Convolutional Neural Networks

MUSTAFA HAJIJ

Convolution

Recall first the fully connected layer on image. Say that we have 32X32X3 size image. We flatten this image to a vector 3072.

Input x is of size 3072

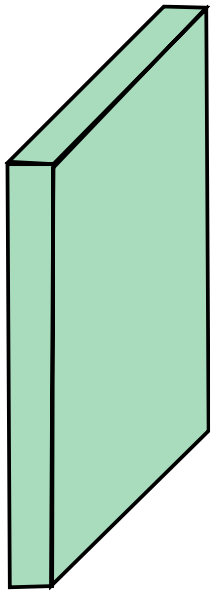
Wx

10 x 3072
weights

Output y is of size 10

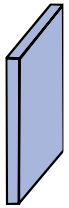
Convolution

32X 32 X 3



image

3X 3 X 3



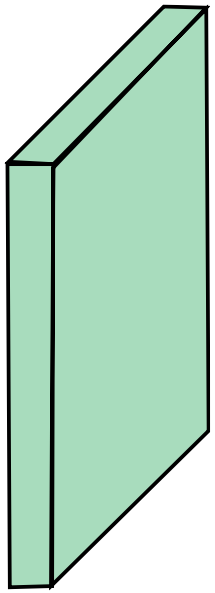
filter

Convolve the filter with the image means sliding the filter,
Over the image spatially and compute the dot product

Convolution

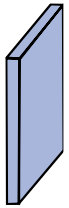
Filters always extends over the full depth of the convolved image

32X 32 X 3



image

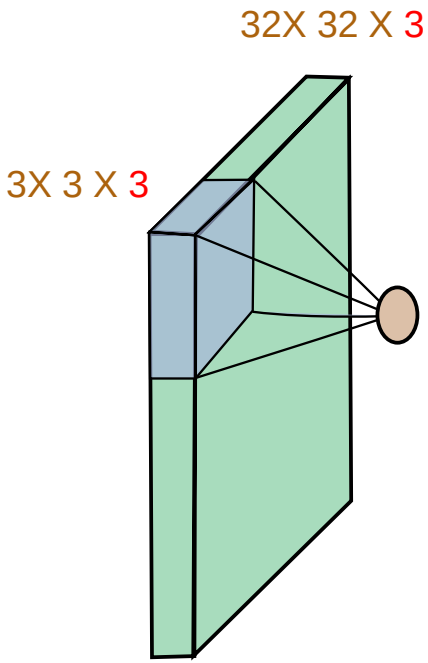
3X 3 X 3



filter

Convolve the filter with the image means sliding the filter,
Over the image spatially and compute the dot product

Convolution

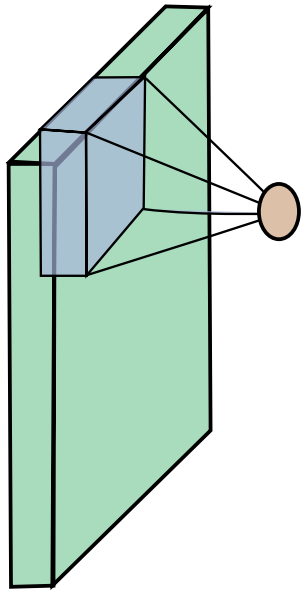


Perform dot product between the
filter and the image (5X5X3 products) + bias

We can write this operation as $w^T x + b$

Convolution

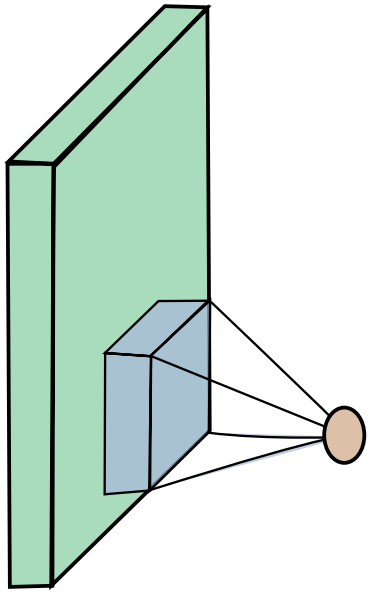
32X 32 X 3



Shift and perform the dot product again

Convolution

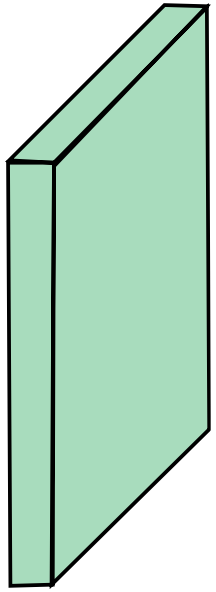
32X 32 X 3



Keep sliding spatially and compute the dot product till
You get to the other bottom right corner

Convolution

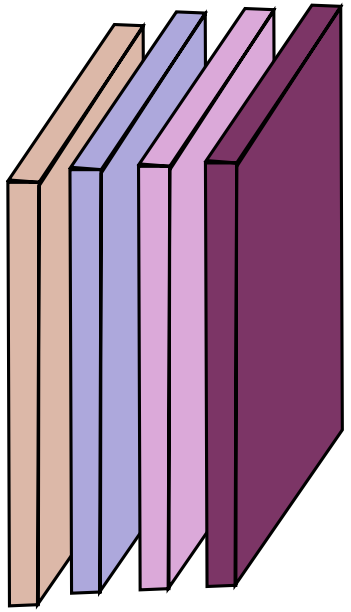
32X 32 X 3



After finishing convolving with 4 filters

Activation maps

30X 30 X 4



Depth = # filters

Convolution

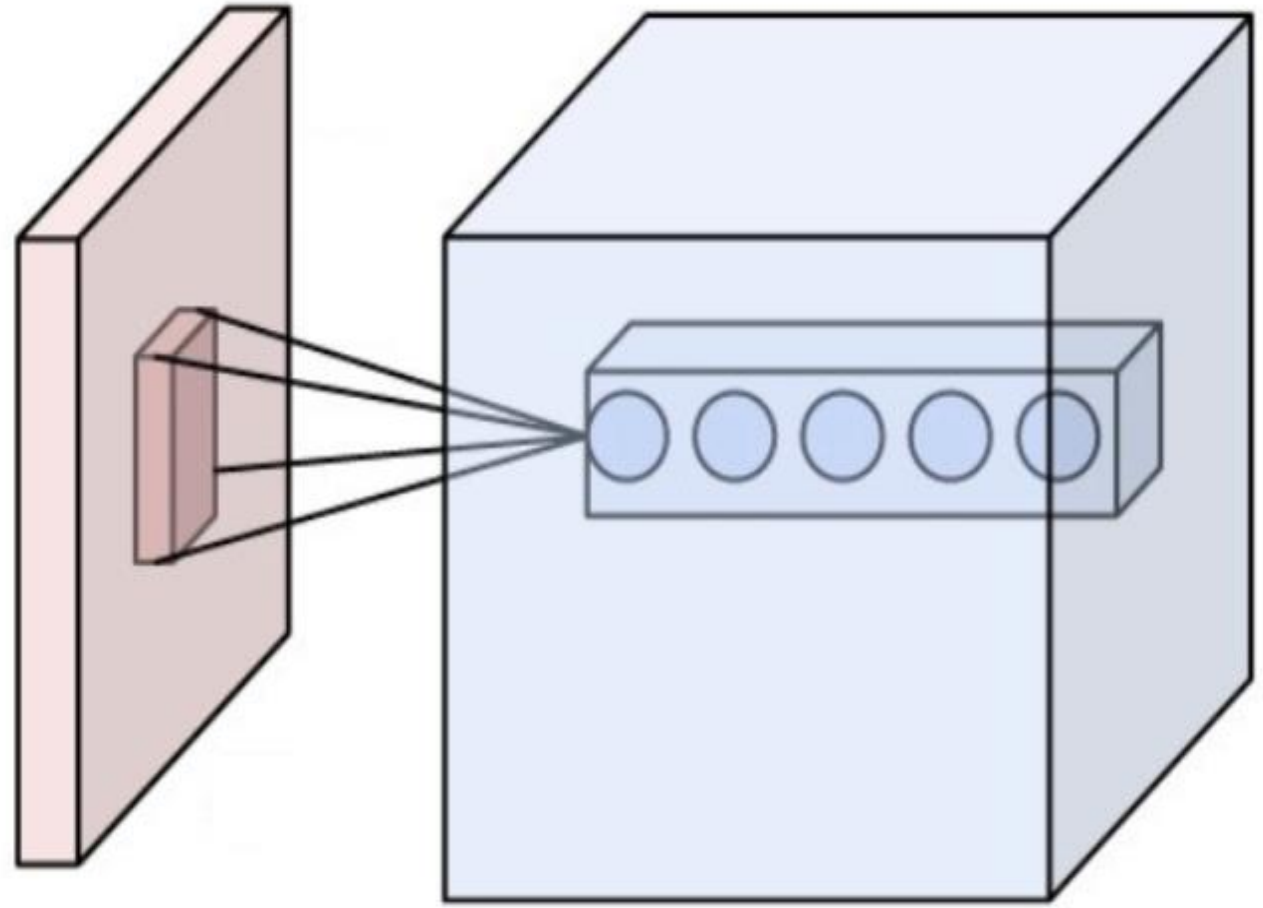
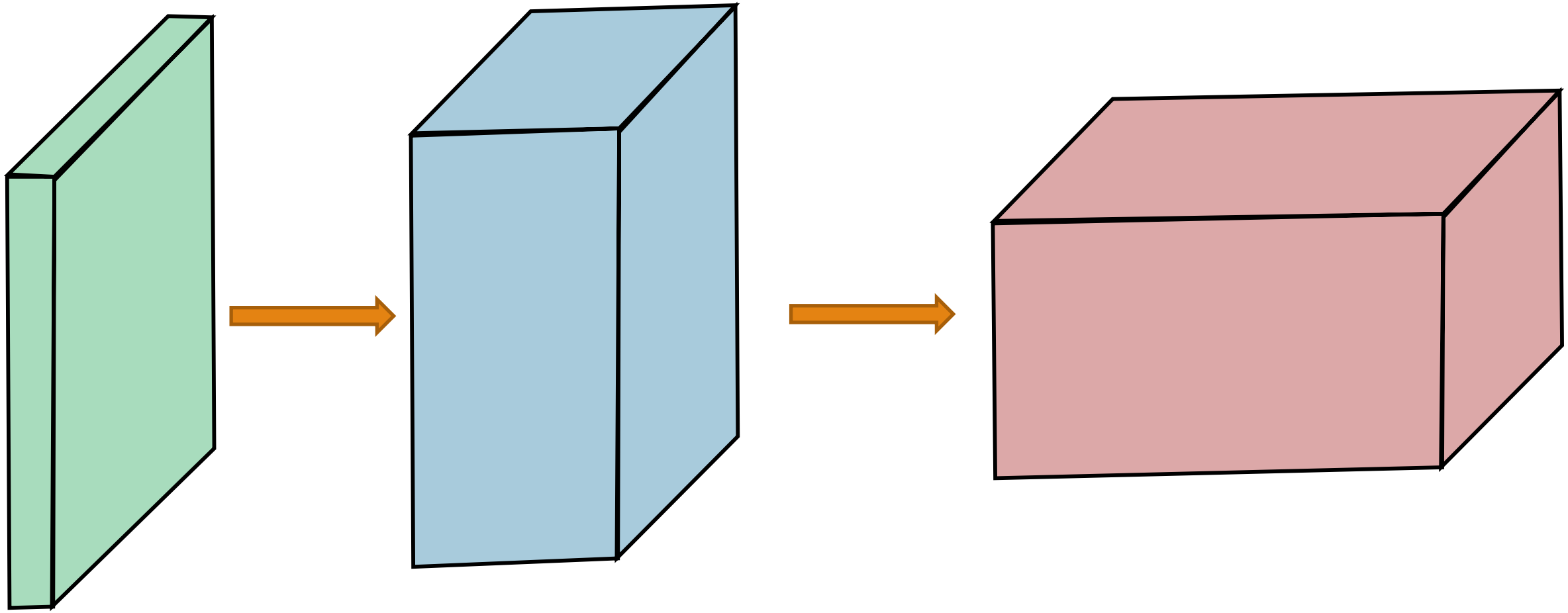


