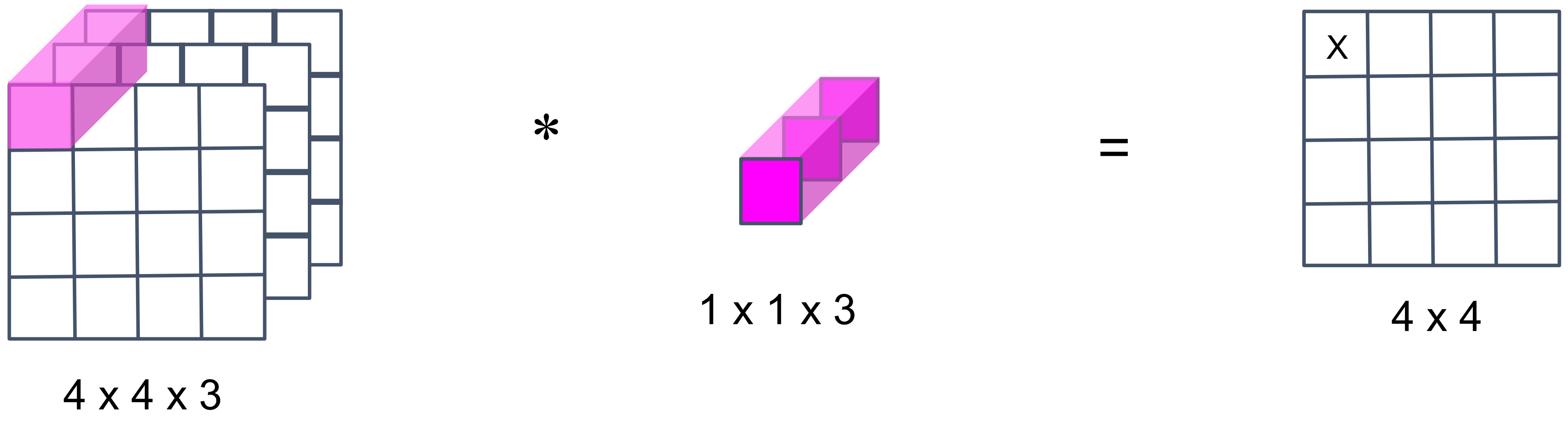
Pointwise Convolution



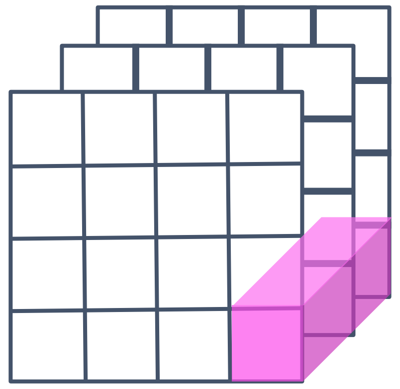
Pointwise Convolution

- The pointwise convolution step involves convolving the intermediate feature maps with a set of 1x1 filters.
- The computational cost of this operation is proportional to the number of filter parameters, the number of filter positions, and the number of output channels.

