Deep Learning

BCSE-332L

Module 3:

Convolutional Neural Network

Dr . Saurabh Agrawal

Faculty Id: 20165

School of Computer Science and Engineering

VIT, Vellore-632014

Tamil Nadu, India

Outline

- Foundations of Convolutional Neural Networks
- CNN Operations
- Architecture
- Simple Convolution Network
- Deep Convolutional Models
  - ResNet
  - AlexNet
  - InceptionNet
  - Others

Foundations of Convolutional Neural Networks

❑What is Convolutional Neural Network(CNN)?

❑A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks.

❑It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

❑The architecture of CNNs is inspired by the visual processing in the human brain, and they are well-suited for capturing hierarchical patterns and spatial dependencies within images.

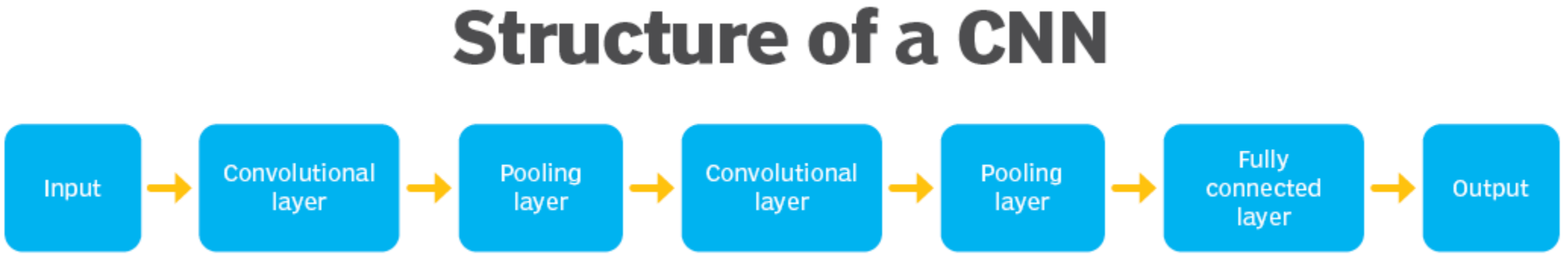
Foundations of Convolutional Neural Networks

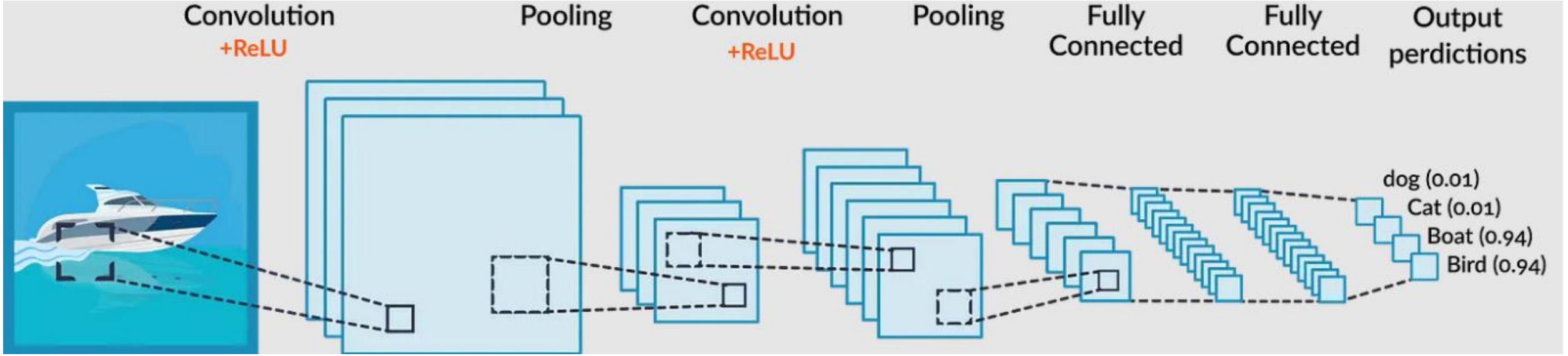
❑Key components of a Convolutional Neural Network include:

1. **Convolutional Layers:** These layers apply convolutional operations to input images, using filters (also known as kernels) to detect features such as edges, textures, and more complex patterns. Convolutional operations help preserve the spatial relationships between pixels.
2. **Pooling Layers:** Pooling layers downsample the spatial dimensions of the input, reducing the computational complexity and the number of parameters in the network. Max pooling is a common pooling operation, selecting the maximum value from a group of neighboring pixels.
3. **Activation Functions:** Non-linear activation functions, such as Rectified Linear Unit (ReLU), introduce non-linearity to the model, allowing it to learn more complex relationships in the data.
4. **Fully Connected Layers:** These layers are responsible for making predictions based on the high-level features learned by the previous layers. They connect every neuron in one layer to every neuron in the next layer.

Foundations of Convolutional Neural Networks

❑Key components of a Convolutional Neural Network include:





Foundations of Convolutional Neural Networks

- CNNs are trained using a large dataset of labeled images, where the network learns to recognize patterns and features that are associated with specific objects or classes.
- Proven to be highly effective in image-related tasks, achieving state-of-the-art performance in various computer vision applications.
- Their ability to automatically learn hierarchical representations of features makes them well-suited for tasks where the spatial relationships and patterns in the data are crucial for accurate predictions.
- CNNs are widely used in areas such as image classification, object detection, facial recognition, and medical image analysis.
- The convolutional layers are the key component of a CNN, where filters are applied to the input image to extract features such as edges, textures, and shapes.
- The output of the convolutional layers is then passed through pooling layers, which are used to down-sample the feature maps, reducing the spatial dimensions while retaining the most important information.
- The output of the pooling layers is then passed through one or more fully connected layers, which are used to make a prediction or classify the image.

Foundations of Convolutional Neural Networks

❑Convolutional Neural Network Design

❑The construction of a convolutional neural network is a multi-layered feed-forward neural network, made by assembling many unseen layers on top of each other in a particular order.

❑It is the sequential design that give permission to CNN to learn hierarchical attributes.

❑In CNN, some of them followed by grouping layers and hidden layers are typically convolutional layers followed by activation layers.

❑The pre-processing needed in a ConvNet is kindred to that of the related pattern of neurons in the human brain and was motivated by the organization of the Visual Cortex.

Foundations of Convolutional Neural Networks

❑**Convolutional Neural Network Training** :CNNs are trained using a supervised learning approach.

❑This means that the CNN is given a set of labeled training images. The CNN then learns to map the input images to their correct labels.

❑The training process for a CNN involves the following steps:

1. **Data Preparation:** The training images are preprocessed to ensure that they are all in the same format and size.
2. **Loss Function:** A loss function is used to measure how well the CNN is performing on the training data. The loss function is typically calculated by taking the difference between the predicted labels and the actual labels of the training images.
3. **Optimizer:** An optimizer is used to update the weights of the CNN in order to minimize the loss function.
4. **Backpropagation:** Backpropagation is a technique used to calculate the gradients of the loss function with respect to the weights of the CNN. The gradients are then used to update the weights of the CNN using the optimizer.

Foundations of Convolutional Neural Networks

❑**CNN Evaluation:** After training, CNN can be evaluated on a held-out test set.

❑A collection of pictures that the CNN has not seen during training makes up the test set.

❑How well the CNN performs on the test set is a good predictor of how well it will function on actual data.

❑The efficiency of a CNN on picture categorization tasks can be evaluated using a variety of criteria.

❑Among the most popular metrics are:

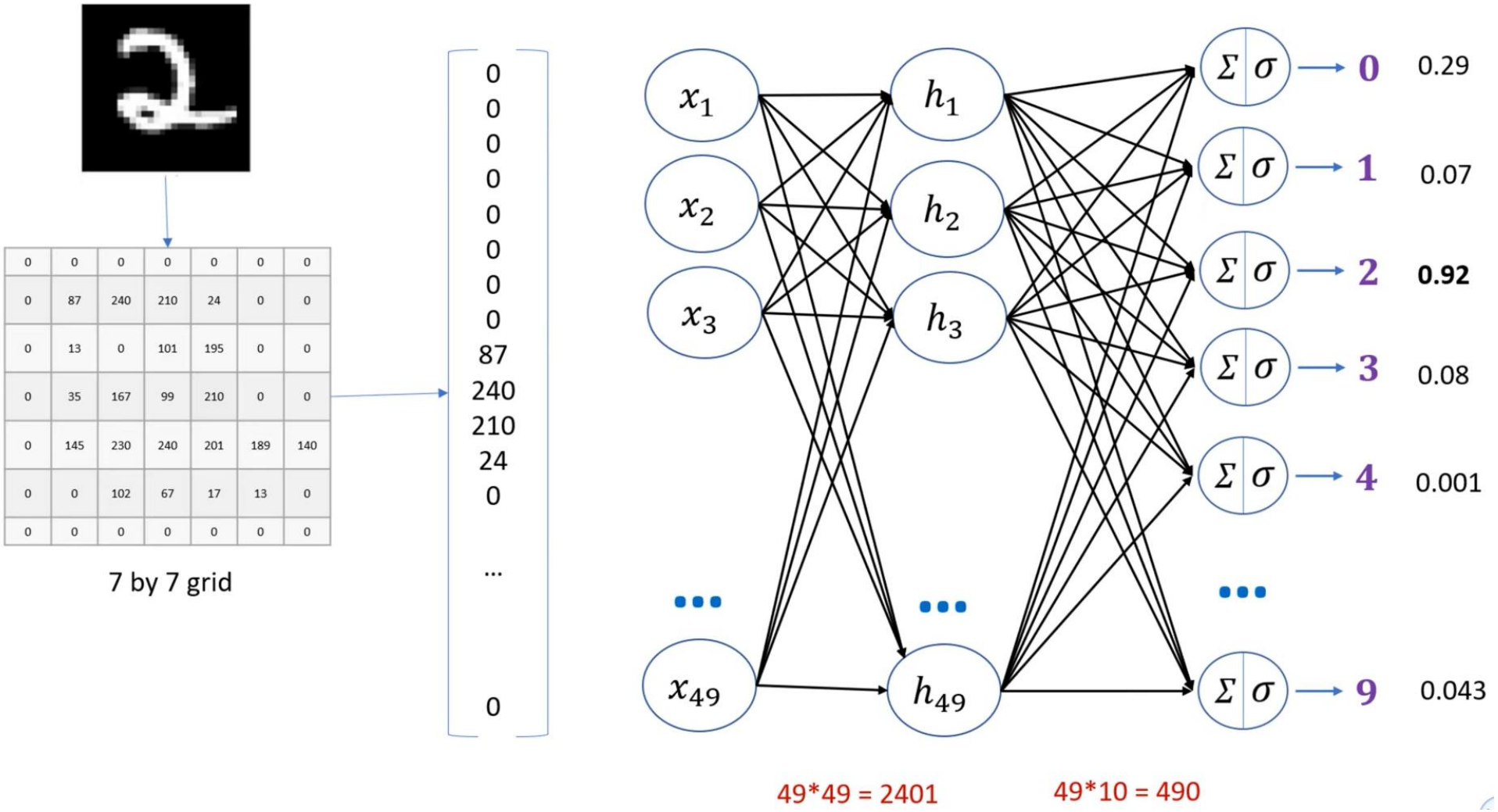
1. **Accuracy:** Accuracy is the percentage of test images that the CNN correctly classifies.
2. **Precision:** Precision is the percentage of test images that the CNN predicts as a particular class and that are actually of that class.
3. **Recall:** Recall is the percentage of test images that are of a particular class and that the CNN predicts as that class.
4. **F1 Score:** The F1 Score is a harmonic mean of precision and recall. It is a good metric for evaluating the performance of a CNN on classes that are imbalanced.