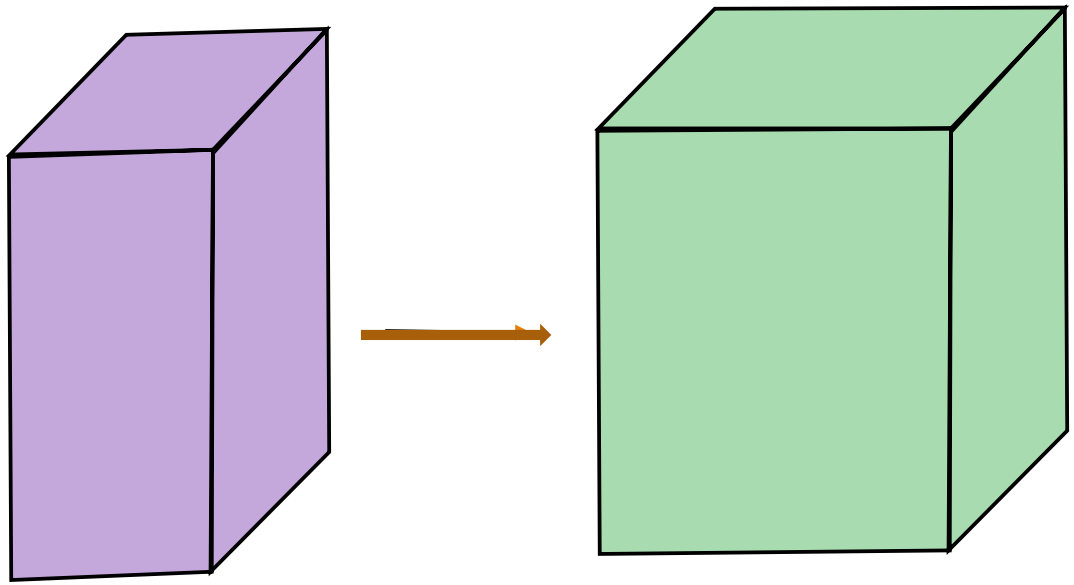
Image source : Convolutional neural network - Wikip

Convolution



32X 32 X 3 —5 X 5 X 20→ 28 X 28 X 20 —3 X 3 X 50→ 26 X 26 X 50

Special cases 1X1 filters



28 X 28 X 20

1 X 1 X 50

28 X 28 X 50

You can think about these filters are changing the volume size but keeping the width and height unchanged

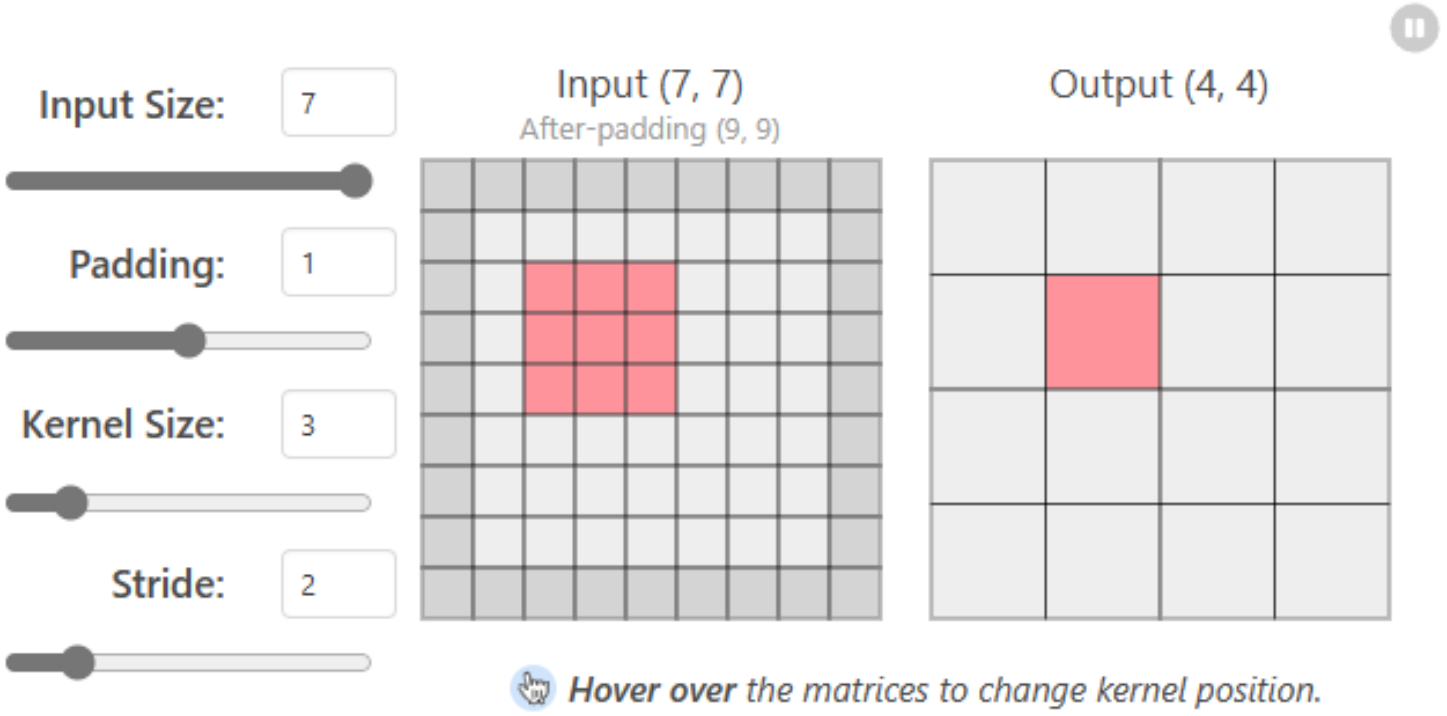
Padding and stride

Padding is s technique used to prevent the image height and width from shrinking as we apply convolution operations. It works by “padding zeros to the boundary of the image”

Stride determines how exactly do we walk when over the image as we convolve the filter With the image. Let's explore these notions interactively!