Pointwise Convolution



Pointwise Convolution



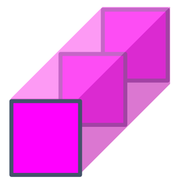
4 x 4 x 3

n_{out} x n_{out} x n_c

$*$

n_c' filter

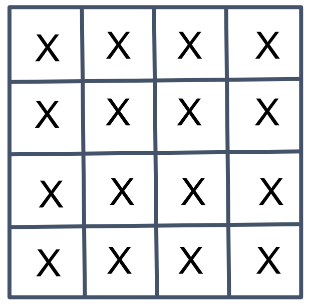
5



1 x 1 x 3

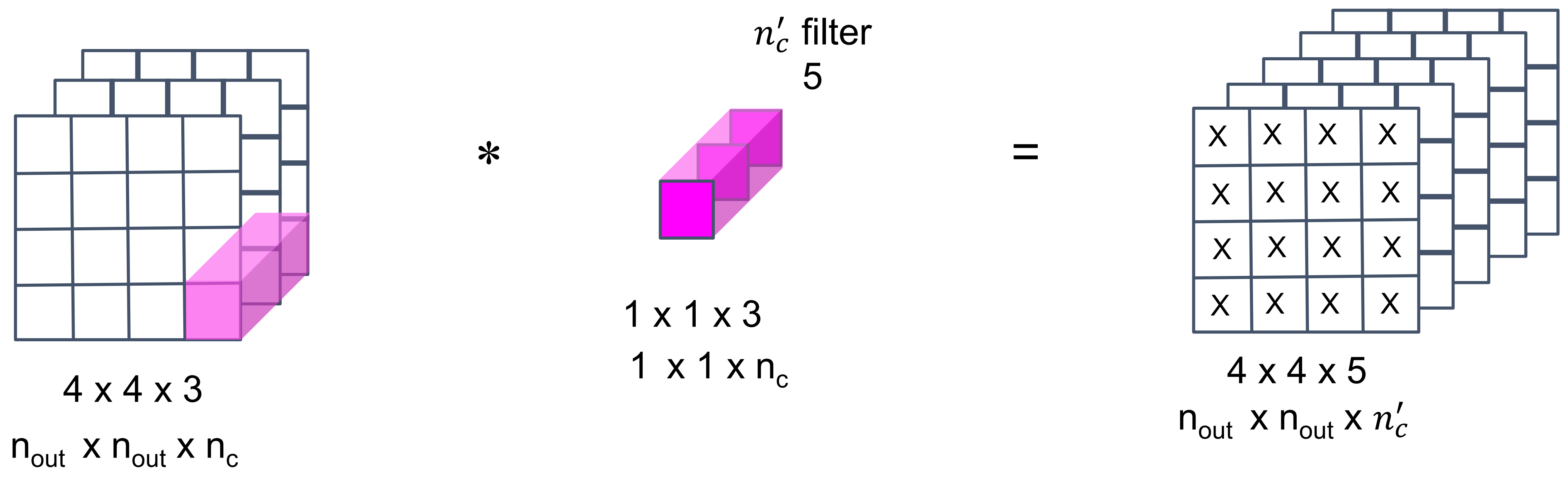
1 x 1 x n_c

=



4 x 4

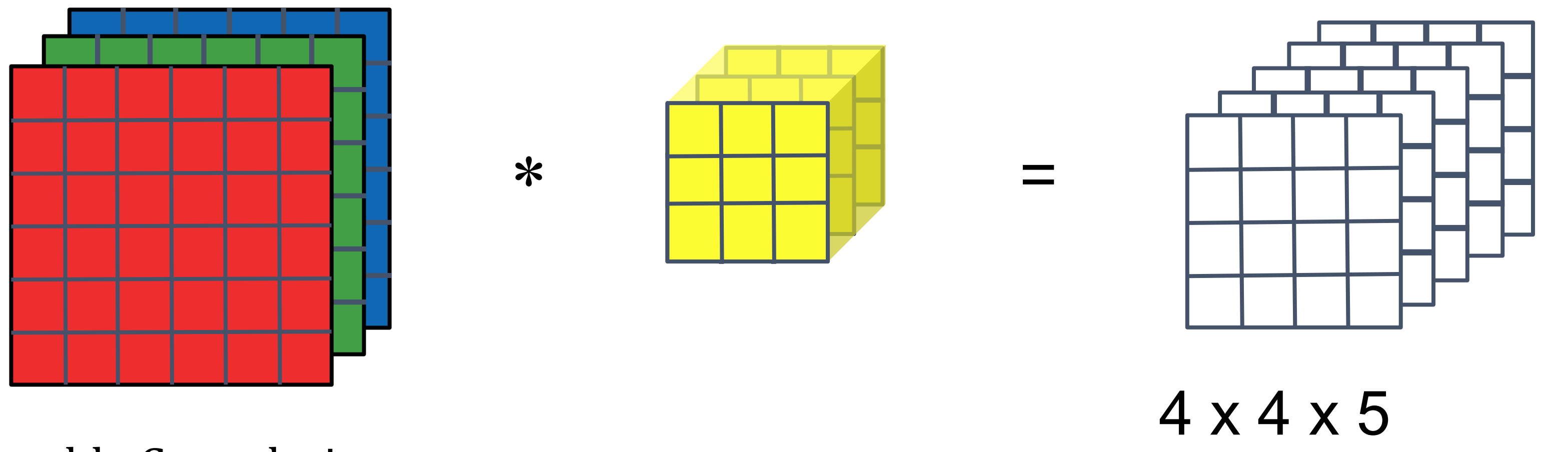
Pointwise Convolution



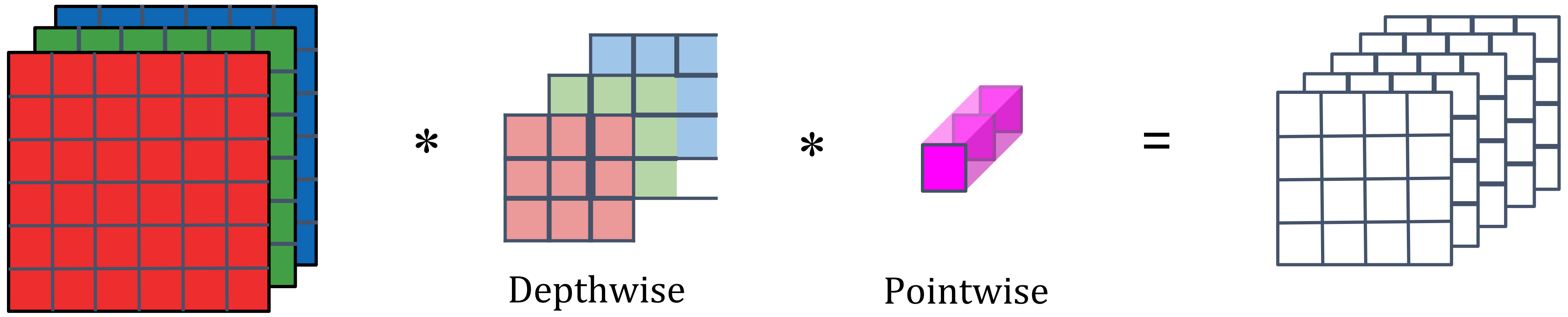
Computational cost	=	#filter params	x	# filter positions	x	# of filters
240		**1 x 1 x 3**		**4 x 4**	**x**	**5**

Depthwise Separable Convolution

- Normal Convolution



Depthwise Separable Convolution



Cost Summary

- Cost of normal convolution: 2160

- Cost of depthwise separable convolution:

  Depthwise + pointwise

  432 + 240 = 672

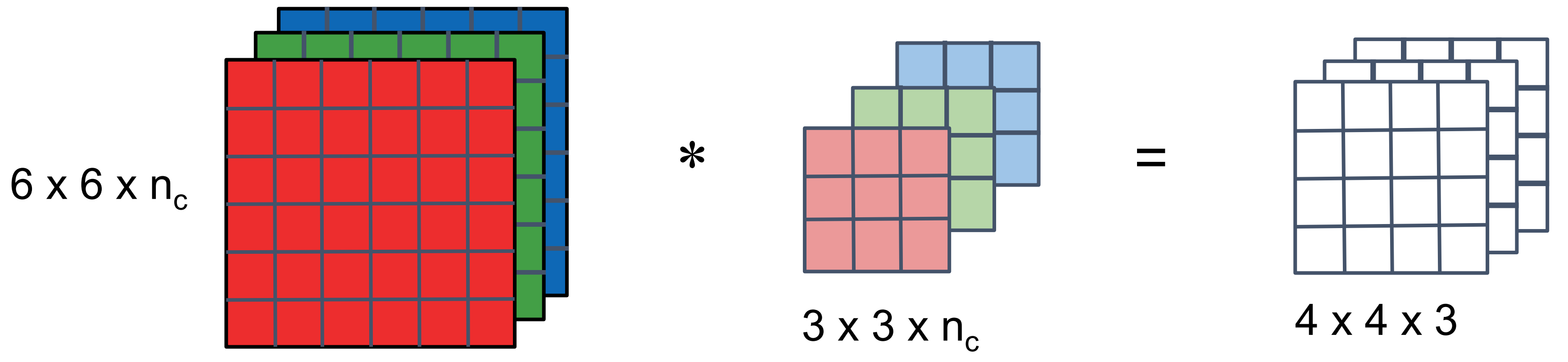
  672 / 2160 = 0.31

  This cost ratio is generally proportional to one over the number of output channels plus one over the filter size squared:

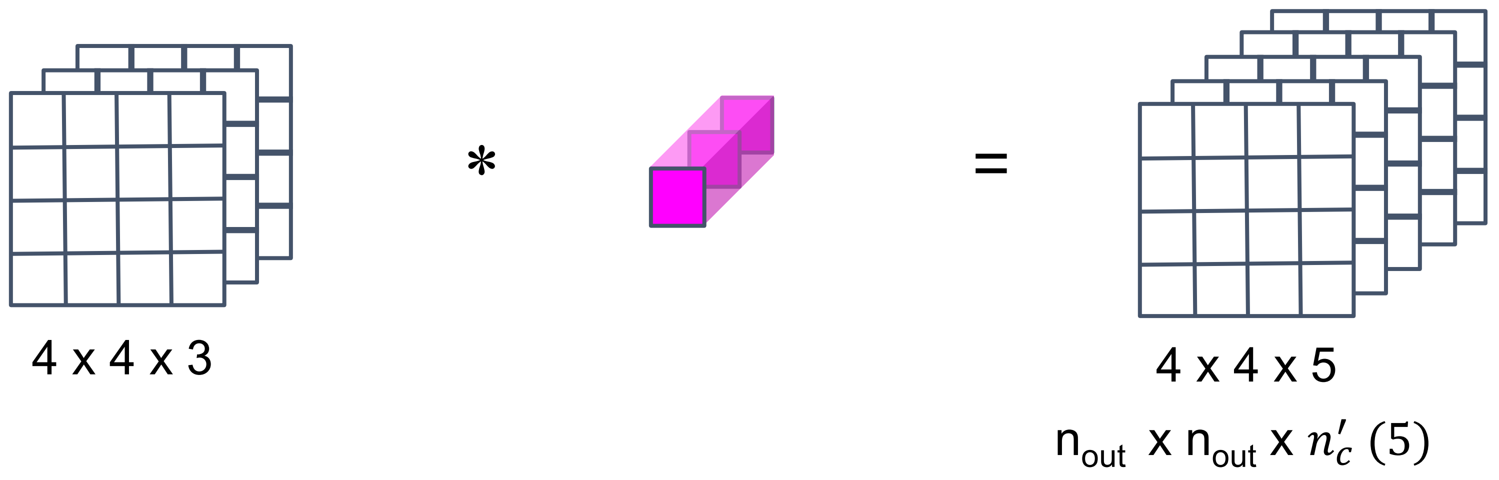
  $\frac{1}{n_c'} + \frac{1}{f^2} = \frac{1}{5} + \frac{1}{9} = 0.31$