❑Why Convolutional Neural Networks?

To handle variety in digits we can use simple artificial neural network (ANN)

Foundations of Convolutional Neural Networks

❑Why Convolutional Neural Networks?



Foundations of Convolutional Neural Networks

❑Why Convolutional Neural Networks?

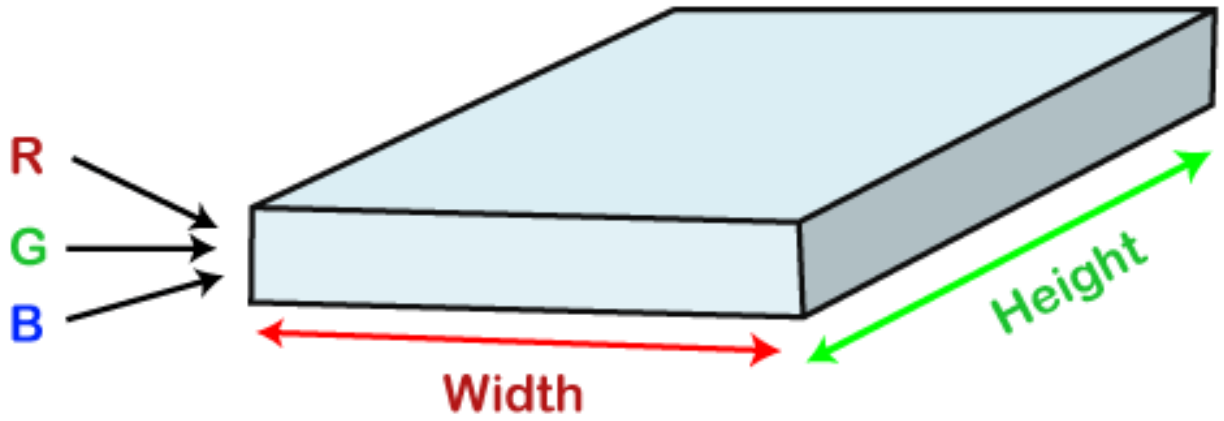
Image size = 1920 x 1080 X 3

First layer neurons = 1920 x 1080 X 3 ~ 6 million

Hidden layer neurons = Let's say you keep it ~ 4 million

Weights between input and hidden layer = 6 mil * 4 mil
= 24 million

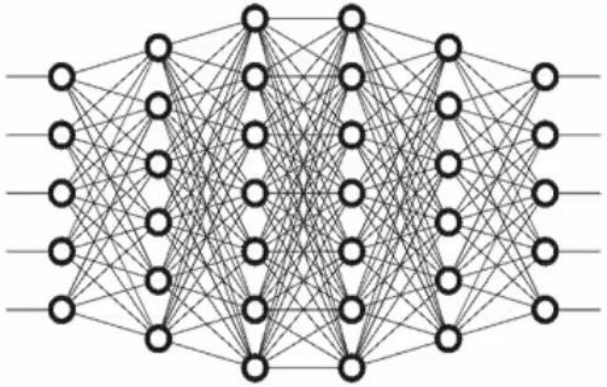
❑Why Convolutional Neural Networks?



❑Why Convolutional Neural Networks?

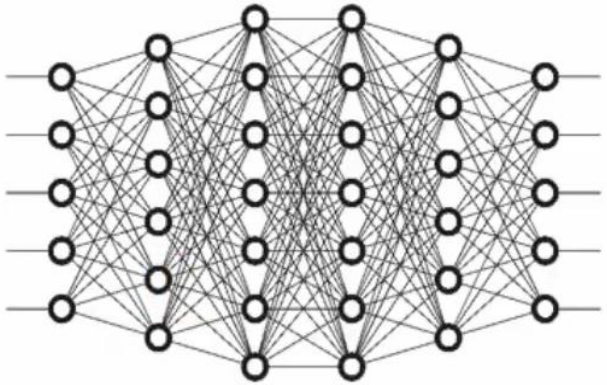
Why Not Fully Connected Networks

Image with 28 x 28 x 3 pixels



Number of weights in the first hidden layer will be 2352

Image with 200 x 200 x 3 pixels



Number of weights in the first hidden layer will be 120,000

CNN Operations

- Convolutional Neural Network is the type of neural network which is used for analysis of visual inputs.
- Some popular applications of CNN includes image classification, image recognition, Natural language processing etc.
- Now, there are certain steps/operations that are involved CNN, these can be categorized as follows:

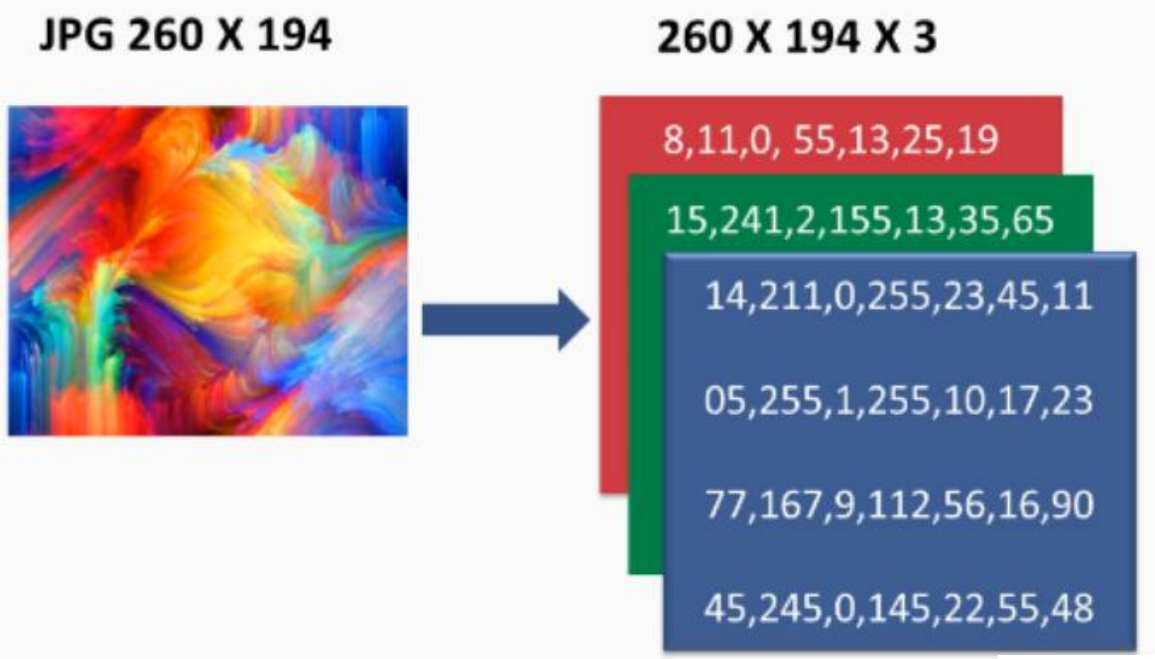
1. **Convolution operation**
2. **Pooling**
3. **Padding**
4. **Flattening**
5. **Fully connected layers**

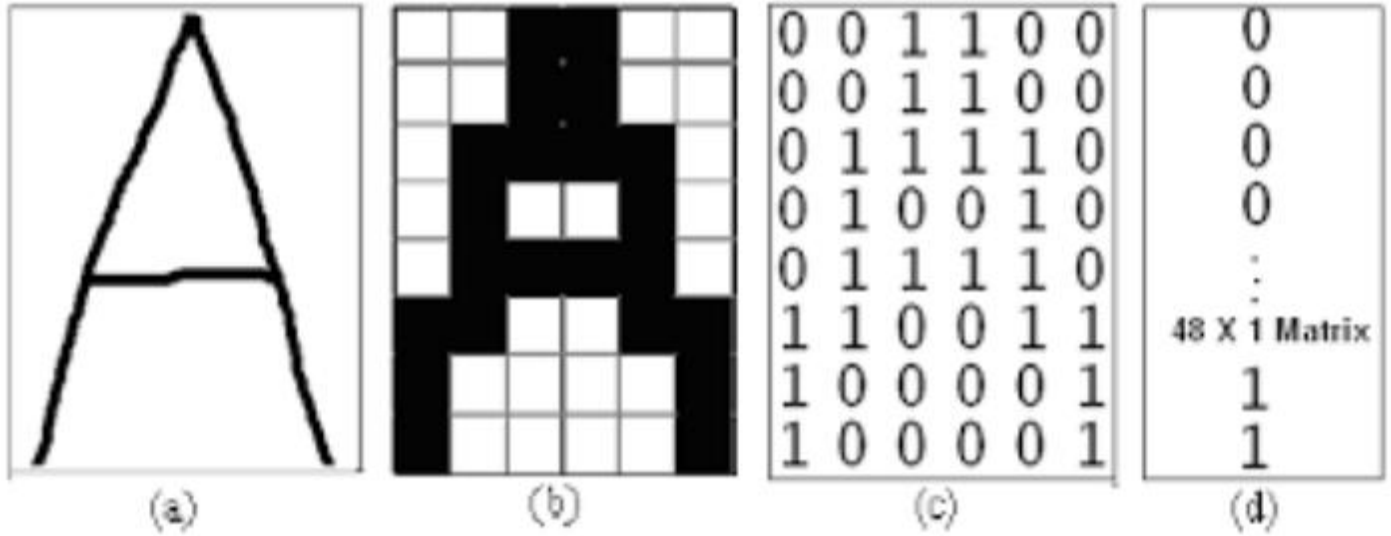
CNN Operations

- **CONVOLUTION OPERATION:** Convolution operations is the first and one of the most important step in the functioning of a CNN.
- Convolution operation focuses on extracting/preserving important features from the input (image etc).
- To understand this operation, let us consider image as input to our CNN. Now when image is given as input, they are in the form of matrices of pixels.
- If the image is grayscale, then the image is considered as a matrix, each value in the matrix ranges from 0-255.
- We can even normalize these values lets say in range 0-1. 0 represents white and 1 represents black.
- If the image is colored, then three matrices representing the RGB colors with each value in range 0-255.
- The same can be seen in the images in the next page:

CNN Operations

❑CONVOLUTION OPERATION:





CNN Operations

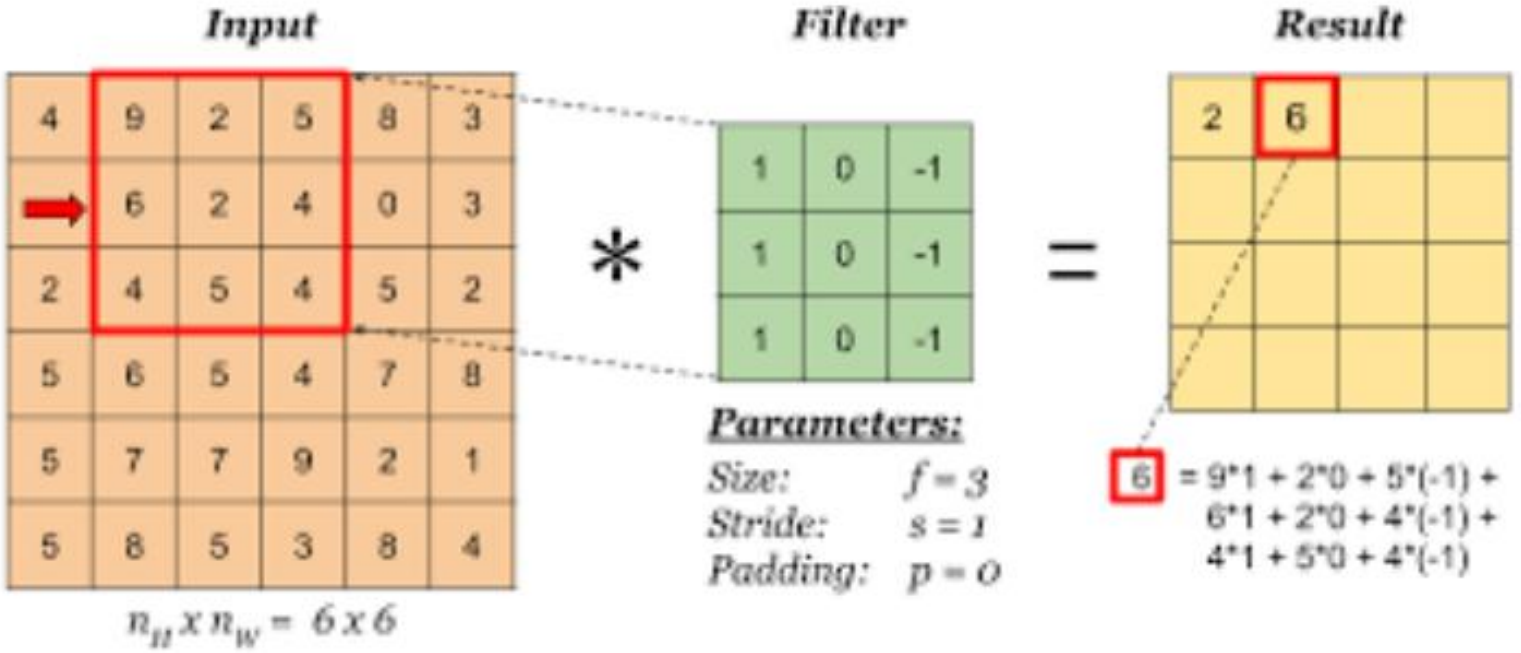
❑**CONVOLUTION OPERATION:** Consider the image given below:

❑Here, input of size 6x6 is given and kernel of size 3x3 is used.

❑The feature map obtained in of size 4x4.

❑To increase non-linearity in the image, Rectifier function can be applied to the feature map.

❑Finally, after the convolution step is completed and feature map is obtained, this map is given as input to the pooling layer.



CNN Operations

❑POOLING:

❑This step helps to maintain spatial invariance or even deal with other kind of distortions in the process.

❑This means that if we are trying to perform image recognition or something like checking if the image contains a dog, there is a possibility that the image might not be straight (maybe tilted), or texture difference is there, the object size in the image is small etc.

❑So, this should not let our model to provide incorrect output.

❑This is what pooling is all about. There can be different types of pooling like min pooling, max pooling etc.

❑It helps to preserve the essential features.

❑What we obtain is called pooled feature map. Here the size is reduced, features are preserved and distortions are dealt with.

CNN Operations

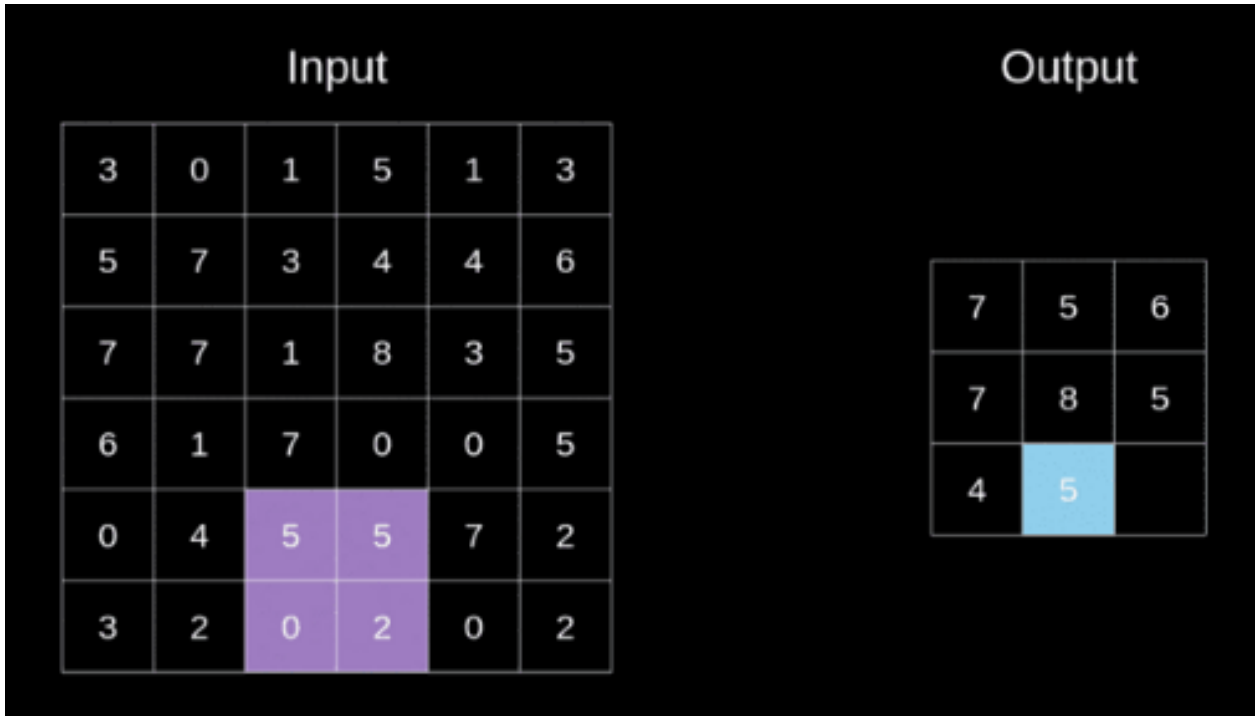
❑POOLING:

❑Mentioned below are some types if pooling that are used:

1. **Max Pooling**: In max pooling, the maximum value is taken from the group of values of patch feature map.
2. **Minimum Pooing**: In this type of pooling, the minimum value is taken from the patch in feature map.
3. **Average Pooling**: Here, the average of values is taken.
4. **Adaptive Pooling**: In this type of pooling, we only need to define the output size we need for the feature map. Parameters such as stride etc are automatically calculated.

CNN Operations

❑**POOLING:** Let is take an example to understand pooling better:



❑In the above image of size 6x6, we can see that on the feature map, max pooling is applied with stride 2 and filter 2 or 2x2 window.

❑This operation reduces the size of the data and preserves the most essential features. The output obtained after the pooling layer is called the pooled feature map.

CNN Operations

❑**Padding:** CNN has offered a lot of promising results but there are some issues that comes while applying convolution layers. There are two significant problems:

1. When we apply convolution operation, based on the size of image and filter, the size of the resultant processed image reduces according to the following rule:
    - ❑Let image size: nxn
    - ❑Let filer size: mxm
    - ❑Then, resultant image size: (n-m+1)x(n-m+1)
2. Another problem comes with the pixel values in the edges of the image matrix. The values on the edges of the matrix do not contribute as much as the values in the middle. Like if some pixel value resides in the corner i.e. (0,0) position, then it will be considered only once in the convolution operation, while values in the middle will be considered multiple times.

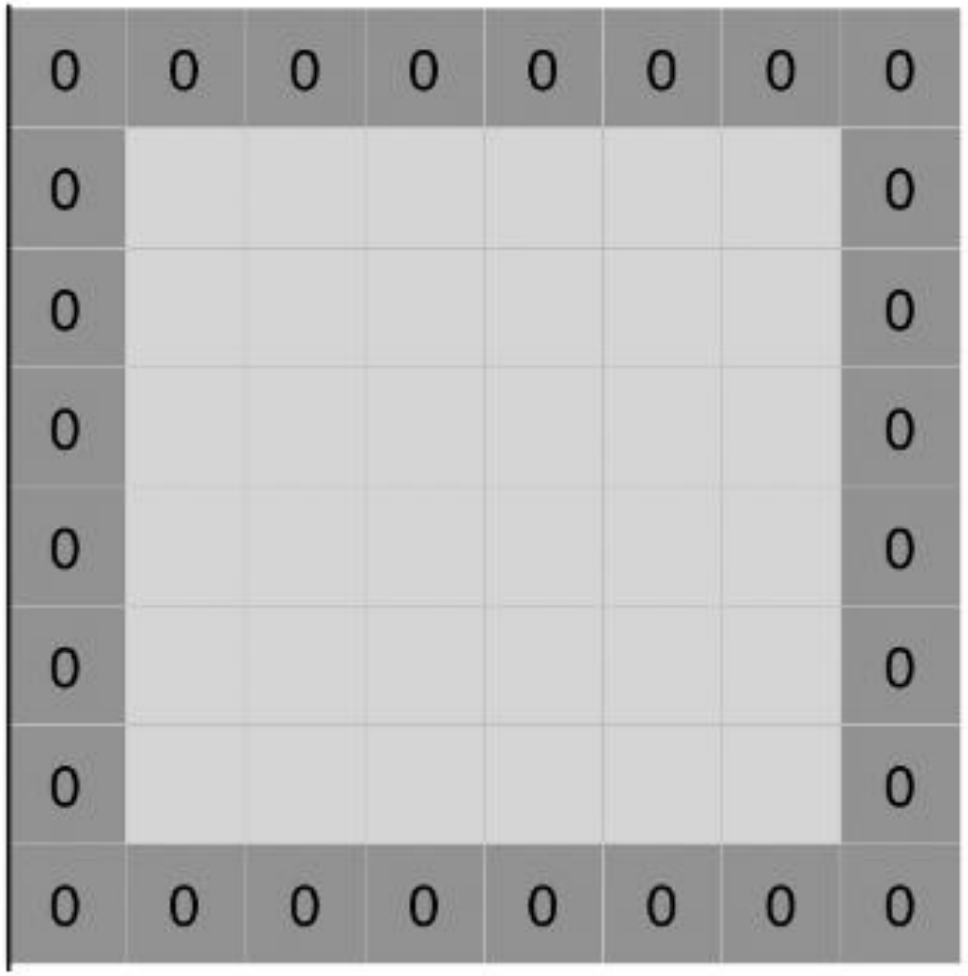
   To overcome these important problems, padding is the solution. In padding, we layer of values like adding layer(s) of zeros around the feature matrix, as shown in the below image.

CNN Operations

❑Padding:

❑If we take an image of size 9x9 and filer of size 3x3, if we add 1 layer of padding, then the image after applying convolution operation if of size 9x9.

❑Hence the problem of reduced size of image after convolution is taken care of and because of padding, the pixel values on the edges are now somewhat shifted towards the middle.