Convolution, padding and other concepts interactive tour

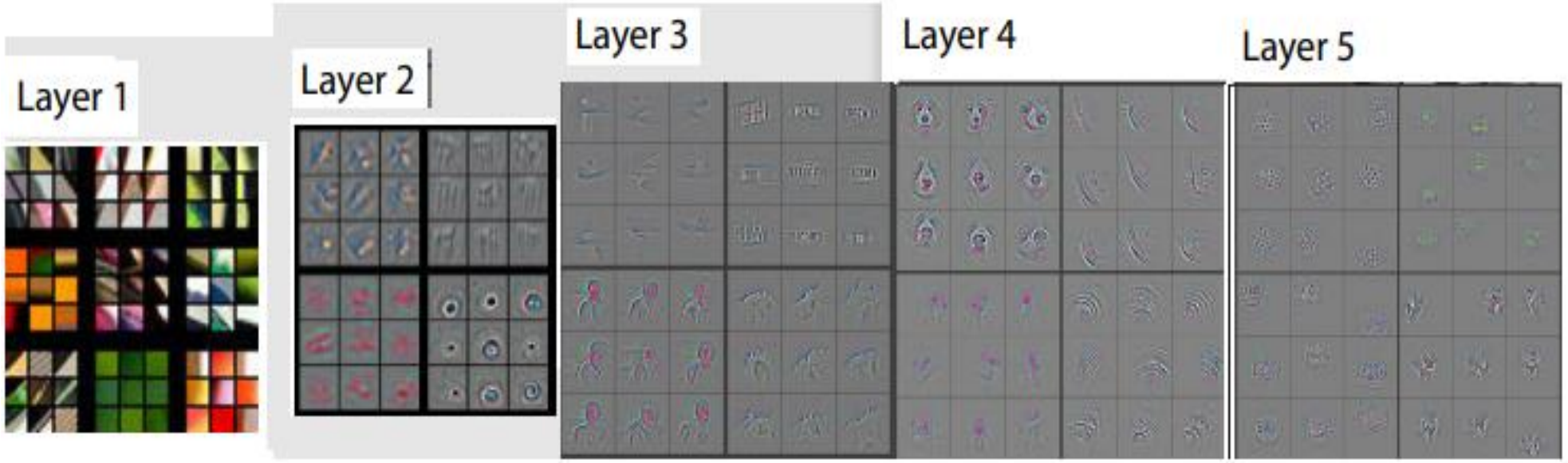
Lets explore this interactively!



[CNN Explainer (poloclub.github.io)]

Convolution

What do the filters learn ?



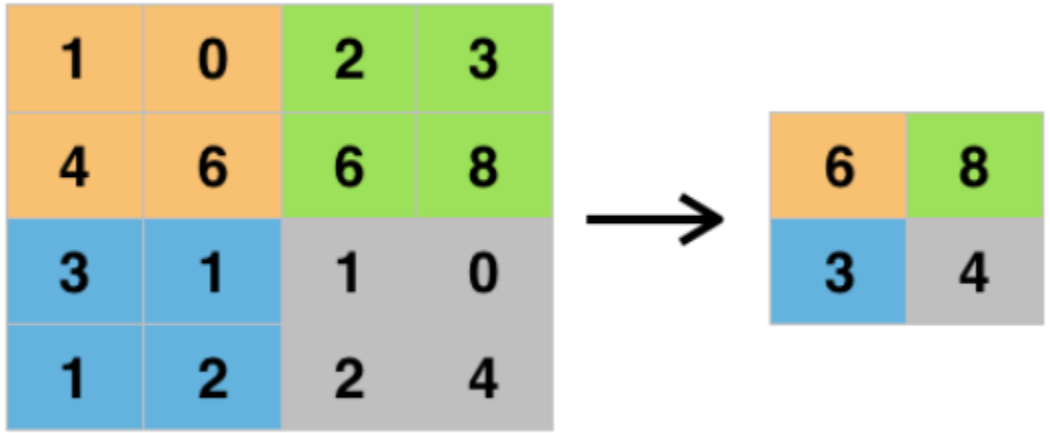
The learned representations refer to the hierarchical and abstract features that the network extracts from the input data during the training process. These representations are obtained by applying convolutional filters and pooling operations throughout the network's layers.

The learned representations in a CNN capture increasingly complex and abstract features from the input data. The earlier layers tend to learn low-level features, such as edges and textures, while deeper layers learn more high-level features that are specific to the task the network was trained on.

Source : Visualizing and Understanding Convolutional Networks (arxiv.org)

Pooling

Pooling in deep learning is a technique that reduces the dimensionality of data by summarizing or **downsampling** it, aiding in capturing important features while reducing computational complexity.



Max pooling

Image source : Convolutional neural network - Wikipedia

Why CNNs work really well in practice?

Inductive bias

- Inductive bias is the set of assumptions (or biases) that a machine learning algorithm makes to generalize from limited training data and make predictions about unseen examples.

- An example of inductive bias is a machine learning algorithm assuming that simpler explanations or hypotheses are more likely to be correct than complex ones.

Inductive bias in CNNs

- In the context of Convolutional Neural Networks (CNNs), the inductive bias refers to the assumptions and design choices made in the network architecture that enable it to effectively learn and extract features from visual data.

Inductive bias in CNNs

- In the context of Convolutional Neural Networks (CNNs), the inductive bias refers to the assumptions and design choices made in the network architecture that enable it to effectively learn and extract features from visual data.

- For example, CNNs have a built-in inductive bias that assumes spatial locality and translation invariance. This means that the network assumes that nearby pixels in an image are more likely to be related and that the learned features should be applicable regardless of their position in the image. By incorporating convolutional layers and pooling operations, CNNs can exploit these assumptions and efficiently capture local patterns and hierarchical representations in images.

Inductive bias in CNNs

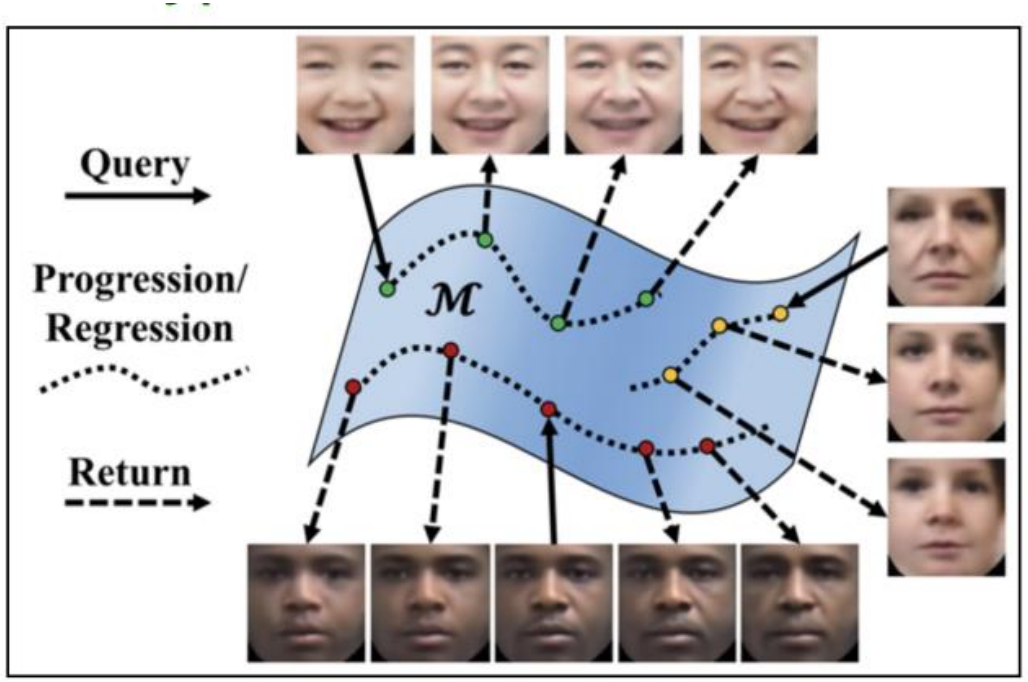
- In the context of Convolutional Neural Networks (CNNs), the inductive bias refers to the assumptions and design choices made in the network architecture that enable it to effectively learn and extract features from visual data.

- For example, CNNs have a built-in inductive bias that assumes spatial locality and translation invariance. This means that the network assumes that nearby pixels in an image are more likely to be related and that the learned features should be applicable regardless of their position in the image. By incorporating convolutional layers and pooling operations, CNNs can exploit these assumptions and efficiently capture local patterns and hierarchical representations in images.