Depthwise Separable Convolution

- Depthwise Convolution

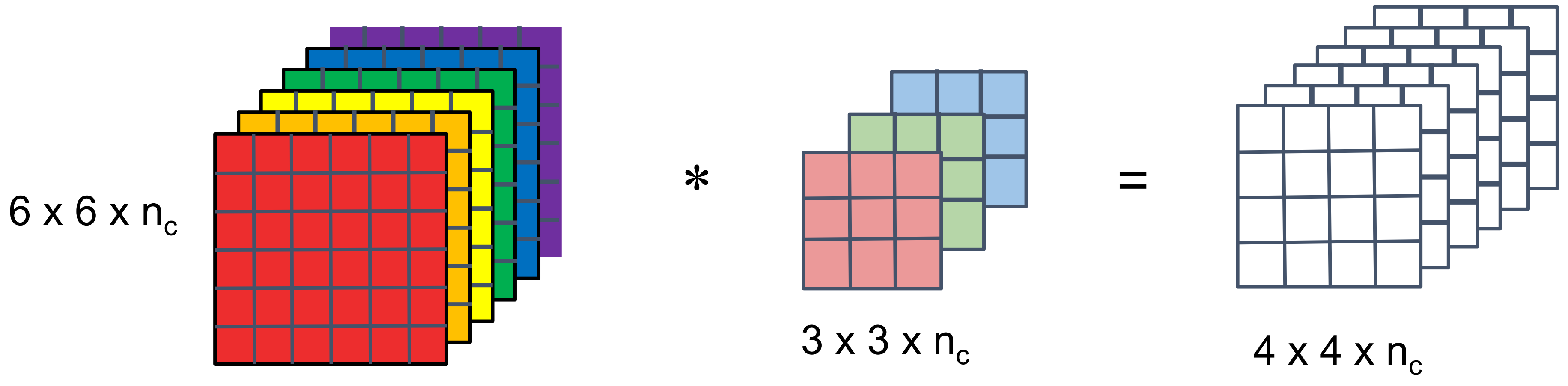


Pointwise Convolution

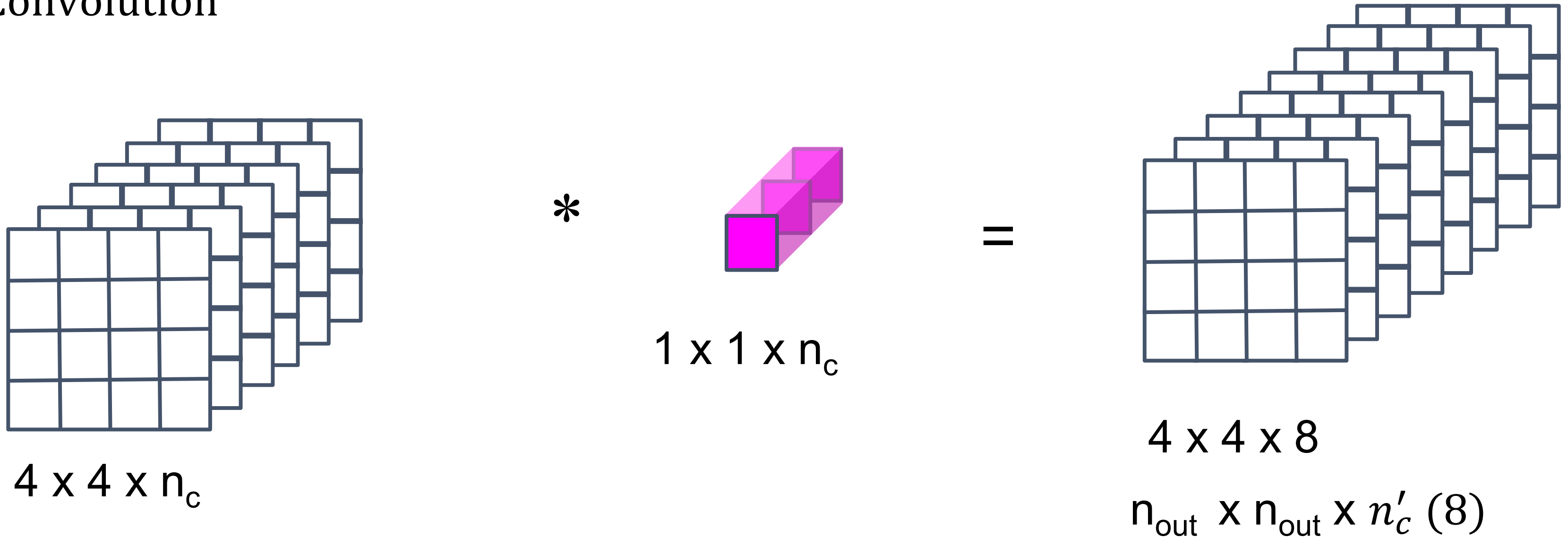


Depthwise Separable Convolution

- Depthwise Convolution



Pointwise Convolution



FPT Education

FPT UNIVERSITY

Convolutional Neural Networks

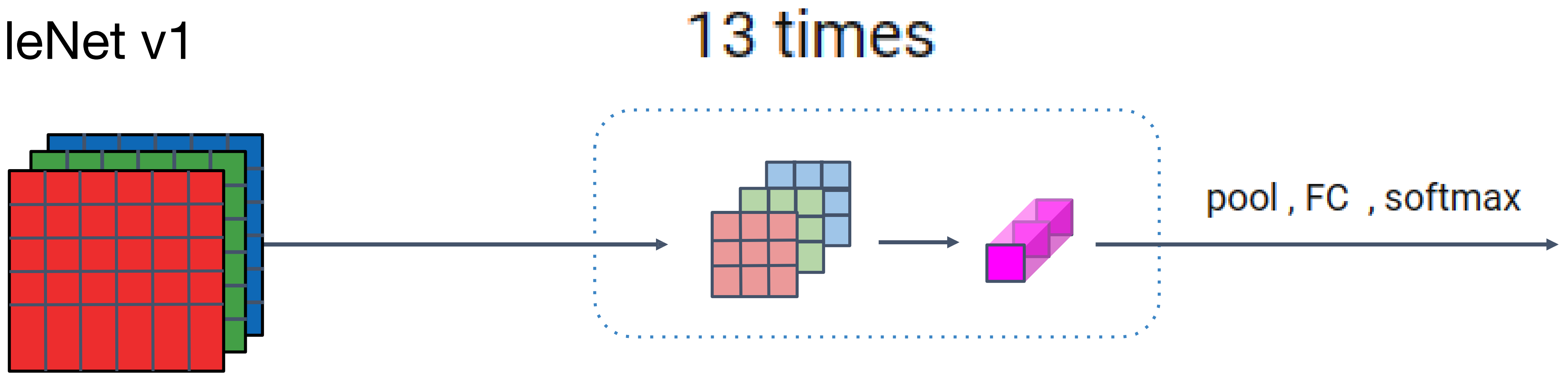
MobileNet Architecture

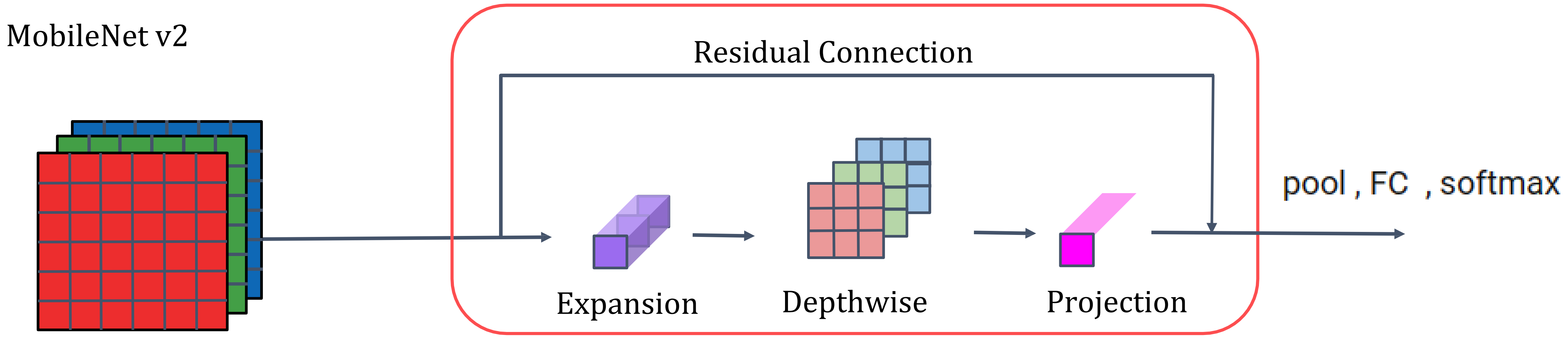
MobileNet

- The main idea behind MobileNet is to replace expensive convolutional operations with less expensive depthwise separable convolutional operations, which consist of a depthwise convolution operation followed by a pointwise convolution operation.
- In MobileNet v1, this idea is implemented using a block that includes a depthwise convolutional operation and a stack of 13 of these layers in order to go from the original raw input image to make a classification prediction.