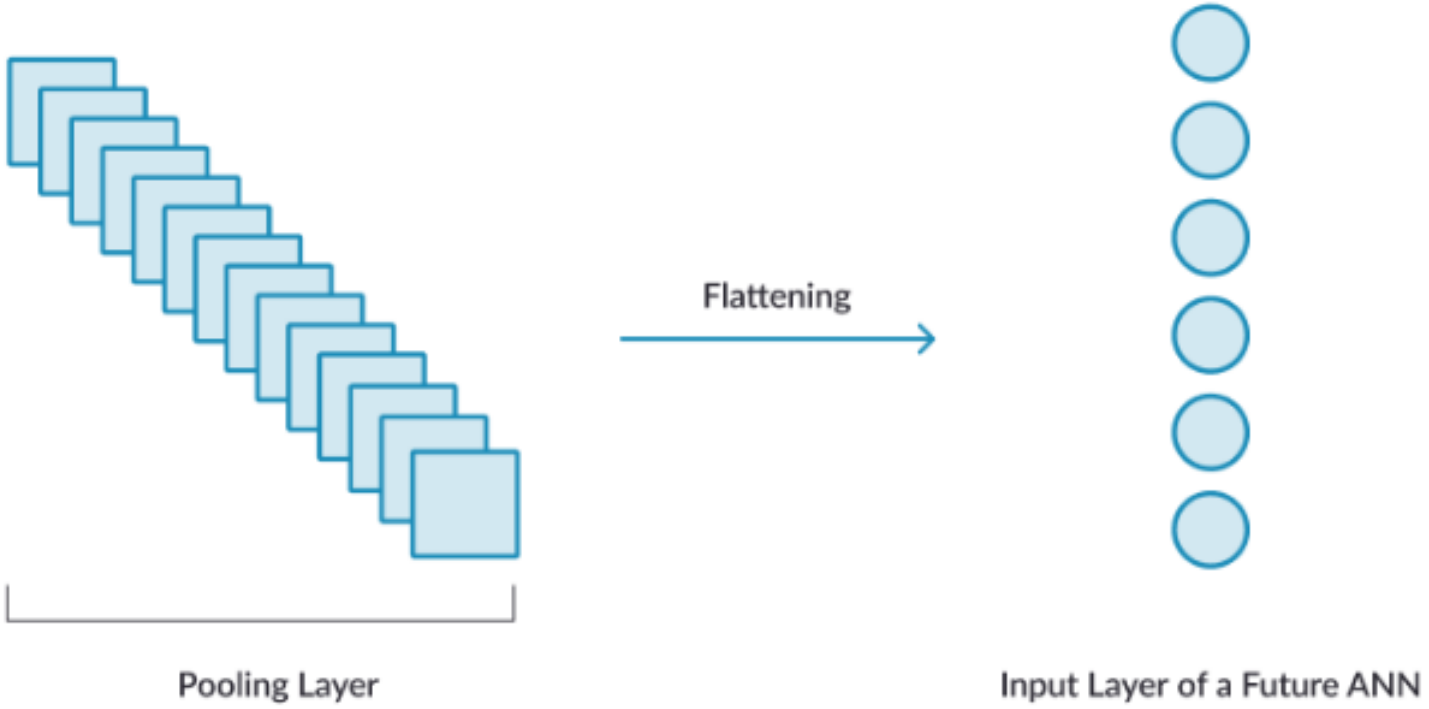


Zero-padding added to image

CNN Operations

❑Flattening

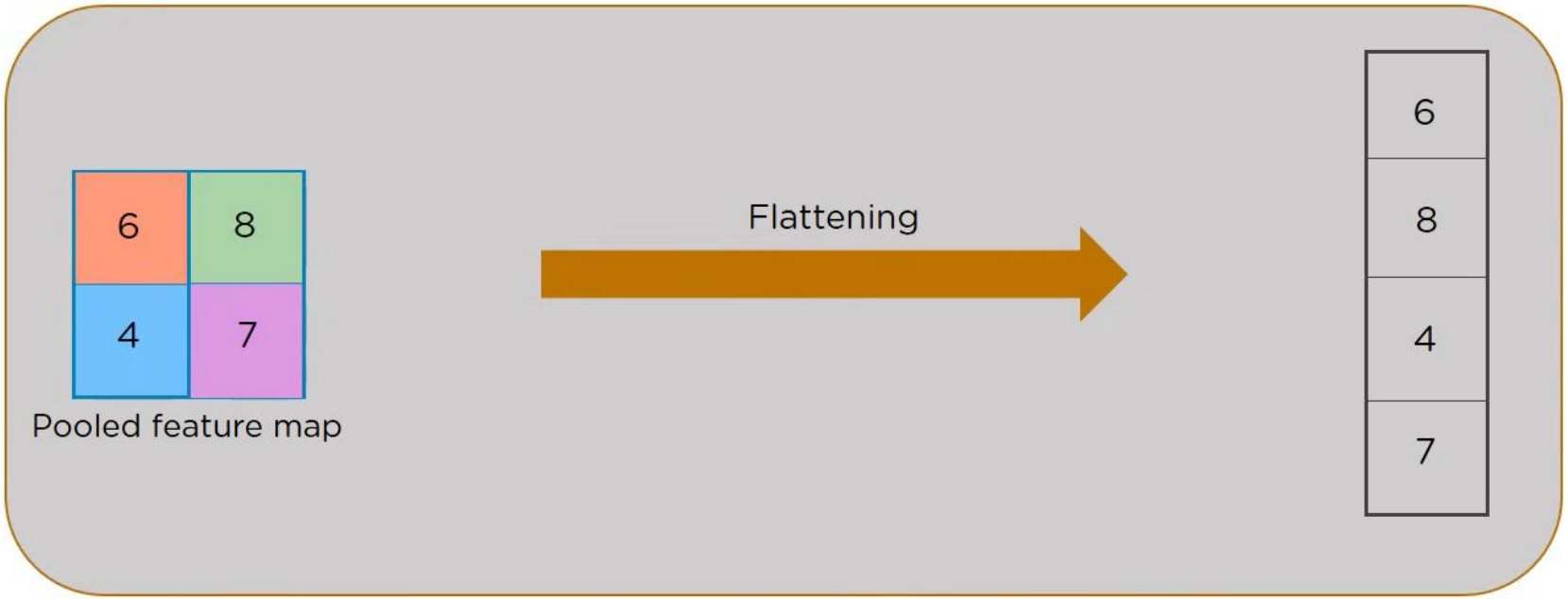
- ❑This is a step that is used in CNN but not always.
- ❑Based on the upcoming layers in the CNN, this step is involved.
- ❑What happens here is that the pooled feature map (i.e. the matrix) is converted into a vector.
- ❑And this vector plays the role of input layer in the upcoming neural networks.



CNN Operations

❑Flattening

Flattening is the process of converting all the resultant 2 dimensional arrays from pooled feature map into a single long continuous linear vector.



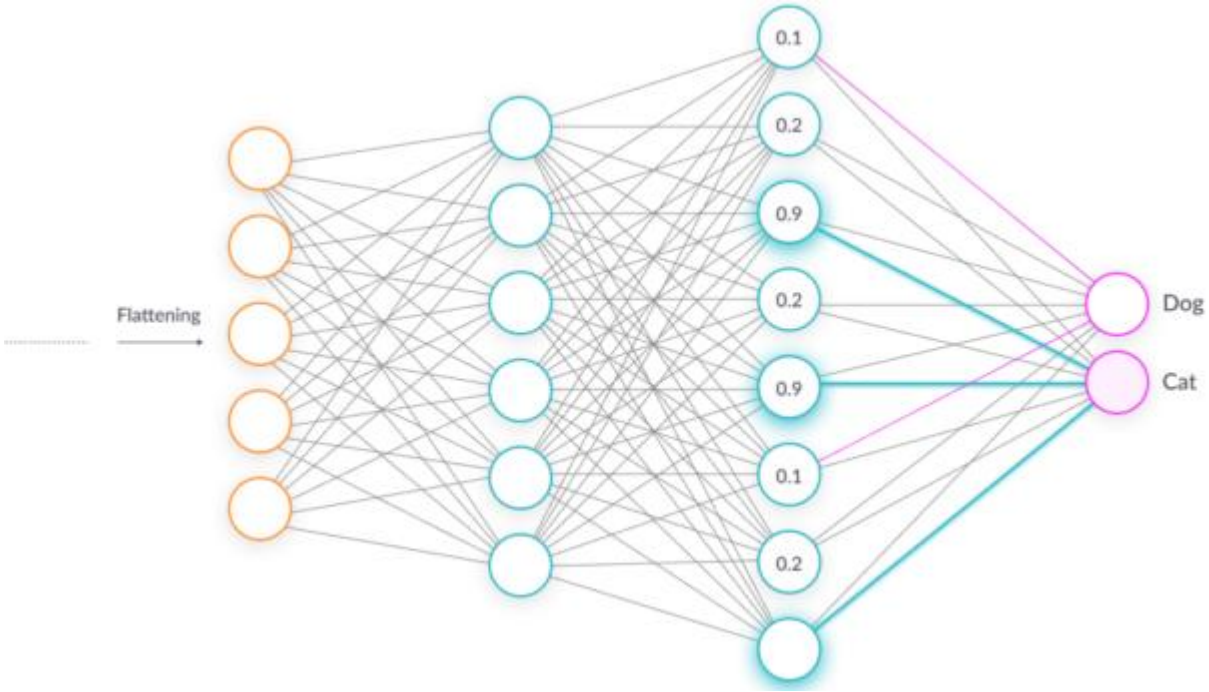
CNN Operations

❑Fully Connected Layers

❑After several layers of convolution and pooling operations are completed, now the final output is given to the fully connected layer.

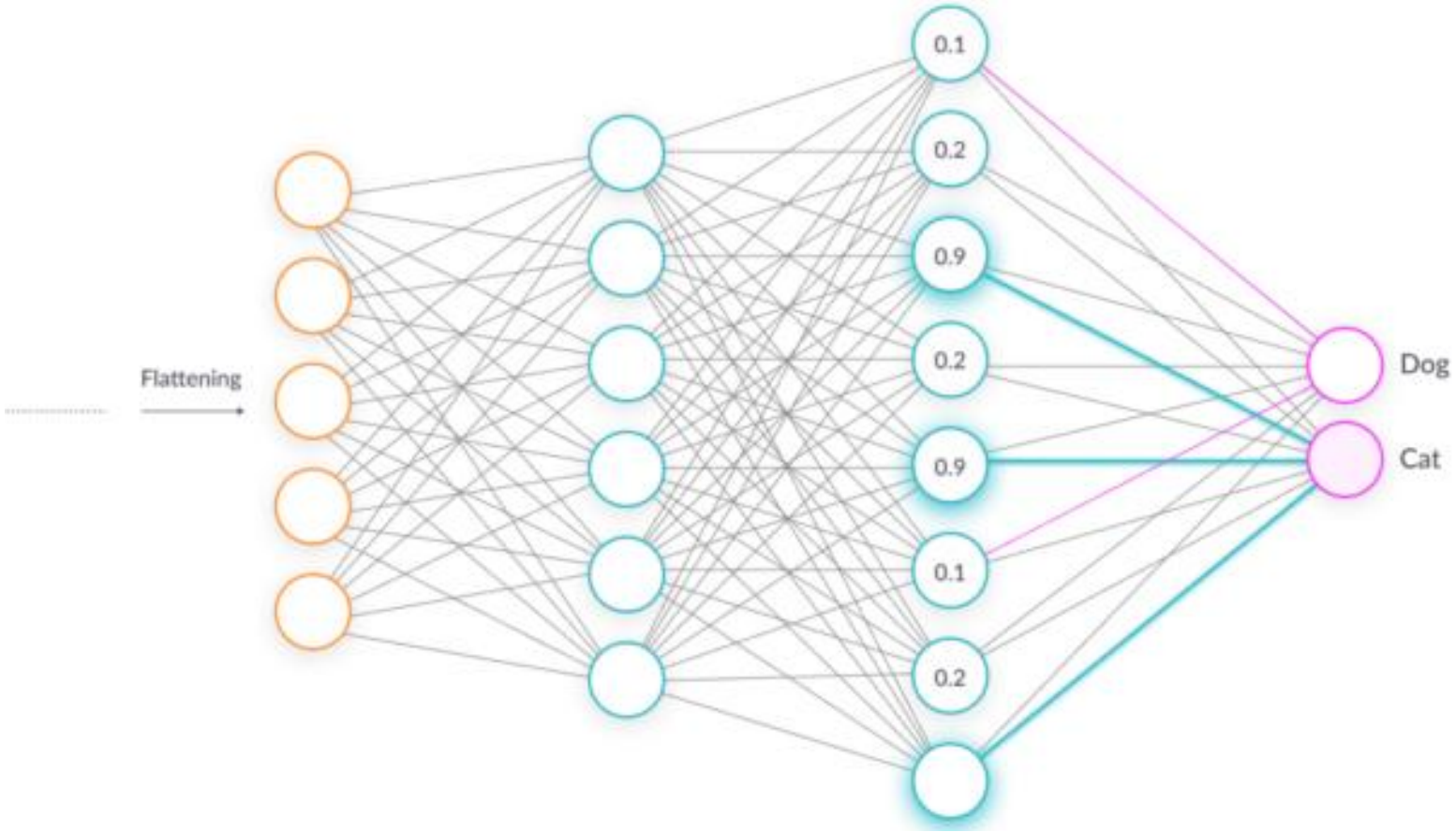
❑This is basically a neural network in which each neuron is connected to every other neuron in the previous layer.