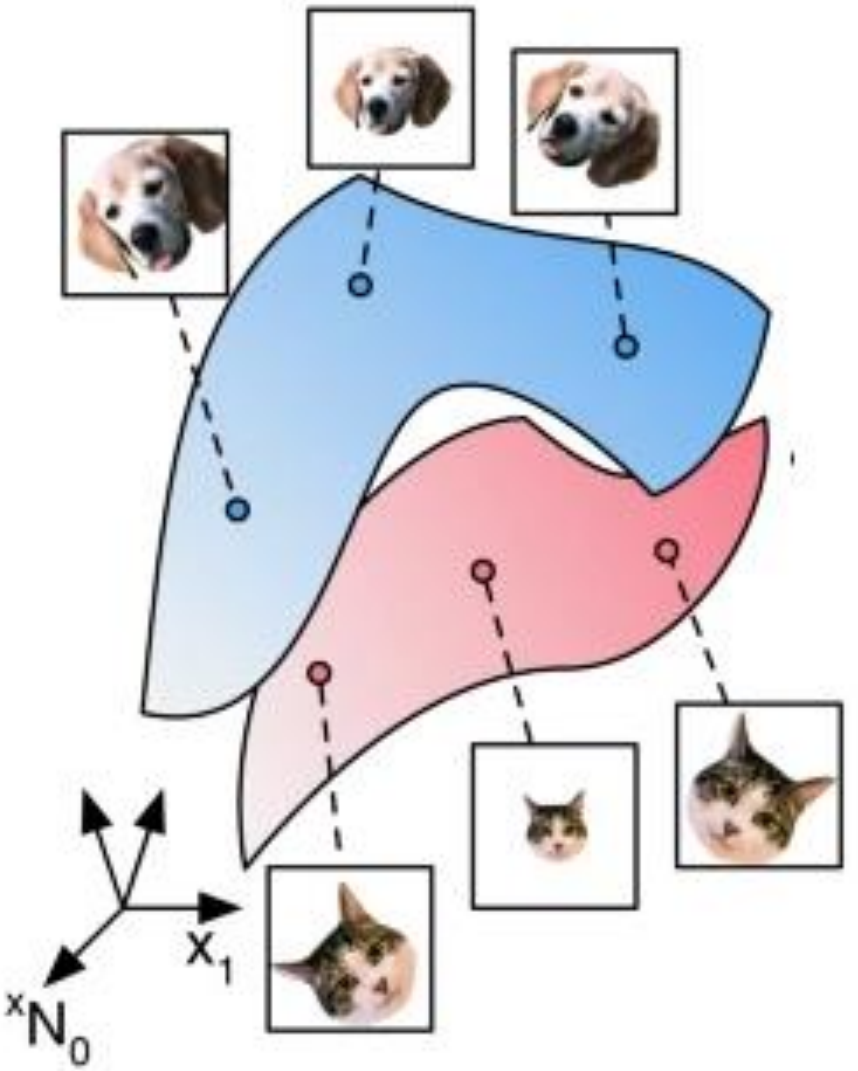
- The inductive bias of CNNs helps them excel in tasks such as image classification, object detection, and image segmentation, where spatial relationships and local patterns play a crucial role.

The Manifold Hypothesis

The manifold hypothesis suggests that real-world data, even in high-dimensional spaces (number of pixels), often concentrates around a lower-dimensional manifold embedded within that space. NNs have the ability to learn and model complex mappings from the input space to the output space, which includes capturing the intricate relationships and patterns present in the data.



14.3 Learning Manifolds (buffalo.edu)

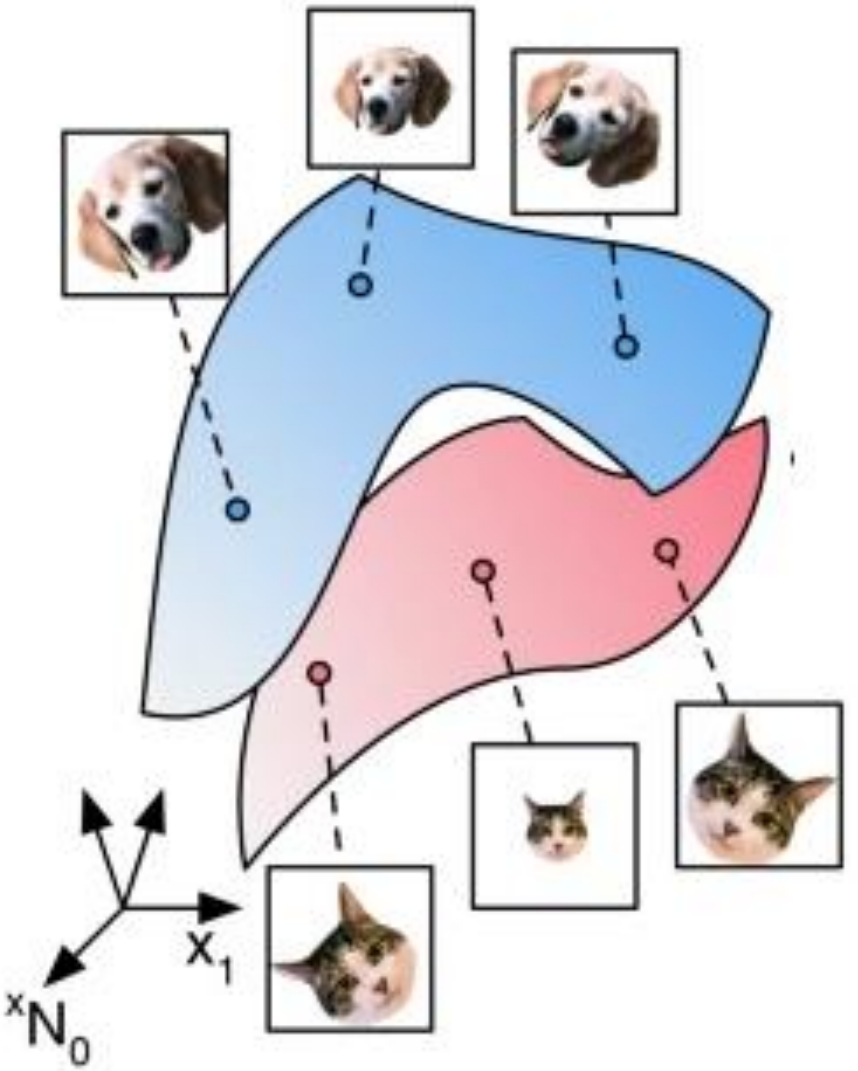


Separability and geometry of object manifolds in deep neural networks | Nature Communications

The Manifold Hypothesis

The manifold hypothesis suggests that real-world data, even in high-dimensional spaces (number of pixels), often concentrates around a lower-dimensional manifold embedded within that space. NNs have the ability to learn and model complex mappings from the input space to the output space, which includes capturing the intricate relationships and patterns present in the data.

Neural Networks work effectively because they are capable of learning and approximating the underlying structure of high-dimensional data that lies on or near a lower-dimensional manifold.



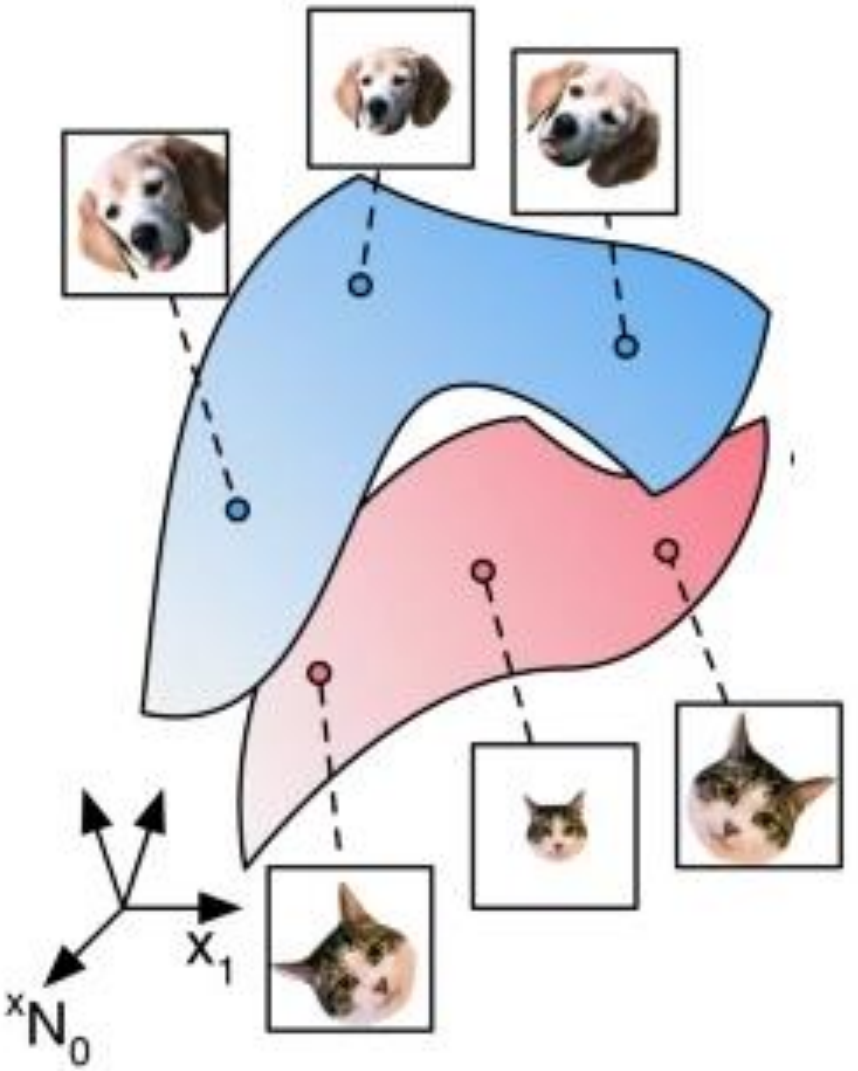
[Separability and geometry of object manifolds in deep neural networks | Nature Communications]

The Manifold Hypothesis

The manifold hypothesis suggests that real-world data, even in high-dimensional spaces (number of pixels), often concentrates around a lower-dimensional manifold embedded within that space. NNs have the ability to learn and model complex mappings from the input space to the output space, which includes capturing the intricate relationships and patterns present in the data.