MobileNet

- MobileNet v1

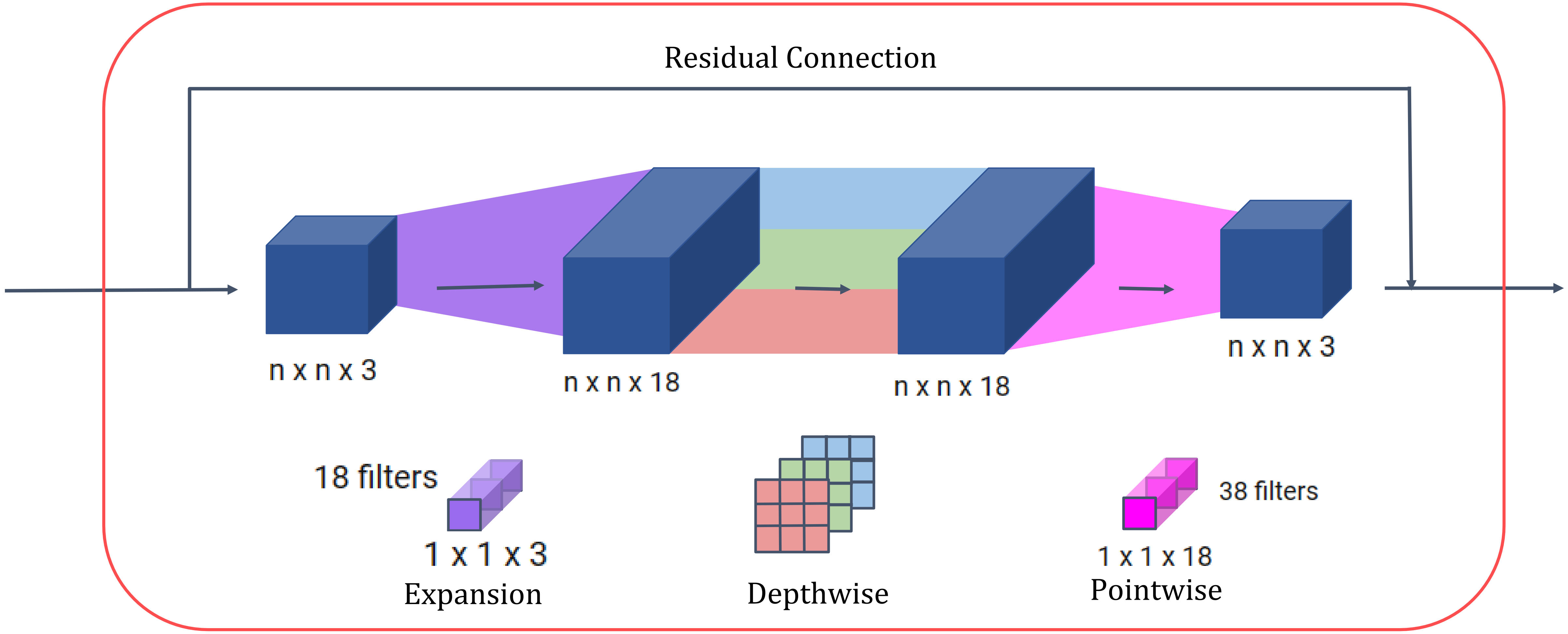




MobileNet v2 Bottleneck

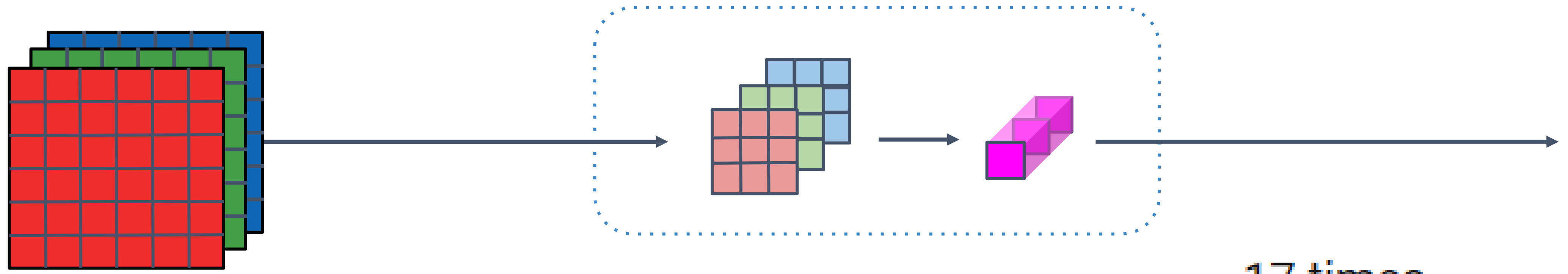
- In MobileNet v2, there are two main changes:
- One is the addition of a residual connection, similar to ResNet, which takes the input from the previous layer and sums it or passes it directly to the next layer, allowing gradients to propagate backward more efficiently.
- The second change is the addition of an expansion layer before the depthwise convolution, followed by the pointwise convolution, which is called projection in MobileNet v2. The expansion layer increases the dimension of the input by a factor of six, while the projection layer reduces it back down to a smaller set of values, thus enabling a richer set of computations while keeping the memory requirements relatively small.

MobileNet v2 Bottleneck

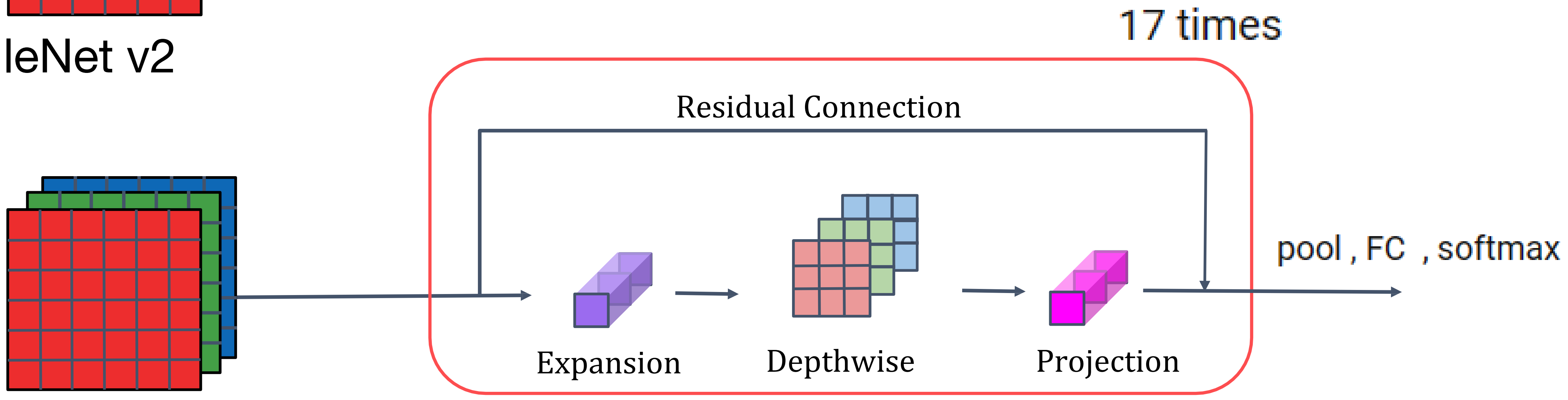


MobileNet

- MobileNet v1



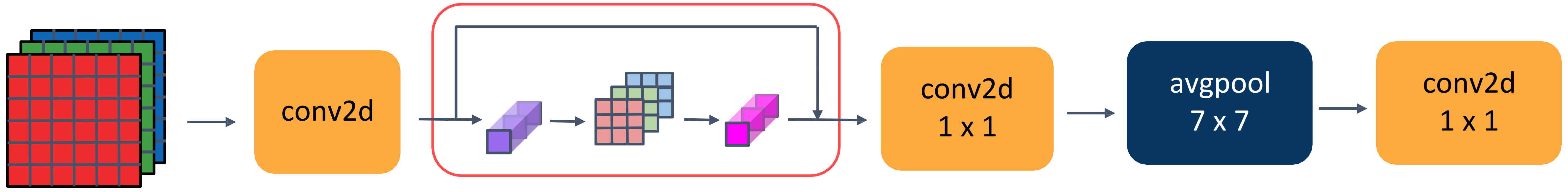
- MobileNet v2



MobileNet v2 Full Architecture

- The bottleneck block in MobileNet v2 uses the expansion operation to increase the size of the representation within the bottleneck block, allowing the neural network to learn a richer function. The pointwise convolution or projection operation is used to project it back down to a smaller set of values, reducing the memory requirements. The MobileNet v2 architecture repeats this bottleneck block 17 times and ends with a pooling layer, a fully connected layer, and a softmax layer to generate the classification output.
- MobileNet v2 achieves better performance than MobileNet v1 while continuing to use only a modest amount of compute and memory resources. If you are interested in building efficient neural networks, the video also briefly mentions the concept of efficient nets as another useful idea to learn about.