❑All the recognition and classification parts are done by this neural network.



CNN Operations

❑Fully Connected Layers

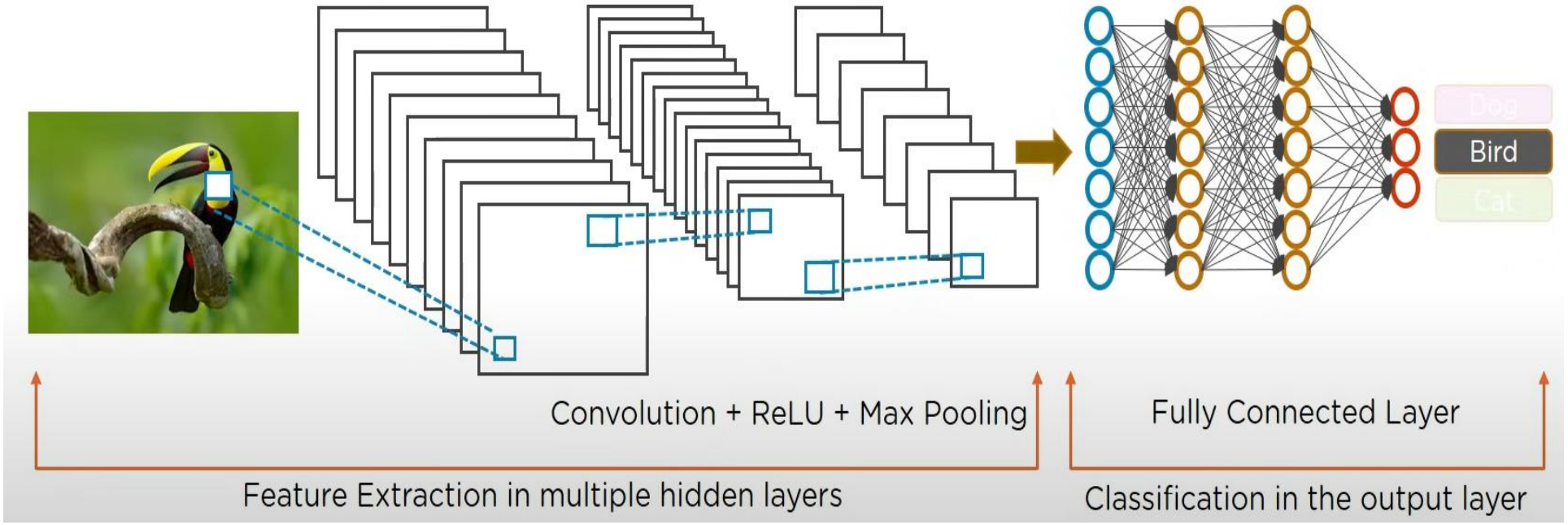


❑In the above image, the output after flattening is given to the fully connected layer and this network helps is in classifying the image as either cat or dog.

CNN Operations

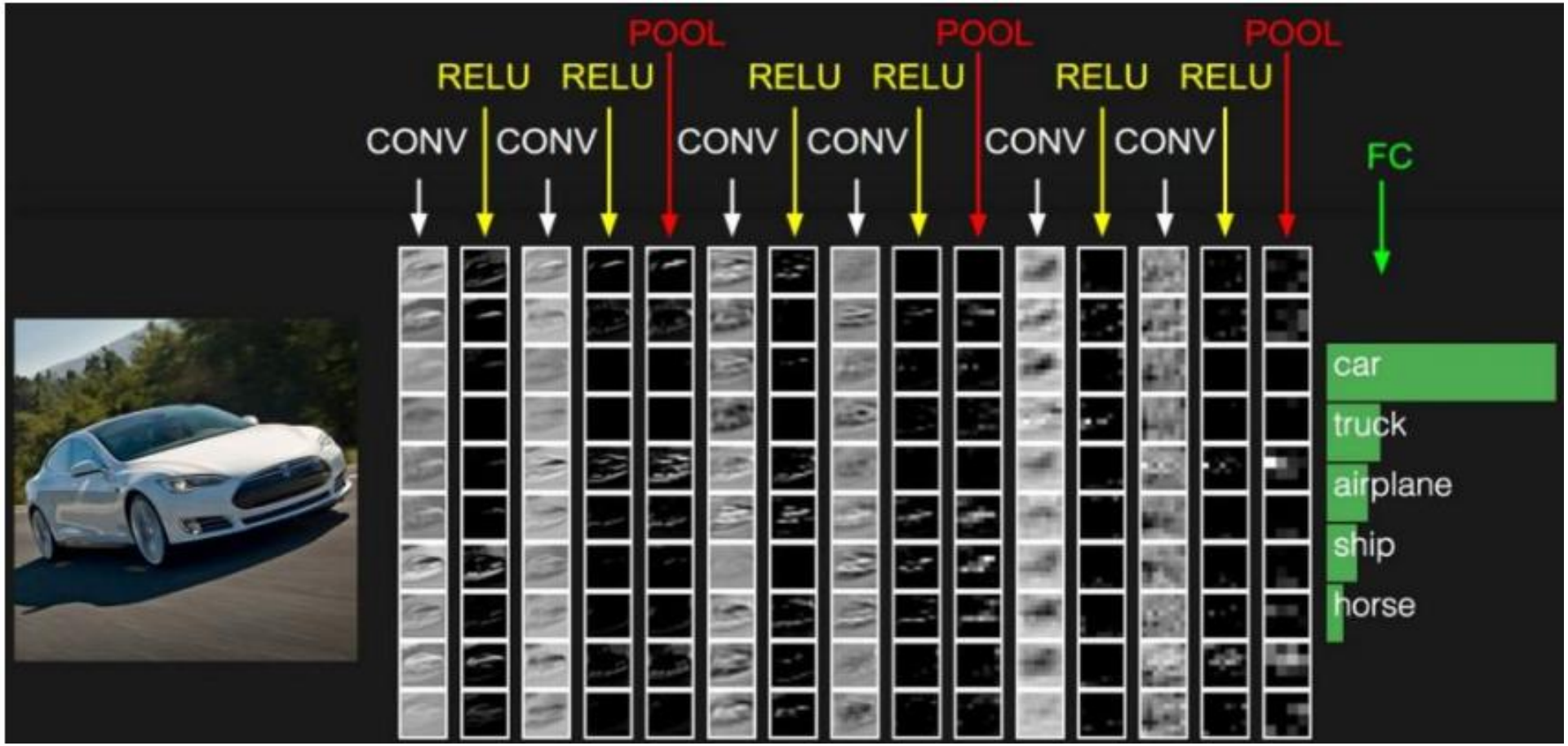
❑Fully Connected Layers

❑These layers are in the last layer of the convolutional neural network, and their inputs correspond to the flattened one-dimensional matrix generated by the last pooling layer.



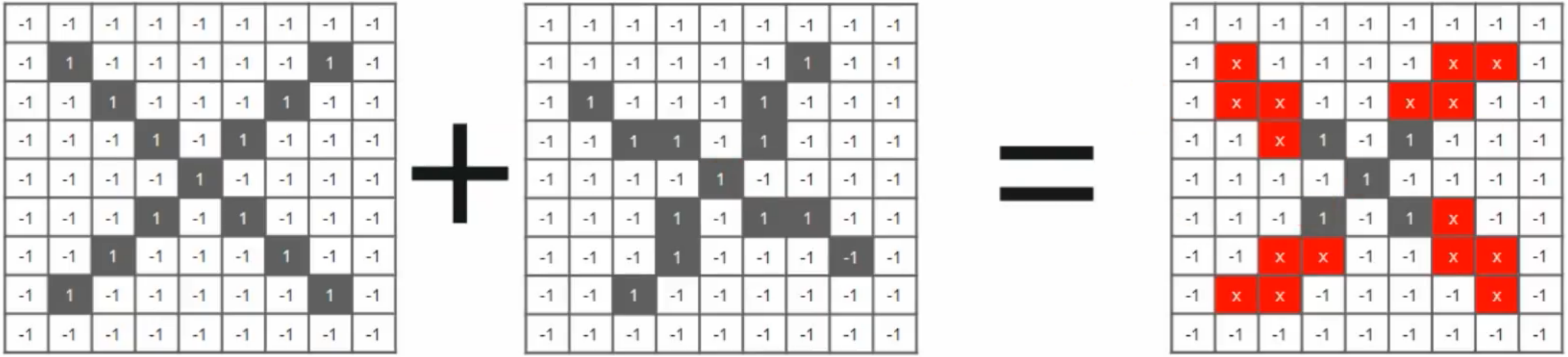
CNN Operations

❑Fully Connected Layers



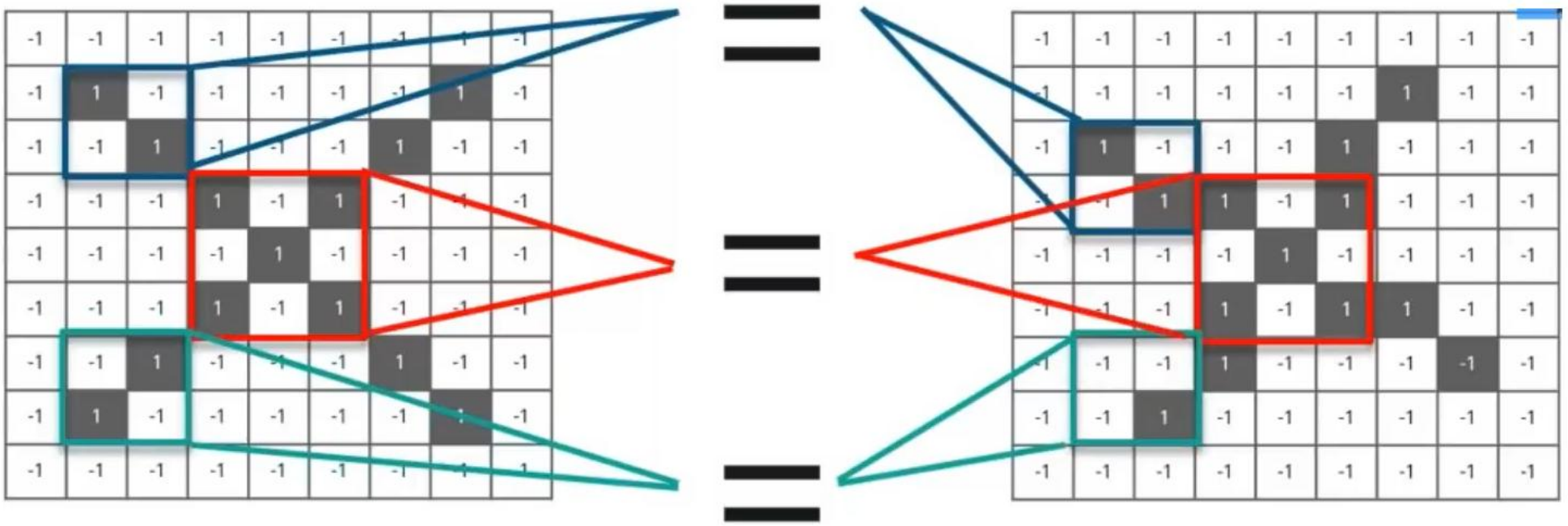
CNN Operations (Examples)