Neural Networks work effectively because they are capable of learning and approximating the underlying structure of high-dimensional data that lies on or near a lower-dimensional manifold.

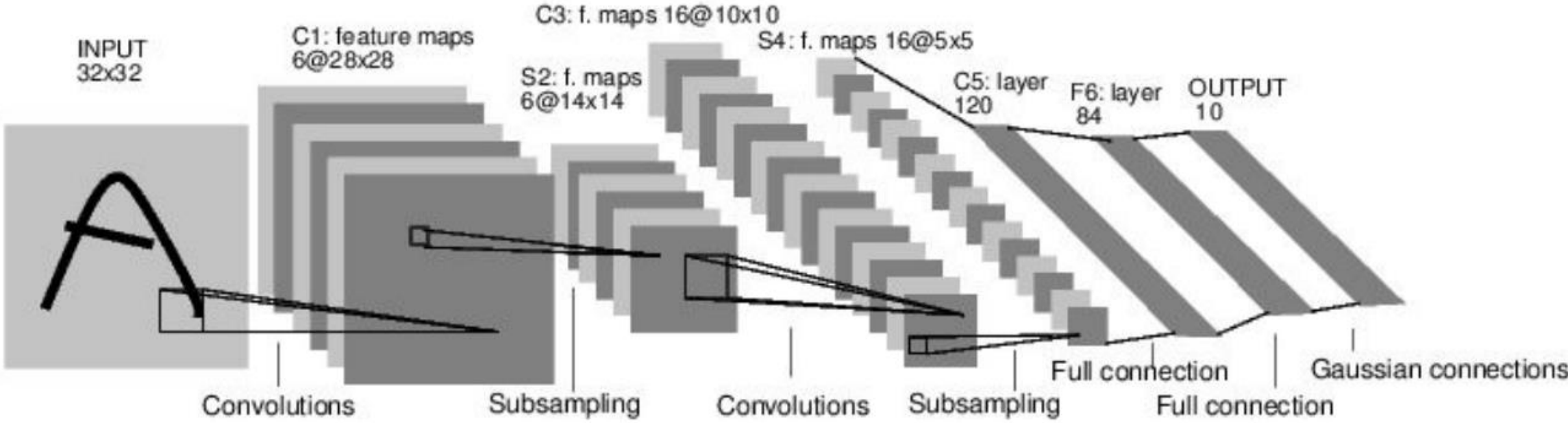
Through a process called backpropagation, NNs adjust their weights and biases during training to minimize the discrepancy between the predicted outputs and the ground truth labels. This optimization process helps the network align its learned representations with the underlying manifold, enabling it to generalize well and make accurate predictions on unseen examples.



Separability and geometry of object manifolds in deep neural networks | Nature Communications

Case study : LeNet

[LeCun et al., 1998]

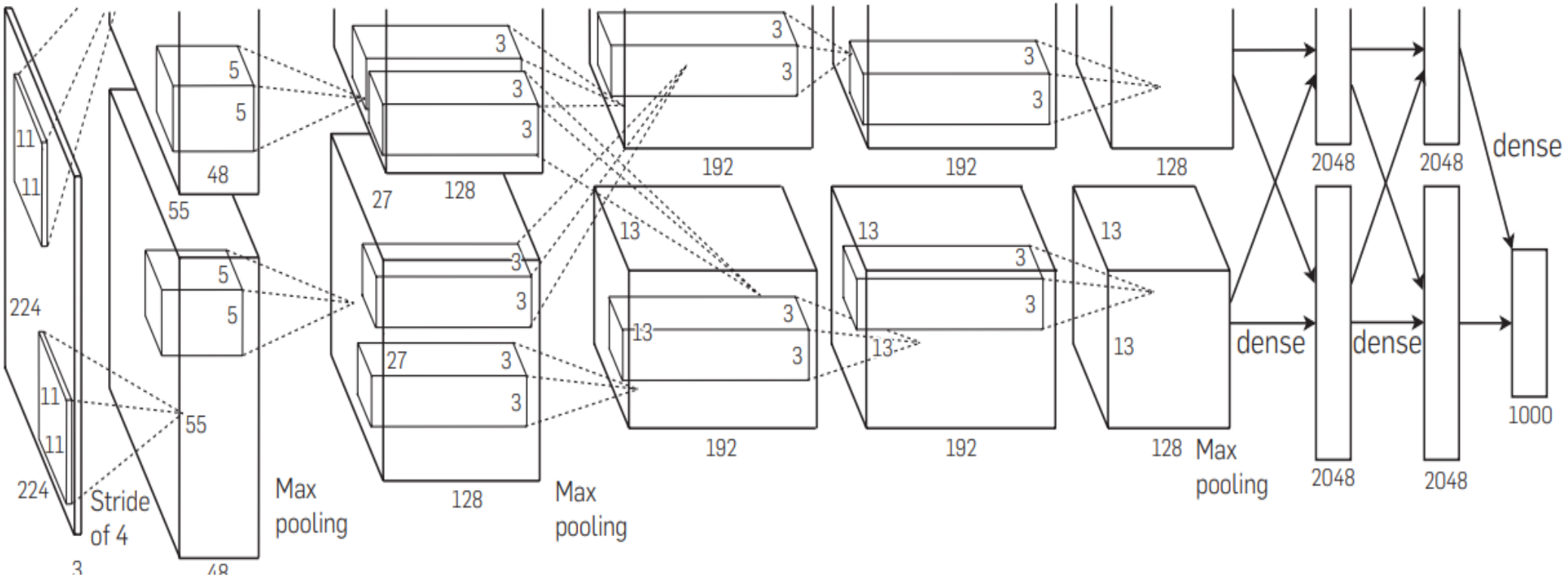


Conv filters were 5x5, applied at stride 1 Subsampling (Pooling) layers were 2x2 applied at stride

architecture is [CONV-POOL-CONV-POOL-CONV-FC] : almost standard nowdays in CNN

Case study : AlexNet

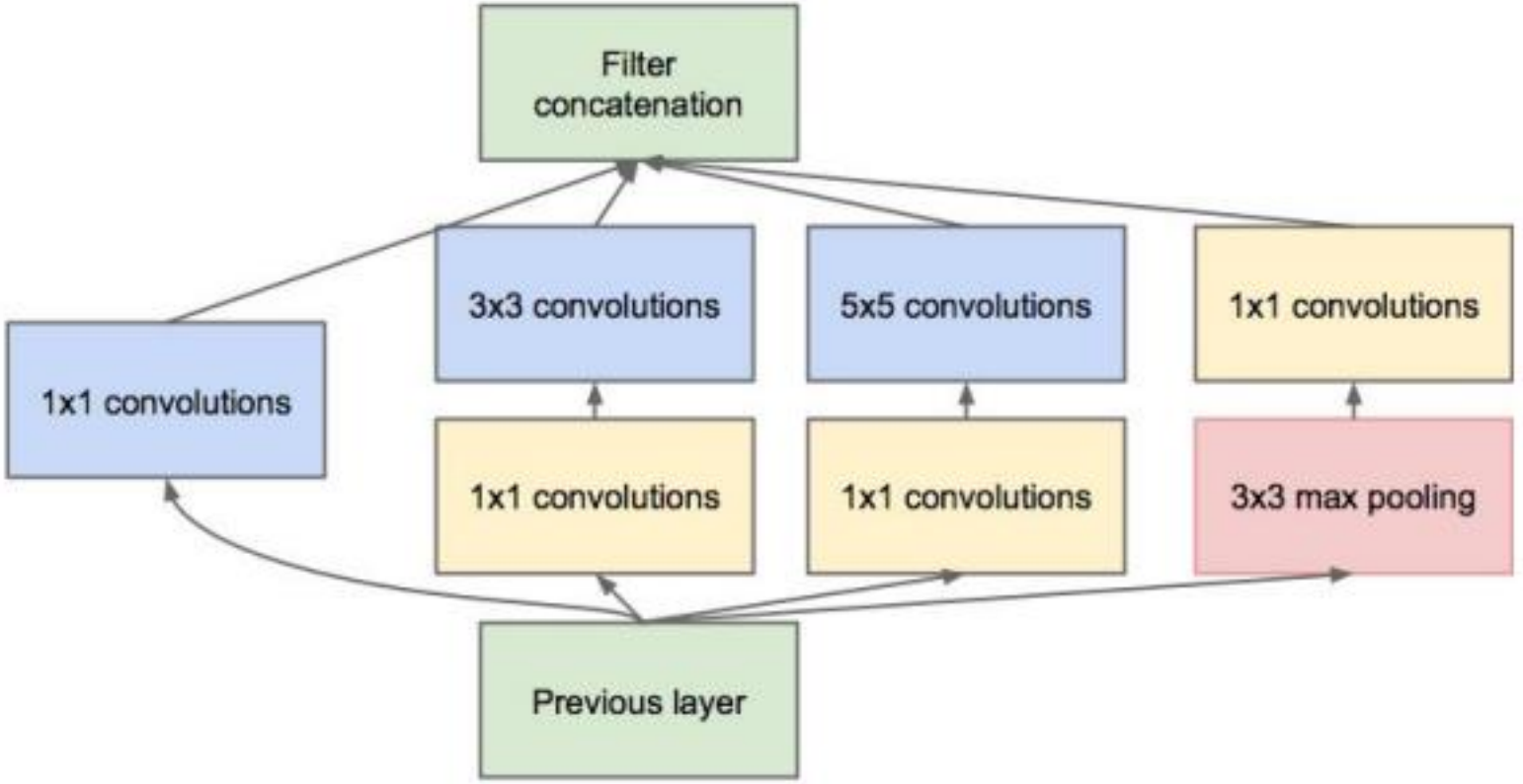
The new arrival of modern DL :