MobileNet v2 Full Architecture



FPT Education

FPT UNIVERSITY

Convolutional Neural Networks

EfficientNet

EfficientNet

- With MobileNet, you can build computationally efficient neural networks by using depthwise separable convolutions and bottleneck blocks. With EfficientNet, you can adapt your network to a specific device, allowing you to build neural networks for mobile and embedded devices, where computation and memory are limited.
- EfficientNet is a convolutional neural network architecture developed by Mingxing Tan and Quoc V. Le at Google. It was designed to achieve state-of-the-art accuracy on image classification tasks while minimizing the number of parameters and computational resources needed.

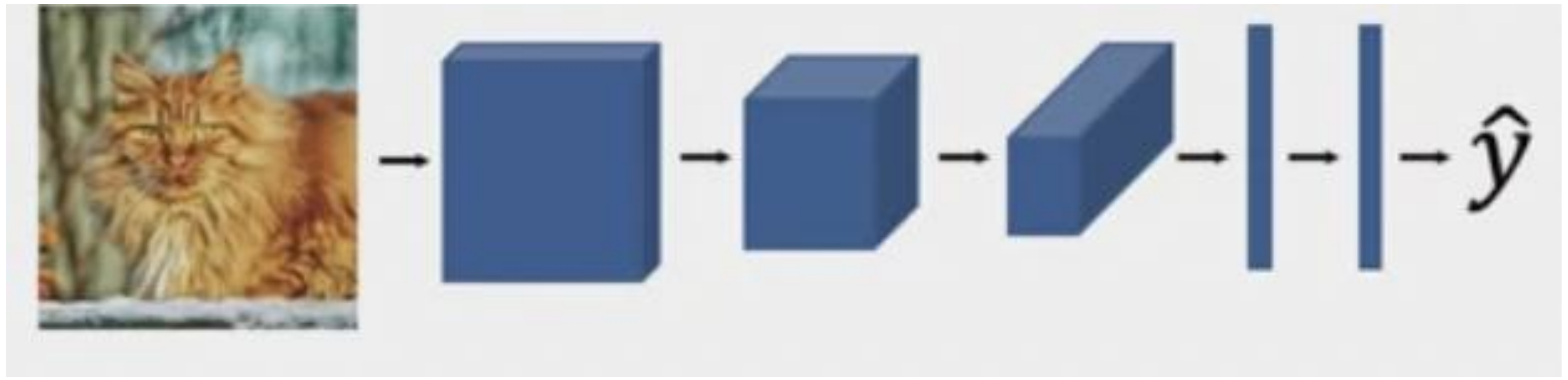
EfficientNet

- EfficientNet is a method that enables scaling of a neural network for a specific device, based on its computational budget. It uses compound scaling to simultaneously scale up or down the resolution of the image, the depth, and the width of the neural network. The goal is to choose the optimal values of these parameters to get the best possible performance.

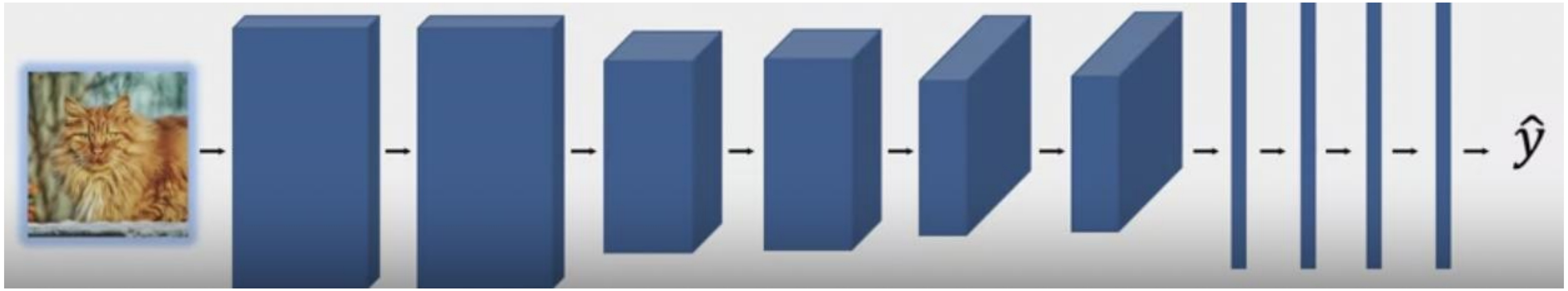
- EfficientNet provides an open-source implementation to help find the right balance between these factors and optimize performance.

EfficientNet

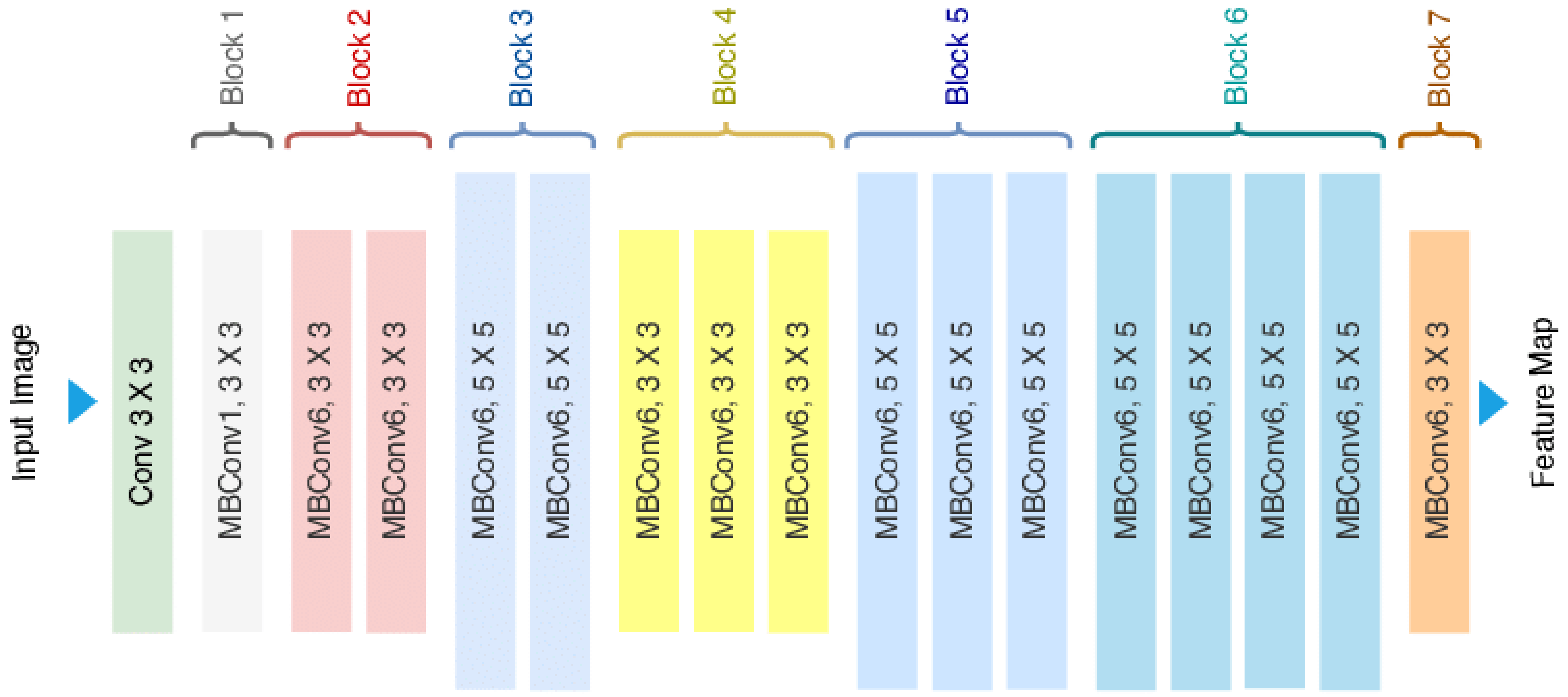
Baseline



Compound scaling



EfficientNet



FPT Education

FPT UNIVERSITY

Practical advice for using ConvNets

Transfer Learning

Transfer Learning

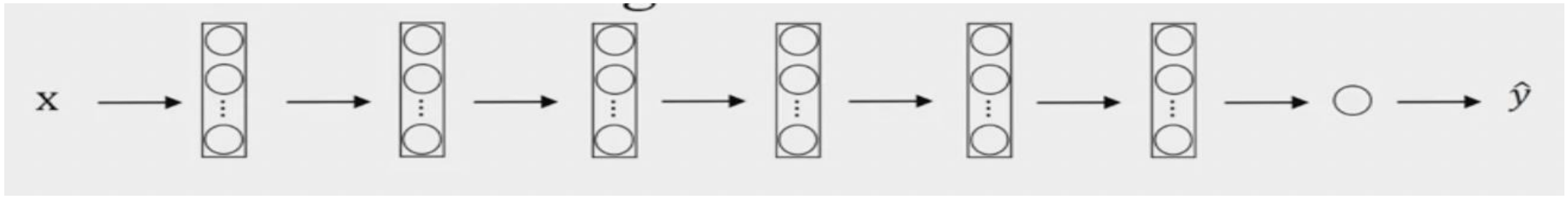
- Transfer learning is a technique in computer vision that enables faster progress by using pre-trained neural network models. Instead of starting from scratch, pre-trained models' weights can be used as a starting point for a new model, which is useful when working with small training sets or limited computational resources.
- Pre-trained models, which are available for free, have been trained on large data sets such as ImageNet, MS COCO, and Pascal. The early layers of the pre-trained model are usually frozen, and the later layers are fine-tuned to the specific problem.