Using normal techniques, computers compare these images as:

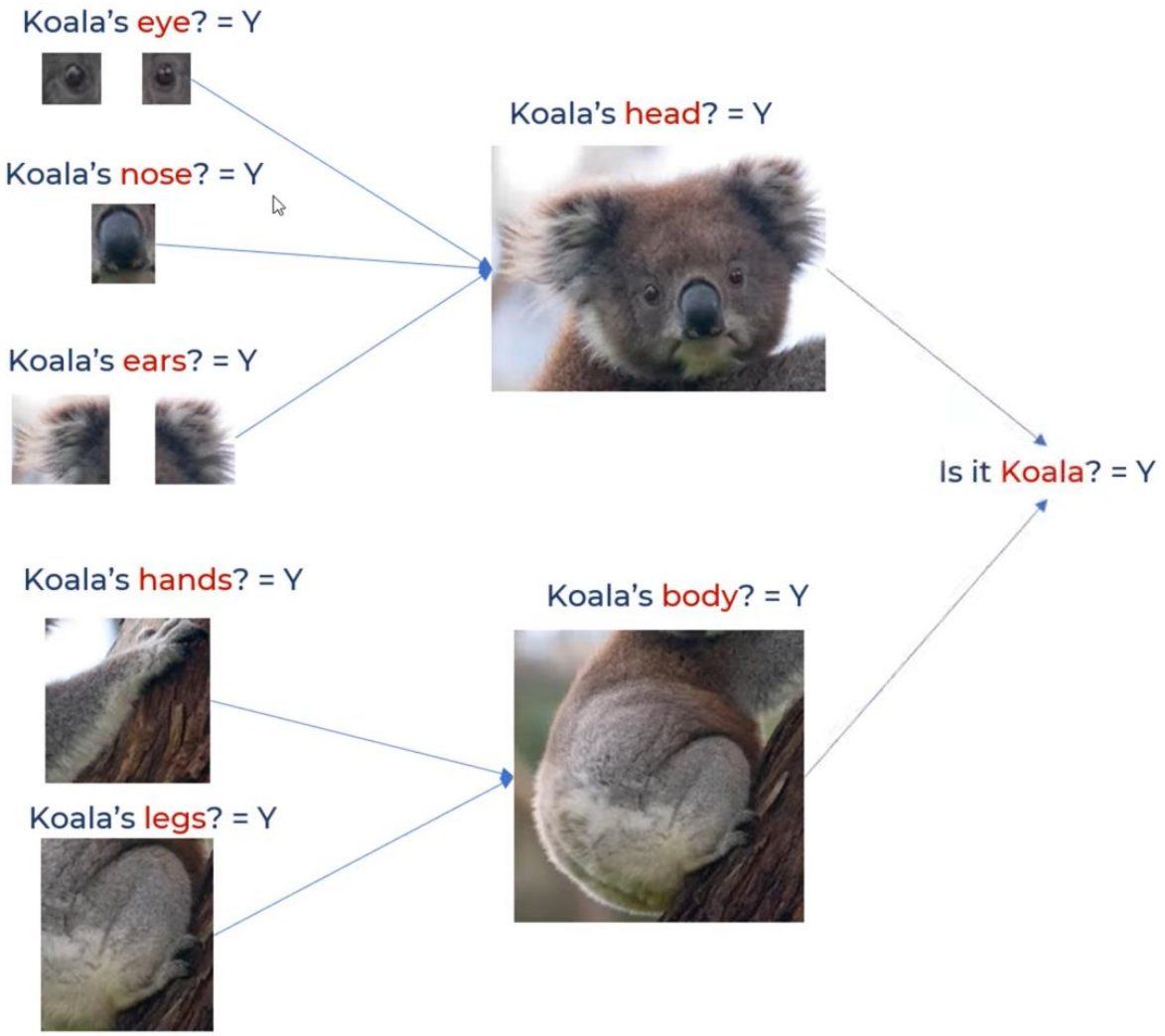


CNN Operations (Examples)

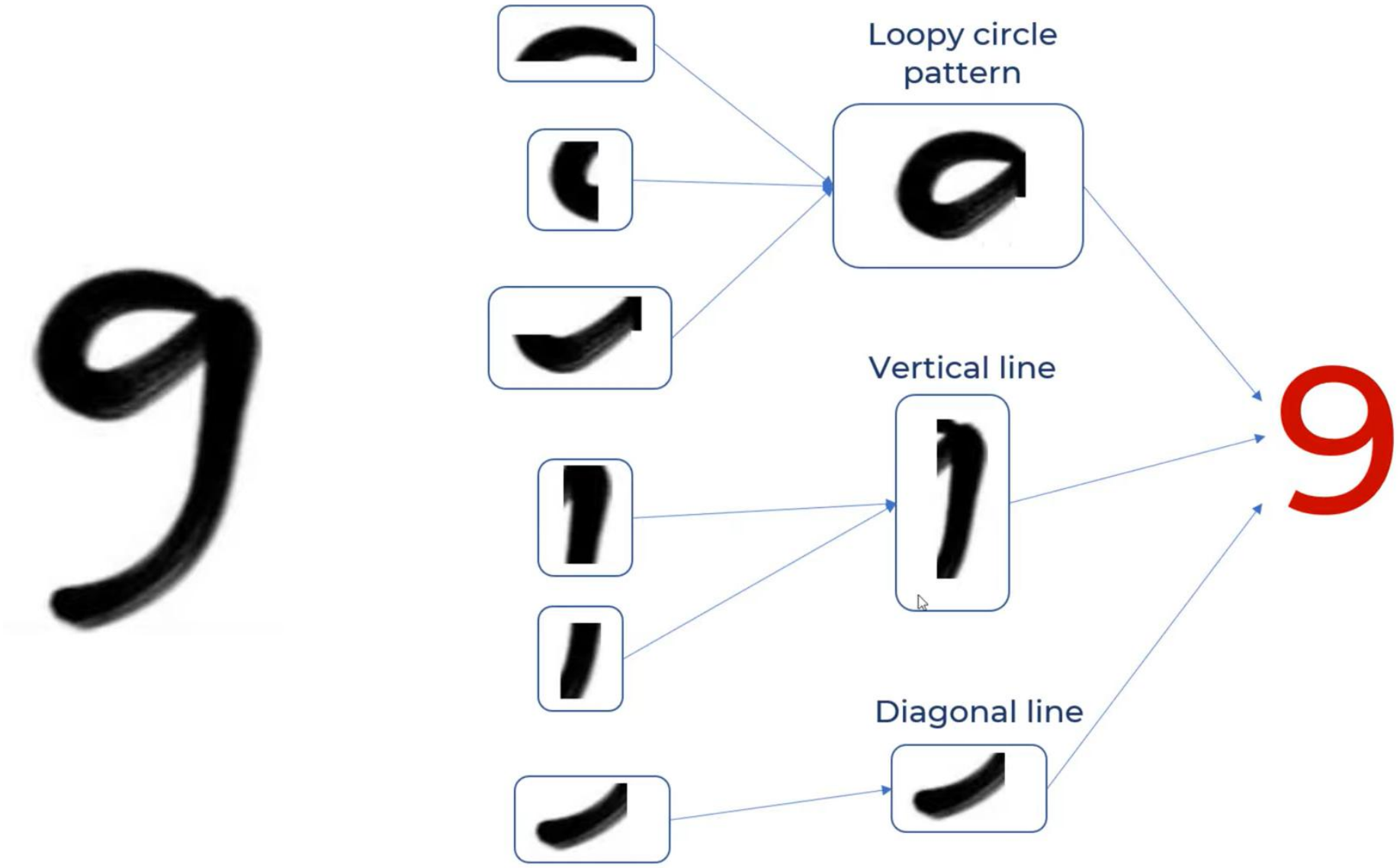
CNN compares the images piece by piece. The pieces that it looks for are called features.
By finding rough feature matches, in roughly the same position in two images, CNN gets a lot better at seeing similarity than whole-image matching schemes.



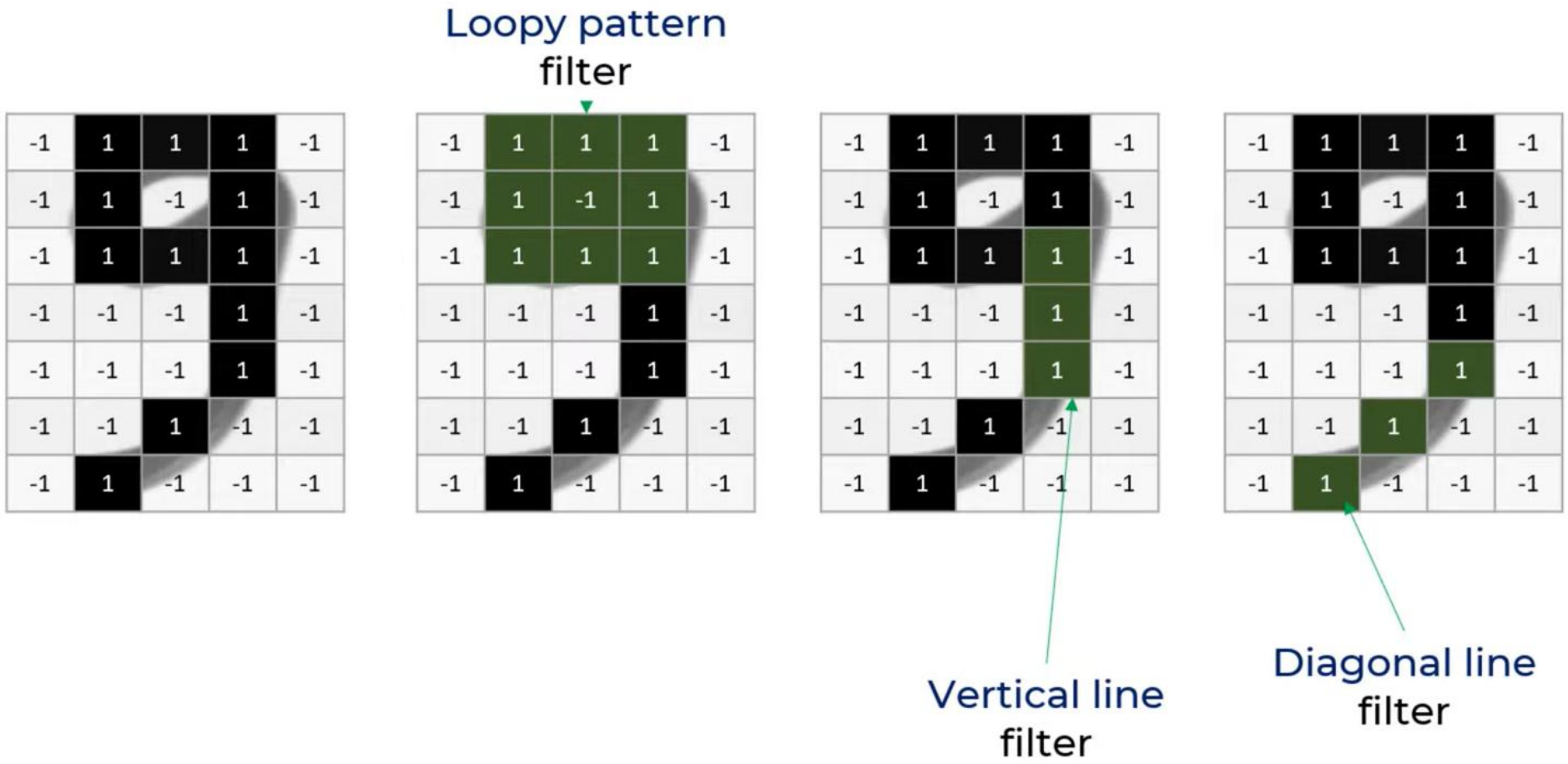
CNN Operations (Examples)