Input: 227x227x3
After CONV1: 55x55x96
After POOL1: 27x27x96

[ImageNet classification with deep convolutional neural networks (acm.org)]

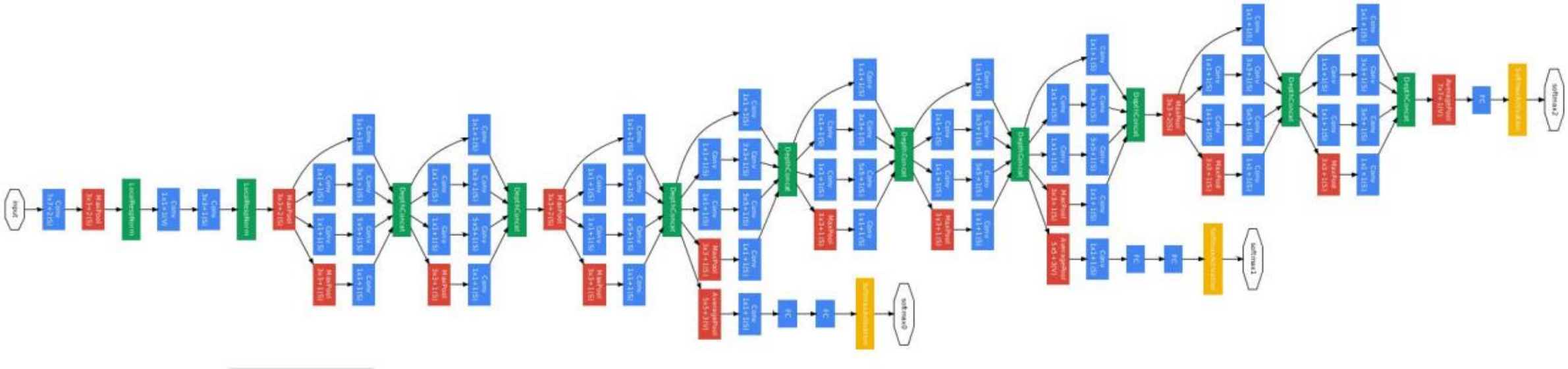
Case study : GoogleNet



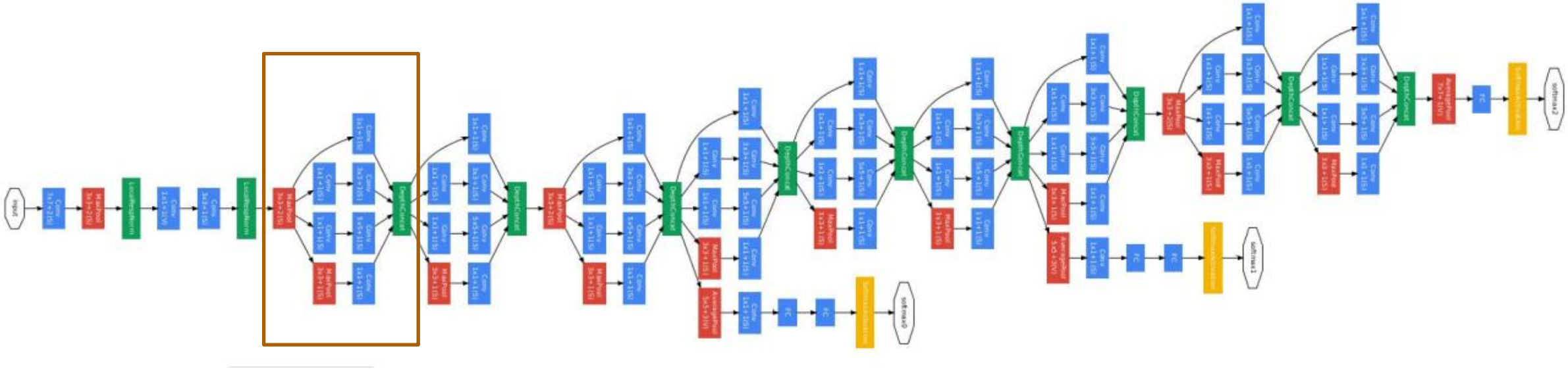
Inception module

Main idea: apply filters with different size on the input image and then concatenate all of them together