Transfer Learning

- Transfer learning is a technique in computer vision that uses pre-trained neural network models to accelerate progress by using their weights as a starting point for your own model.
- The early layers of the pre-trained model are typically frozen, while the later layers are fine-tuned to your specific problem.
- The number of layers that are frozen and fine-tuned depends on the size of your training set, and pre-computing activations for the frozen layers can speed up training.
- Overall, transfer learning is a powerful technique that can save time and improve performance in computer vision applications.

Transfer Learning

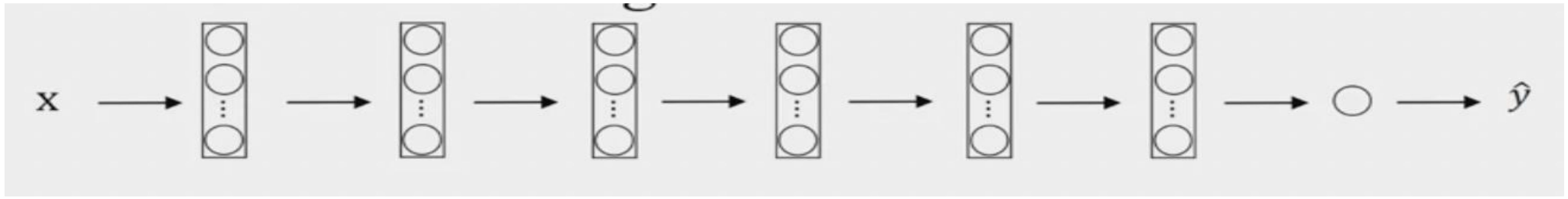
Tigger | Misty | Neither



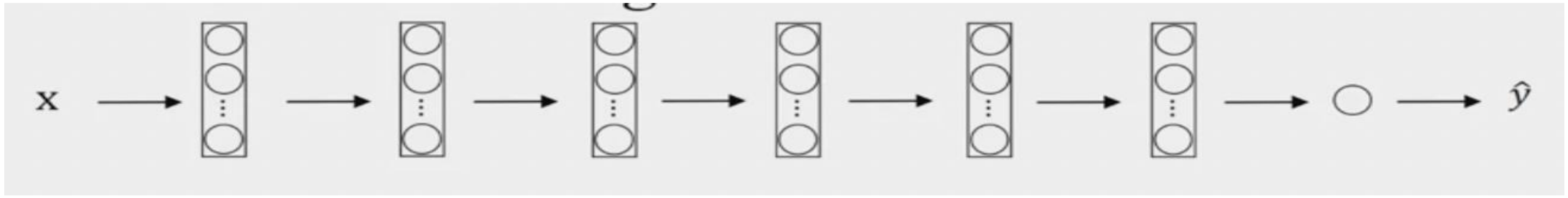
freeze

trainable parameters = 0

freeze = 1

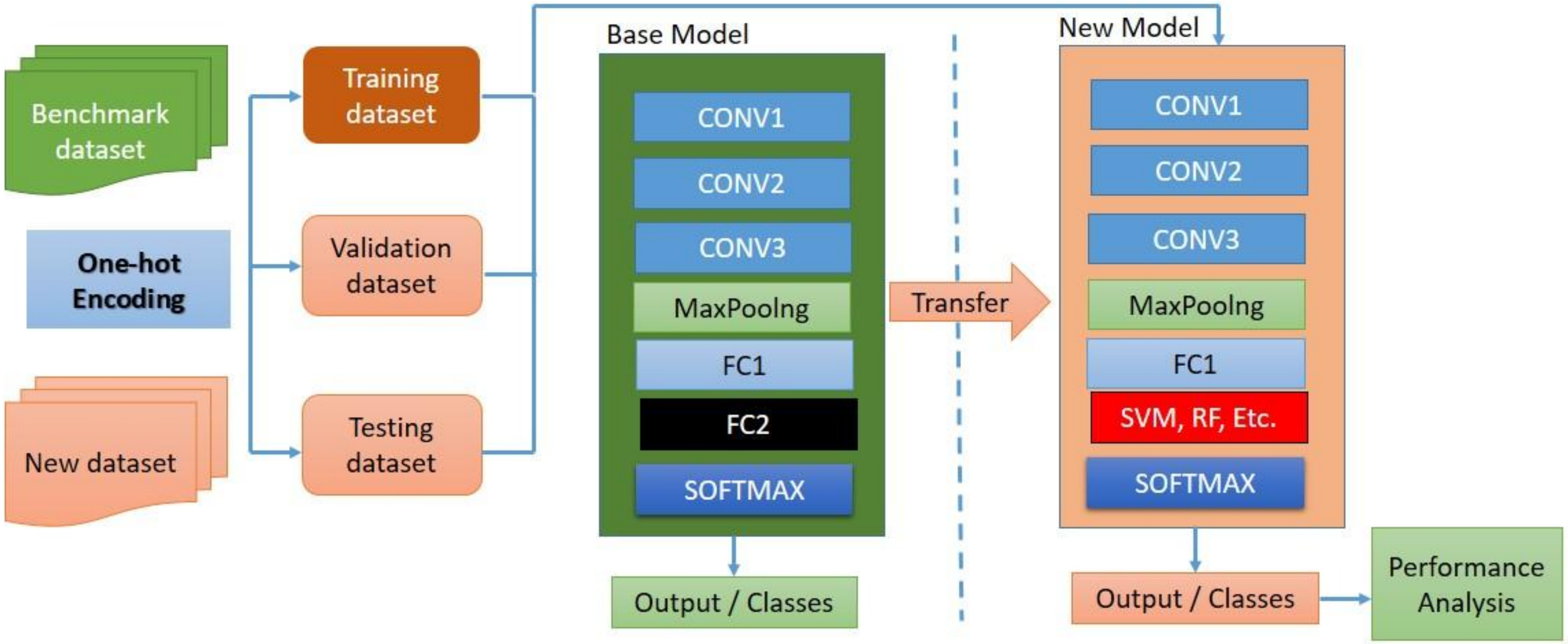


freeze



train

Transfer Learning: Base Model and the New Model



© Source: www.AmitRay.Com Fight Against COVID-19 AI Projects - Dr. Amit Ray

FPT Education

FPT UNIVERSITY

Practical advice for using ConvNets

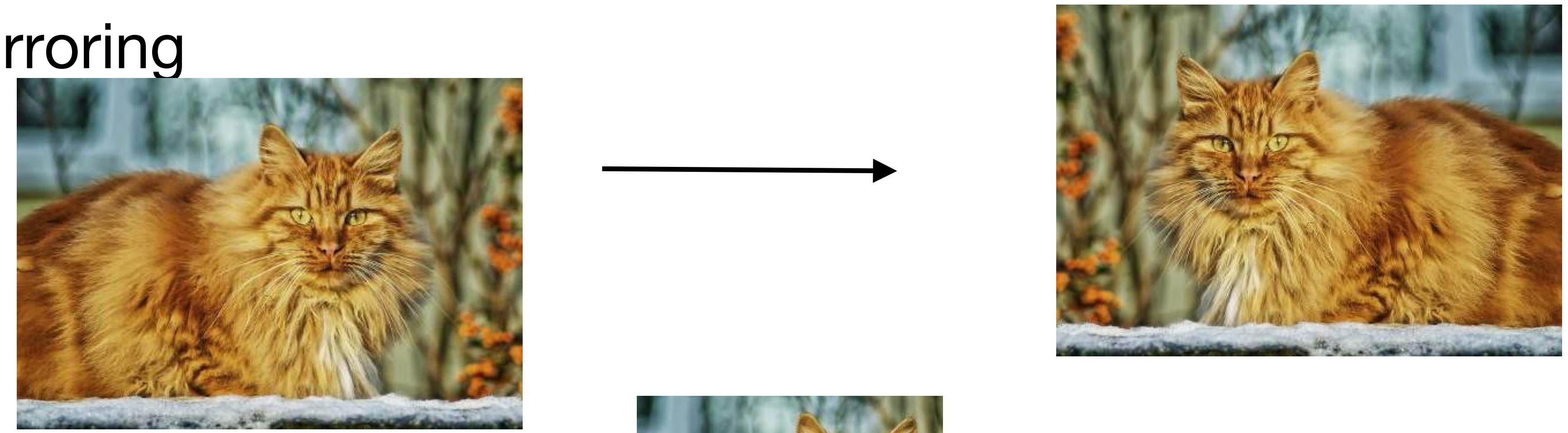
Data augmentation

Data augmentation