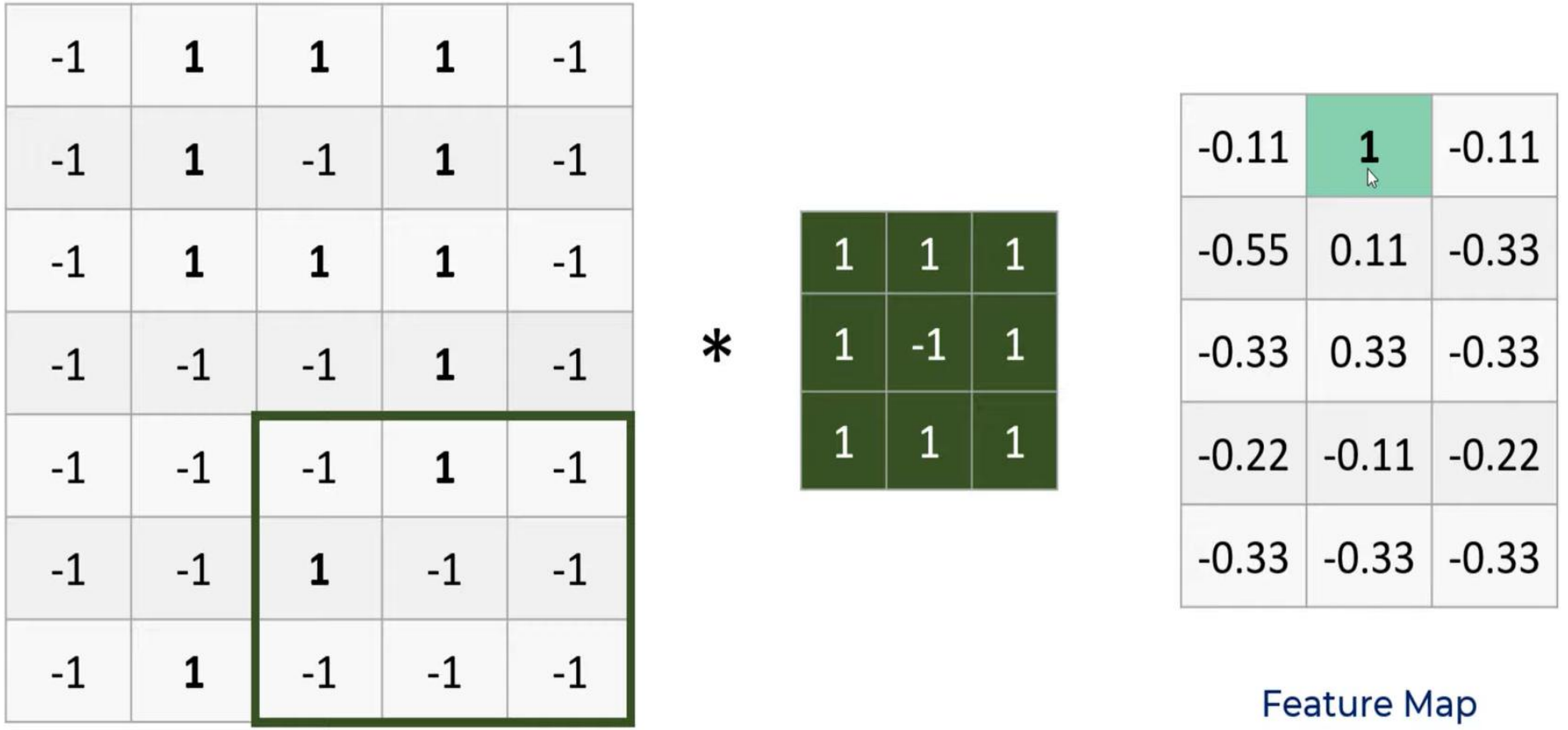
CNN Operations (Examples)



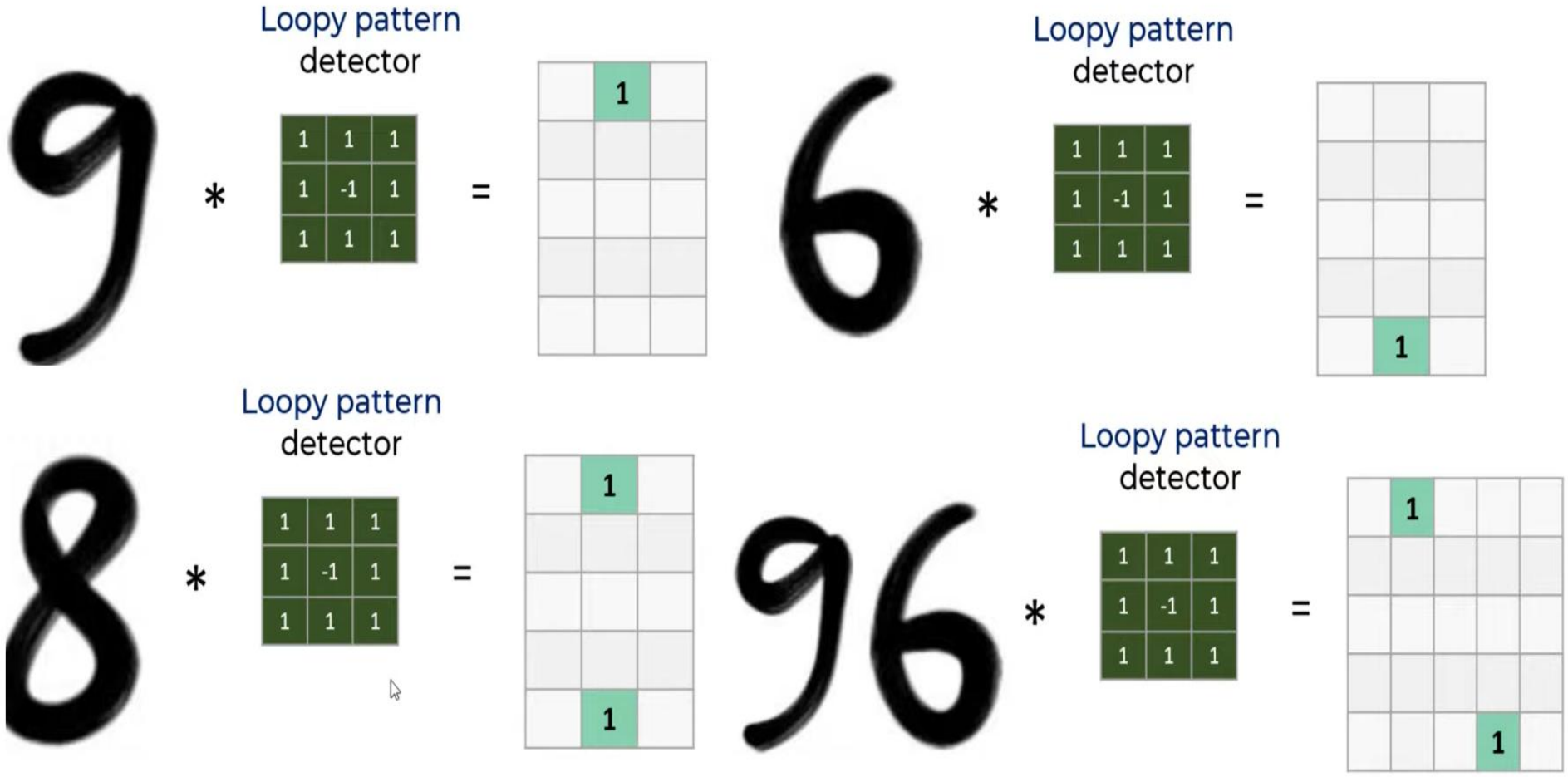
CNN Operations (Examples)



CNN Operations (Examples)

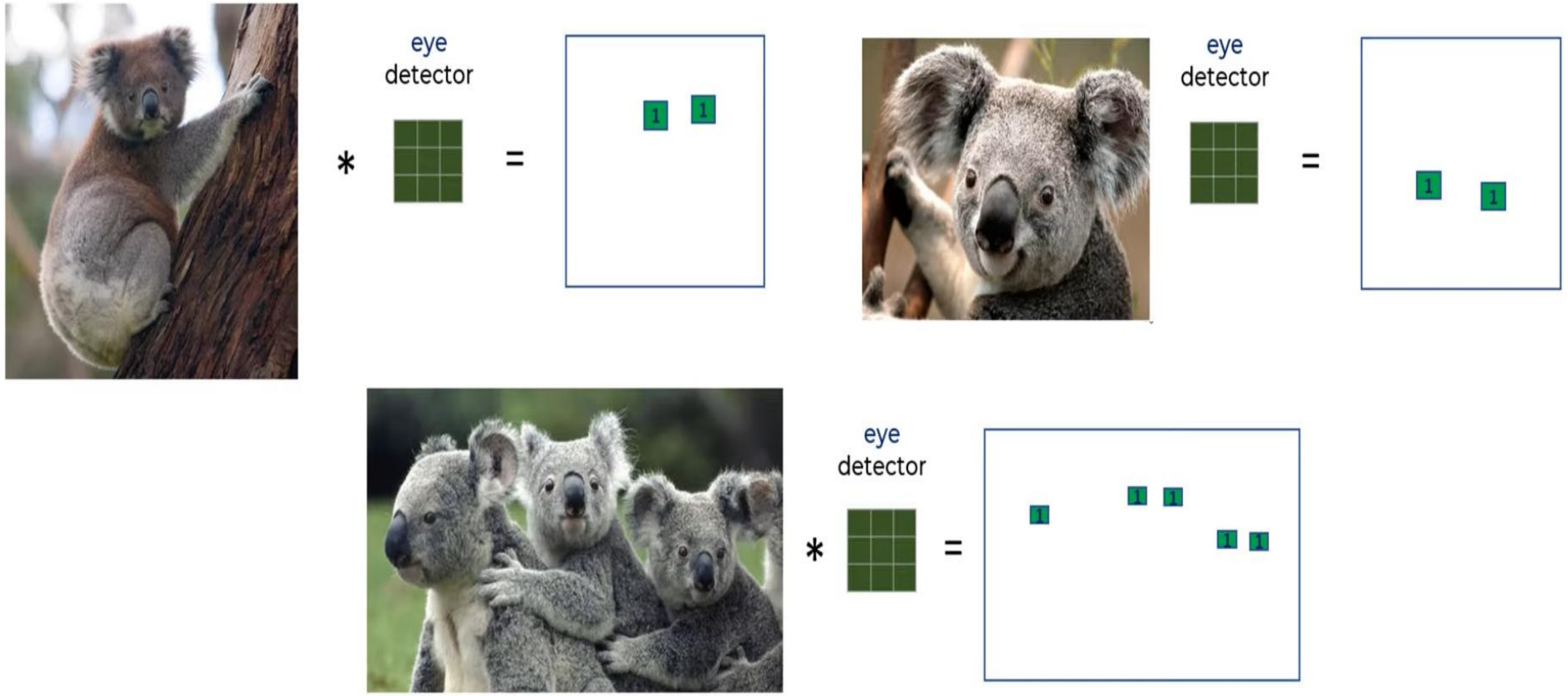


CNN Operations (Examples)