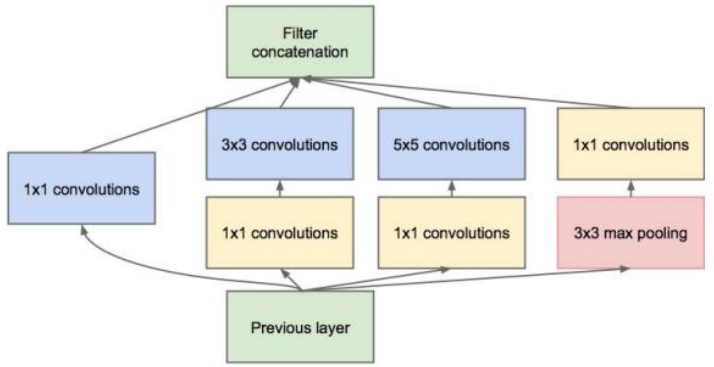
Case study : GoogleNet



Case study : GoogleNet



Although it looks complicated : it repeats the inception module multiple times



The Vanishing Gradient Problem

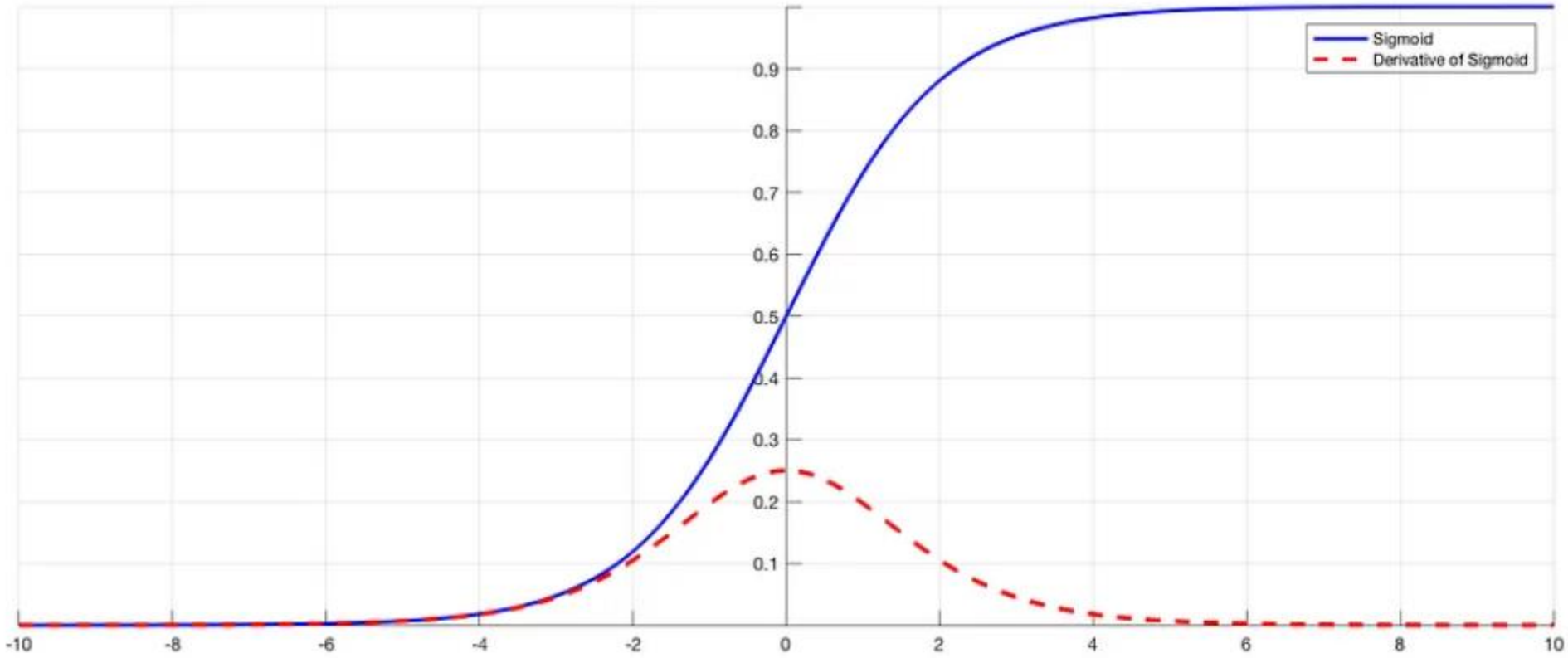


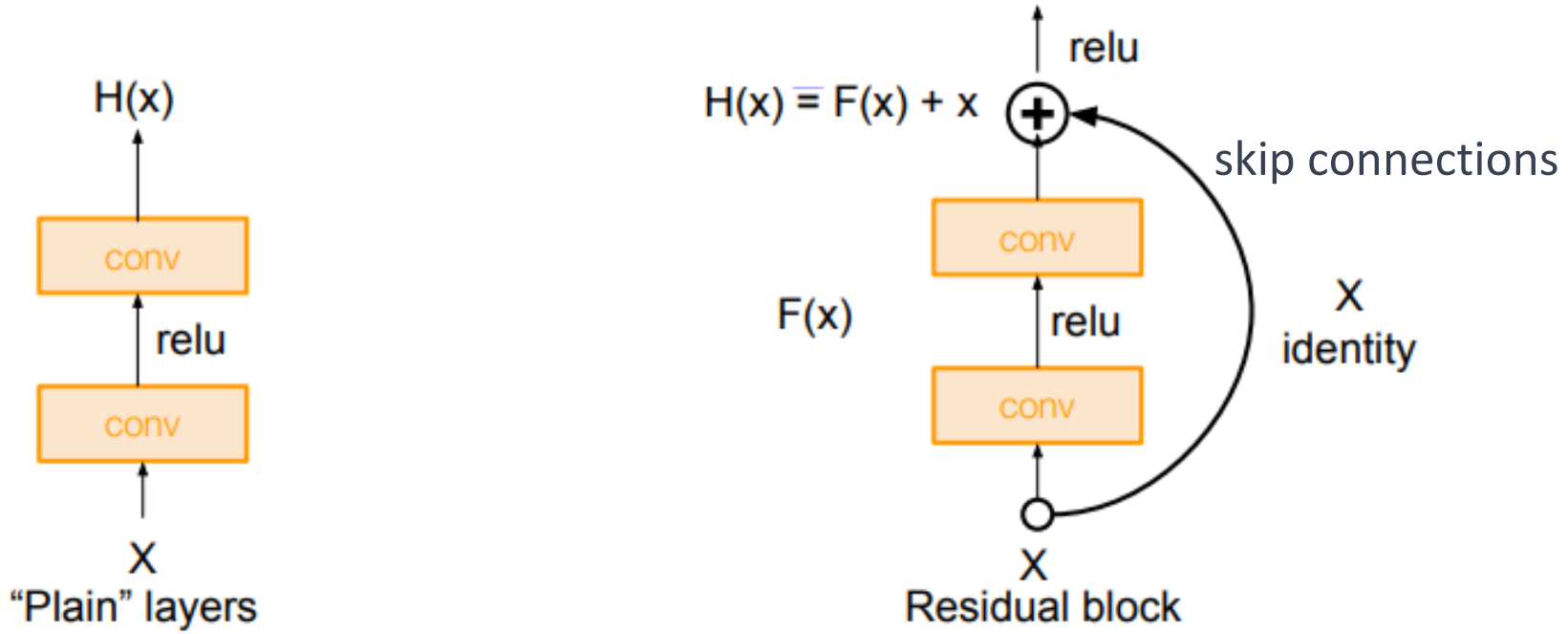
Image 1: The sigmoid function and its derivative // Source

Activation functions such as the sigmoid function compress a wide input range into a narrow output range, causing a small change in output for a large input change. As a result, the derivative becomes small.

The Vanishing Gradient Problem

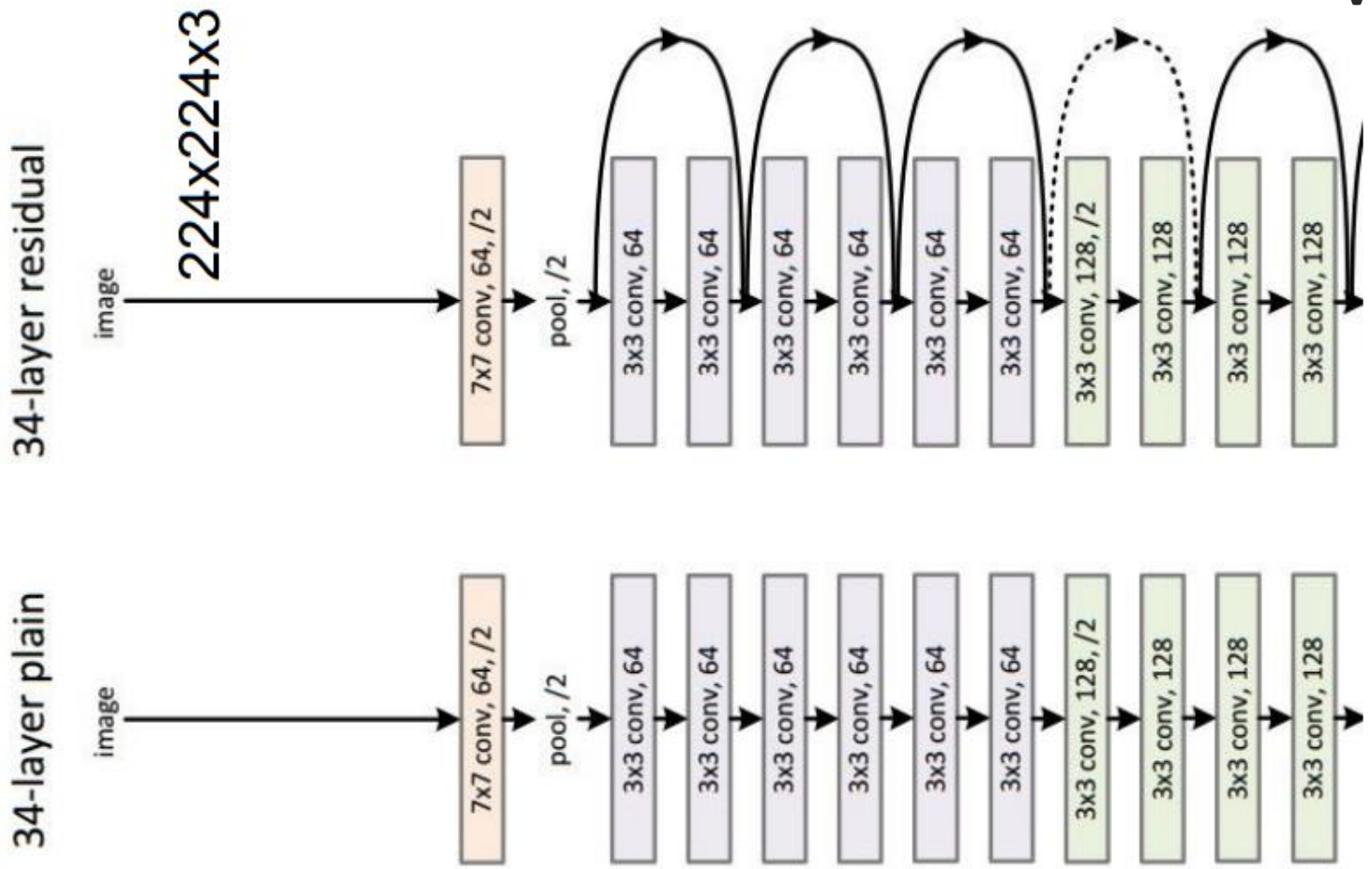
- Gradients of neural networks are found using backpropagation.
- Simply put, backpropagation finds the derivatives of the network by moving layer by layer from the final layer to the initial one.
- By the chain rule, the derivatives of each layer are multiplied down the network (from the final layer to the initial) to compute the derivatives of the initial layers.
- However, when *n* hidden layers use an activation like the sigmoid function, *n* small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers.
- A small gradient means that the weights and biases of the initial layers will not be updated effectively with each training session. Since these initial layers are often crucial to recognizing the core elements of the input data, it can lead to overall inaccuracy of the whole network.