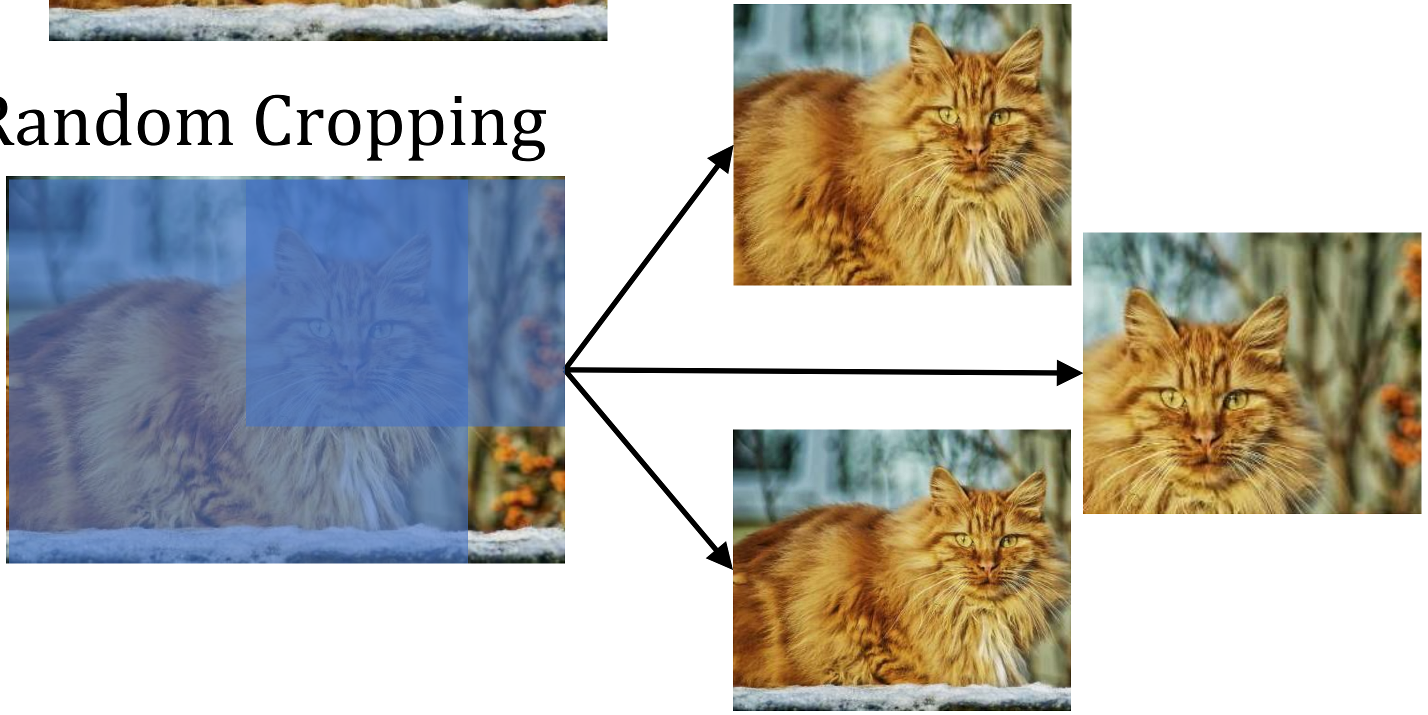
- Data augmentation is a technique used in machine learning and computer vision to artificially increase the size of a dataset by applying various transformations to the original data.
- The goal is to create new and diverse examples that can help improve the performance and generalization of a machine learning model.
- Common data augmentation techniques include flipping, rotating, scaling, cropping, adding noise, changing brightness and contrast, and applying various filters. By applying these transformations, the model is exposed to a wider range of inputs, which can help it learn to recognize patterns and variations in the data more effectively.
- Random cropping is another common technique where different random crops of the dataset are taken to create new examples. In addition, color shifting can be used to introduce different color distortions to the image, making the model more robust to changes in lighting conditions.

Common augmentation method

- Mirroring



Random Cropping



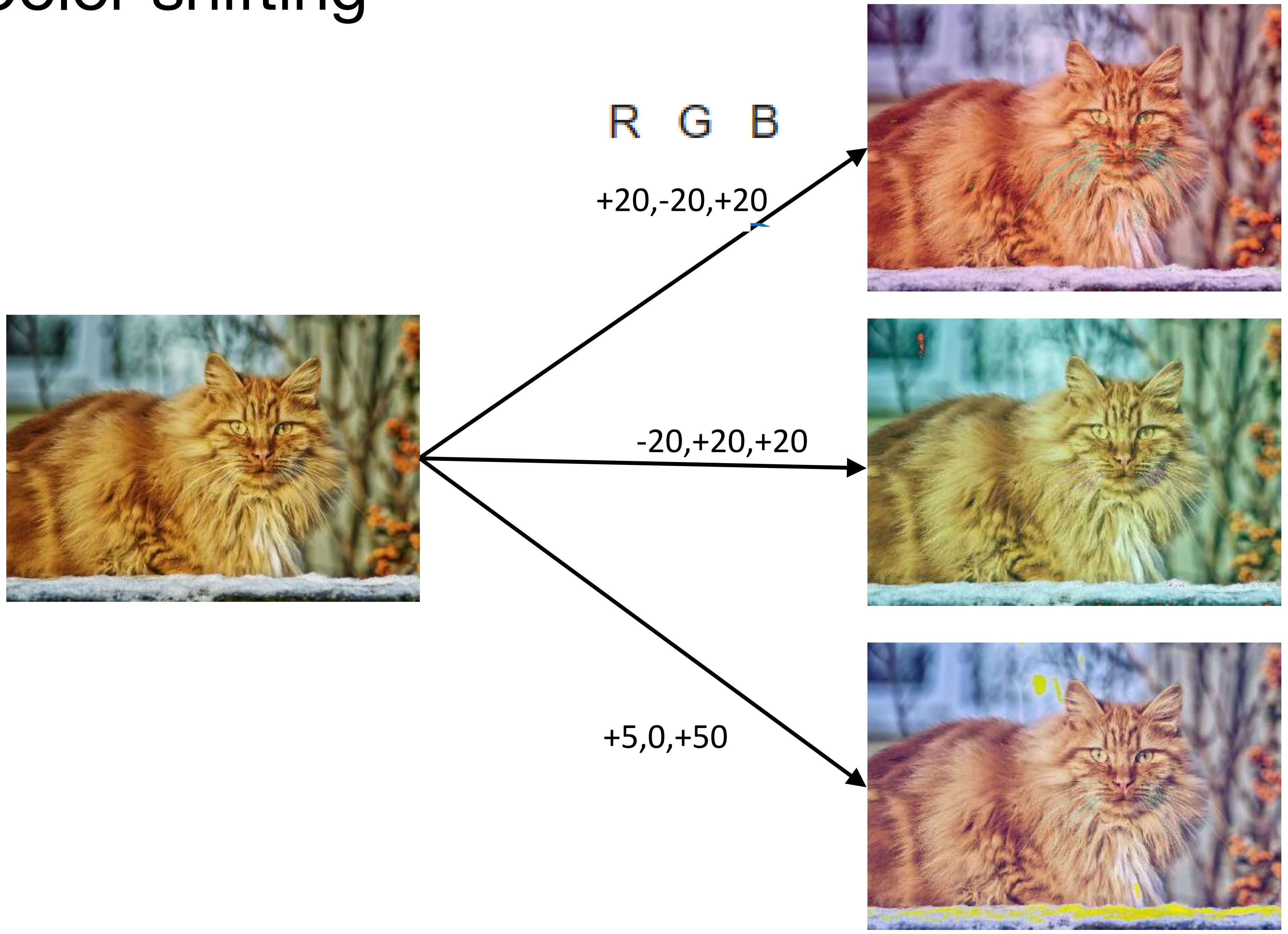
Rotation

Shearing Local

warping

...