Case study : RezNet



The residual connection directly adds the value at the beginning of the block, **x**, to the end of the block (F(x)+x). This residual connection doesn't go through activation functions that "squashes" the derivatives, resulting in a higher overall derivative of the block.

Case study : ResNet



ResNets allow training very deep neural networks

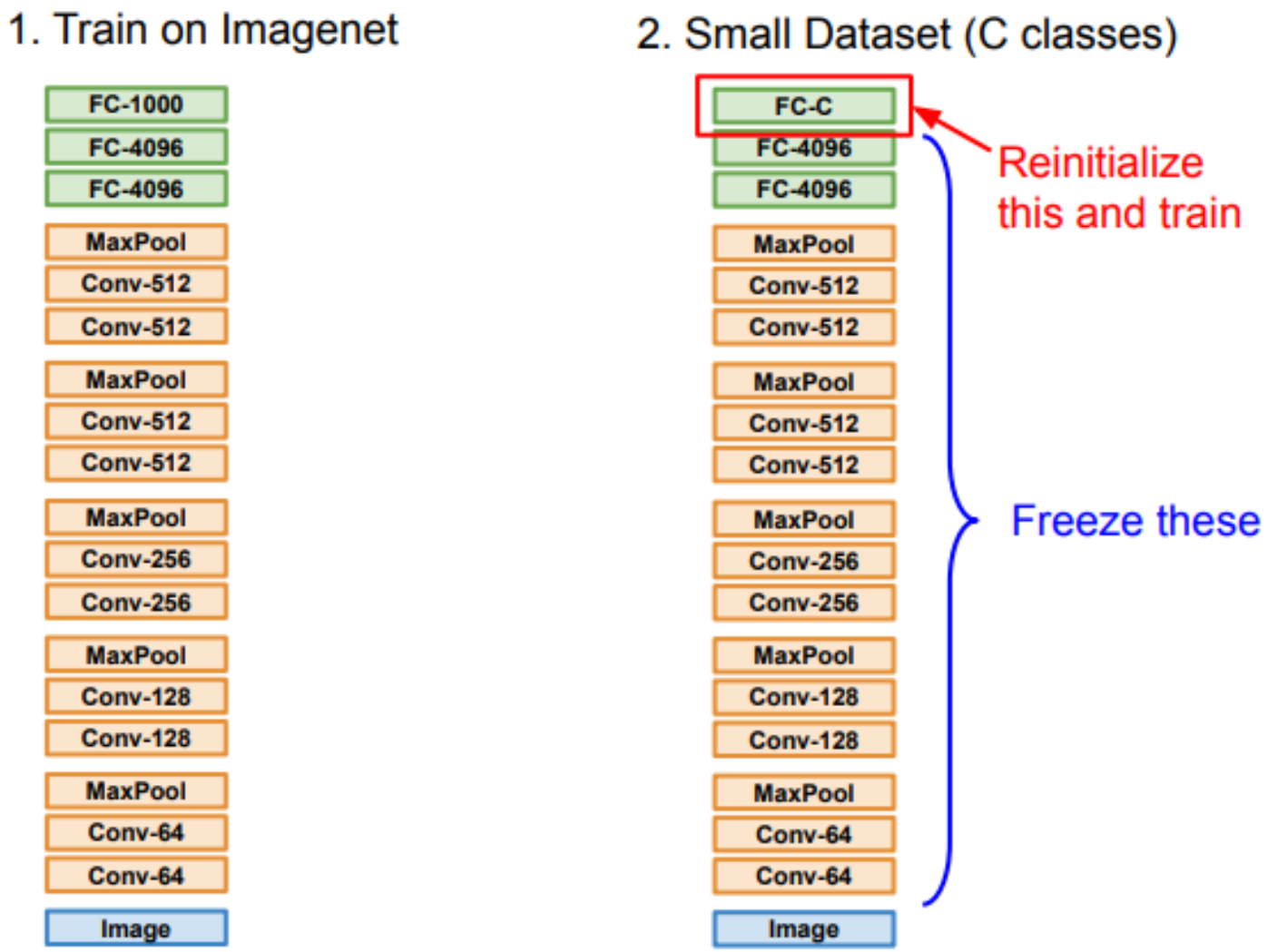
Transfer Learning

•Transfer learning is a technique in machine learning where knowledge gained from training on one task is leveraged to improve performance on a different but related task.

•It involves using pre-trained models, which are neural networks trained on large-scale datasets, as a starting point for a new task.

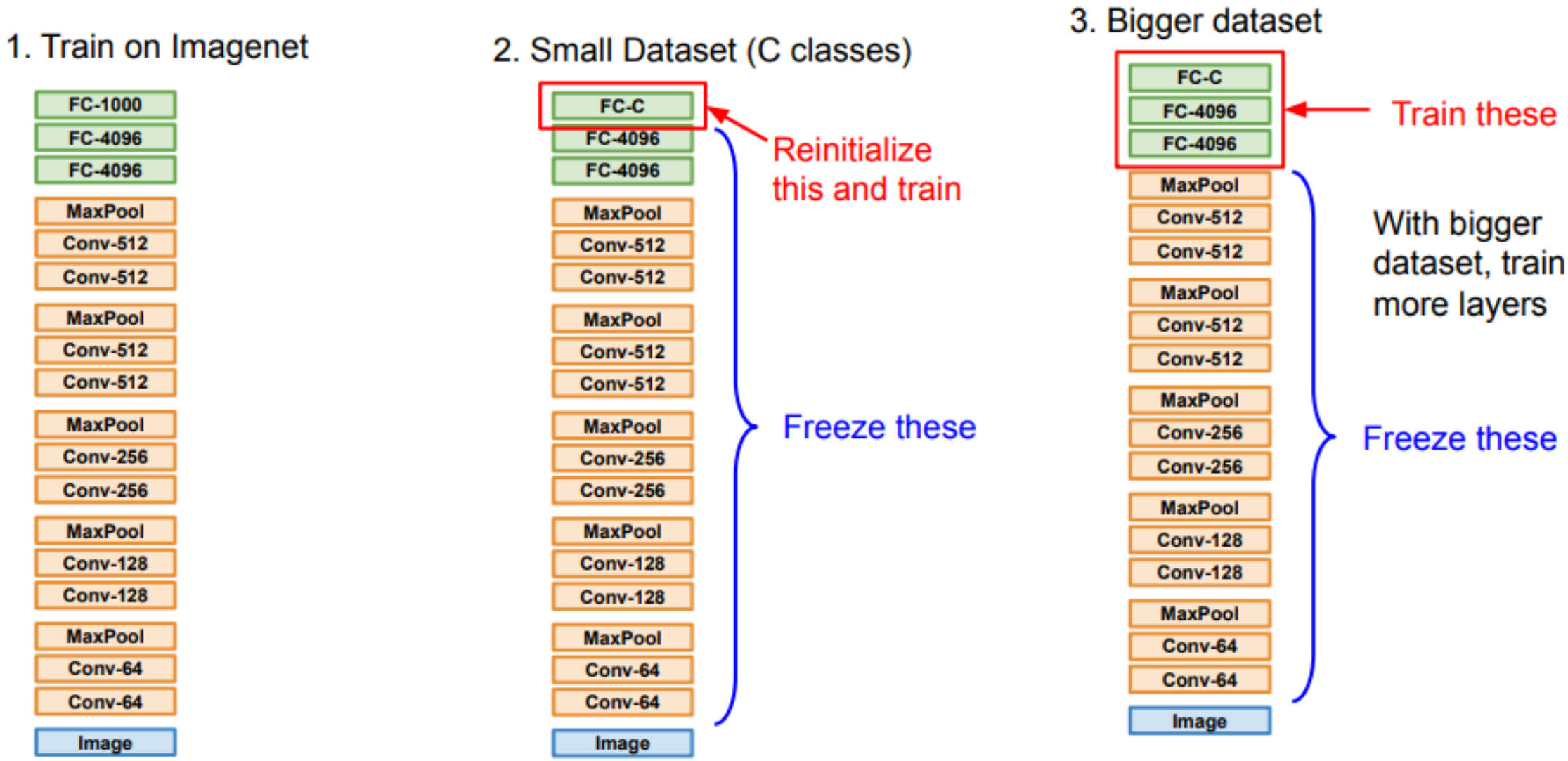
•By utilizing the learned representations and knowledge from the pre-trained model, transfer learning allows for faster convergence, better generalization, and improved performance, especially when the new task has limited data.

Transfer Learning



Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014 Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

Transfer Learning



Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014 Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

Refs

The Vanishing Gradient Problem. The Problem, Its Causes, Its… | by Chi-Feng Wang | Towards Data Science

Lecture 5.pptx (stanford.edu)

Lecture 6.pptx (stanford.edu)

winter1516_lecture7.pdf (stanford.edu)

14.3 Learning Manifolds (buffalo.edu)