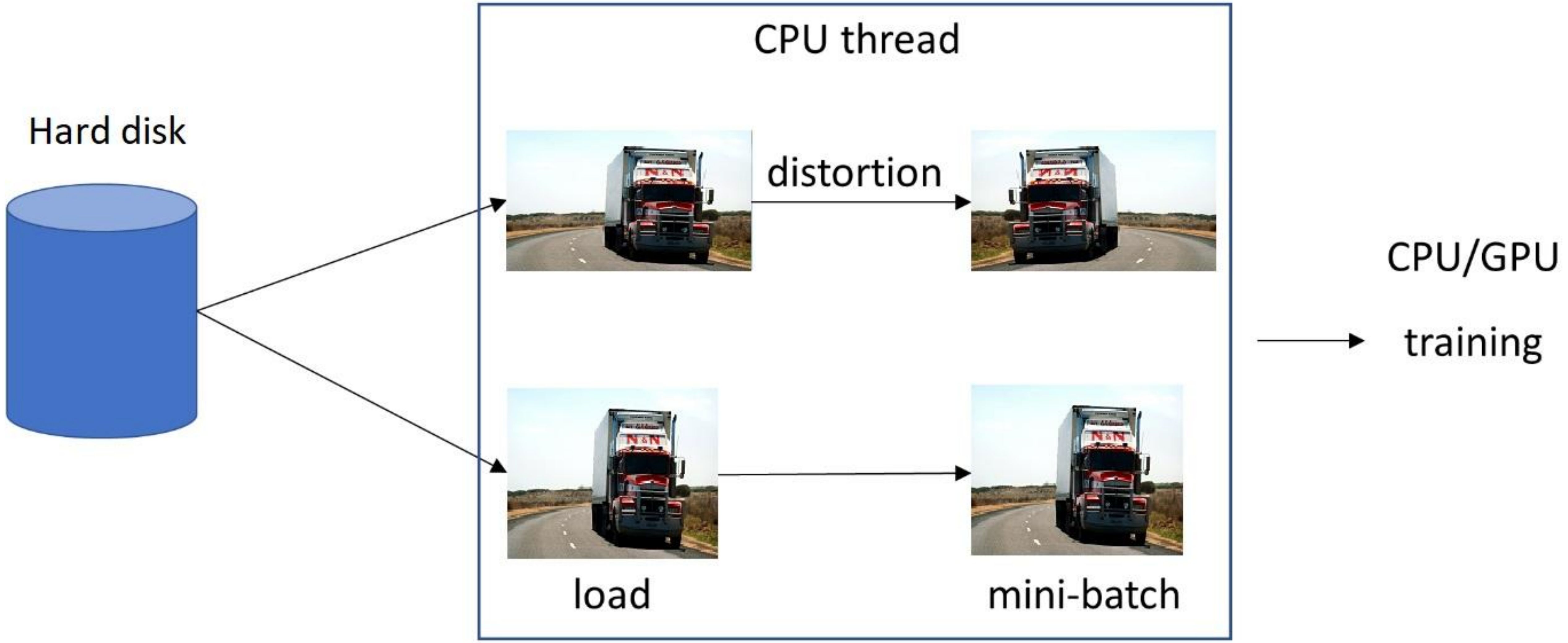
Color shifting



Implementing distortions during training

- To implement data augmentation, one can use a CPU thread to constantly load images from the hard disk and apply distortions to each image before passing it to another thread or process responsible for training the deep neural network. Hyperparameters such as the amount of color shifting or the specific parameters for random cropping can be adjusted to capture more invariances and improve the model's performance.
- In general, using someone else's open-source implementation for data augmentation is a good place to start, but it may also be necessary to fine-tune hyperparameters for specific applications.

Implementing distortions during training



Practical advice
for using ConvNets

The state of computer vision

The state of computer vision

- Deep learning has been successfully applied to various fields, including computer vision, natural language processing, speech recognition, online advertising, logistics, and more.
- However, computer vision is unique in the sense that it requires a lot of data to train a model due to the complexity of the problem. Even with large datasets, it feels like there's still not enough data to train a good model for computer vision, and therefore, hand-engineering is often necessary to get good performance.

Data vs. hand-engineering

- The two sources of knowledge for a learning algorithm: labeled data and hand-engineering.
- When there is little labeled data, more hand-engineering is required to get good performance.
- However, when there is a lot of data, simpler algorithms can be used, and less hand-engineering is needed.

Data vs. hand-engineering

Little data

Lost of data

Model had-engineering

Transfer learning

Objective detection

Little data

Image recognition

Speech recognition

Simpler algorithms

Two sources of knowledge

- Labeled data
- Hand engineered features/network architecture/other components

Tips for doing well on benchmarks/winning competitions

- There is a lot of emphasis in the computer vision community on doing well on standardized benchmark datasets and winning competitions. To achieve better performance on these benchmarks, people often use techniques like ensembling (averaging the outputs of multiple independently trained neural networks) and multi-crop at test time (applying data augmentation to the test image). While these techniques are useful for winning competitions, they may not be practical for real-world applications.
- Using someone else's pre-trained model and fine-tuning it on your dataset to get started quickly on an application, especially if you have limited compute resources. However, if you have the resources and inclination, training your own networks from scratch may be necessary to invent your own computer vision algorithm.

Tips for doing well on benchmarks/winning competitions

- Ensembling
- Train several networks independently and average their outputs
- Multi-crop at test time
- Run classifier on multiple versions of test images and average results

Use open source code

- Use architectures of networks published in the literature
- Use open source implementations if possible
- Use pretrained models and fine-tune on your dataset

Summarization

- Classic networks like LeNet, AlexNet, and VGG laid the foundation for modern CNNs.
- ResNets address the vanishing gradient problem using residual connections.
- Network in Network utilizes 1×1 convolutions for enhanced model capacity.
- Inception networks capture information at multiple scales using diverse convolutions.
- MobileNet, optimized for mobile devices, employs depthwise separable convolutions.
- EfficientNet achieves efficiency through compound scaling.
- Transfer learning leverages pre-trained models for new tasks, reducing training time.
- Data augmentation enhances model generalization by applying diverse transformations.
- The current state of computer vision includes self-supervised learning and real-time applications, showcasing advancements and challenges.

Question

- 1. What typically happens to the height (nH), width (nW), and number of channels (nC) as you go deeper in a ConvNet?
- 2. What patterns do you typically observe in a ConvNet architecture?
- 3. In deep networks using “valid” padding, do we usually rely only on pooling layers to reduce the spatial dimensions (height and width)?
- 4. Does adding more layers to a plain network always result in a lower training error?
- 6. Which statements are true about Residual Networks (ResNets)?
- 7. How many parameters (including bias) does a 1×1 convolutional filter have for an input of size 64×64×16?
- 8. What is true about 1×1 convolutions and pooling layers?
- 9. What is true about Inception Networks?
- 10. Why are open-source ConvNet implementations commonly used?
- 11. What are the values of W, Y, and Z in the MobileNet v2 bottleneck block?
- 12. What is true about depthwise separable convolutions?
- 15. What is true about Inception Networks?
- 16. If the input to a MobileNet v2 depthwise convolution layer is 64×64×16 and the expansion uses 32 filters, what is the input and output size of the depthwise convolution layer?
- 17. What is true about depthwise separable convolutions?
- 18. What transfer learning strategy is recommended for small datasets?
- 20. Are 1×1 convolutions the same as multiplying by a single number? (True/False)
- 22. What happens to model performance if a ResNet block is added at the end of a network?
- 24. In a ResNet equation, which part corresponds to the skip connection?
- 26. Is the main motivation behind ResNets to reduce overfitting caused by model complexity? (True/False)