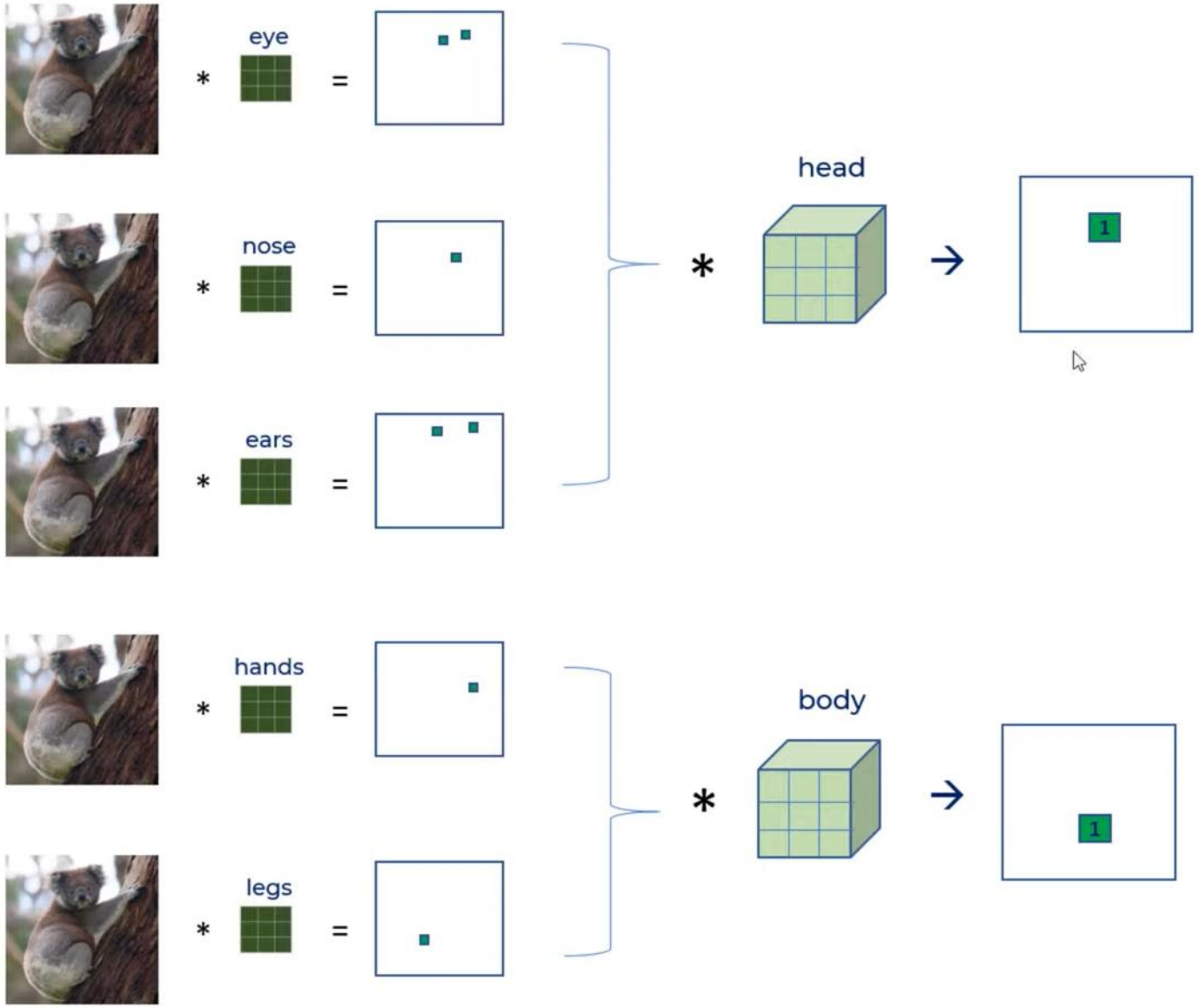


CNN Operations (Examples)

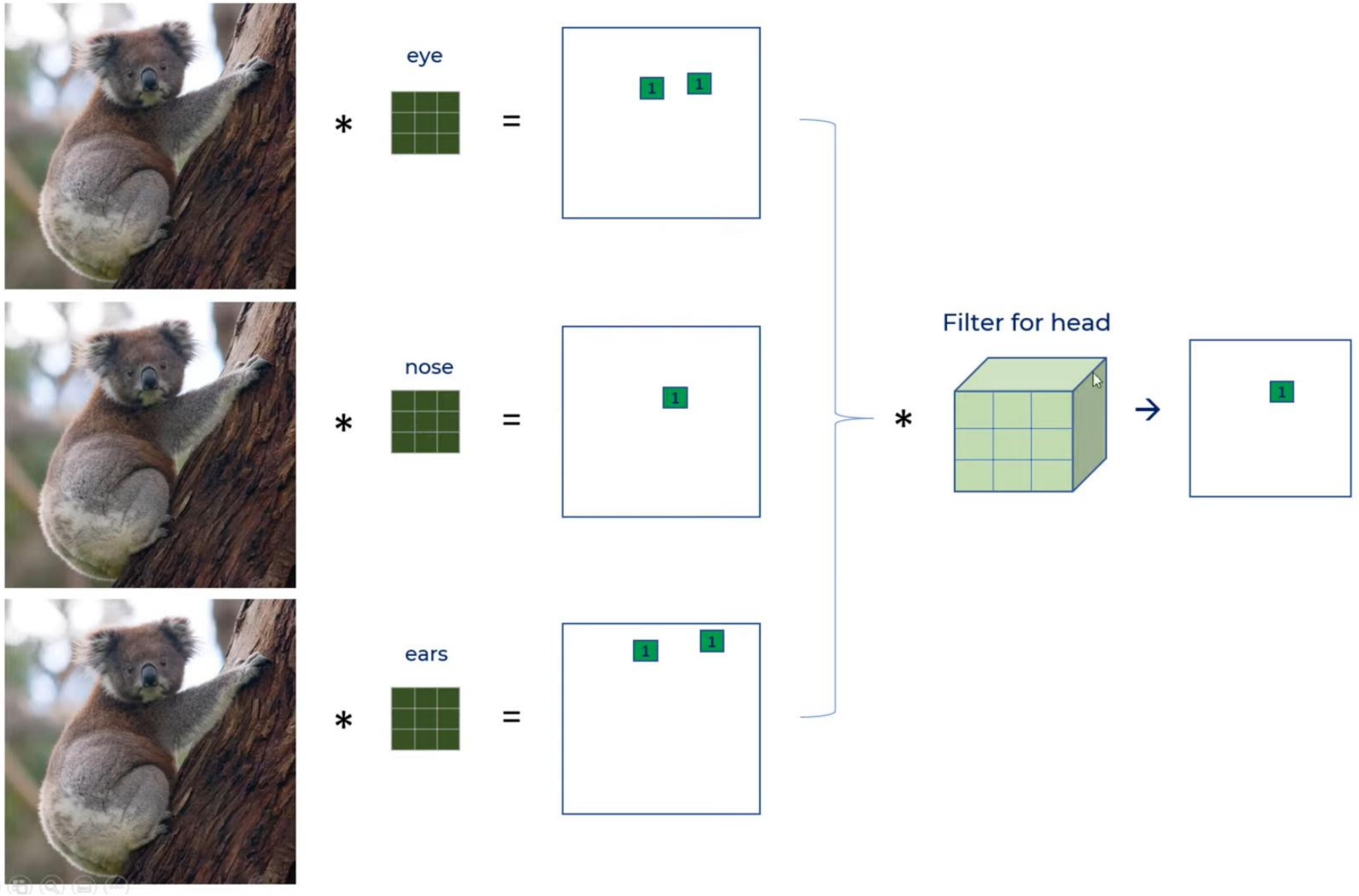
Location invariant: It can detect eyes in any location of the image



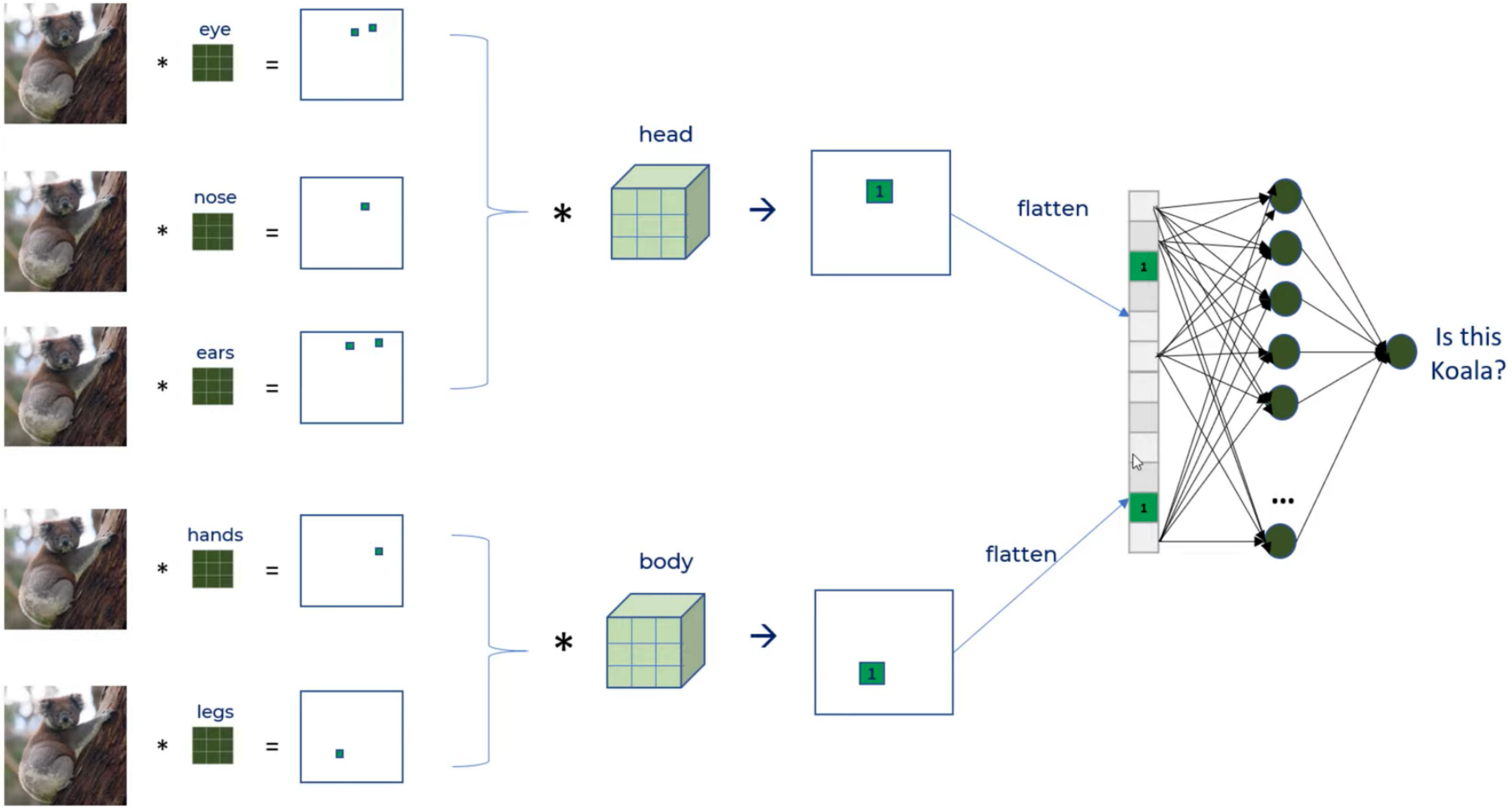
CNN Operations (Examples)



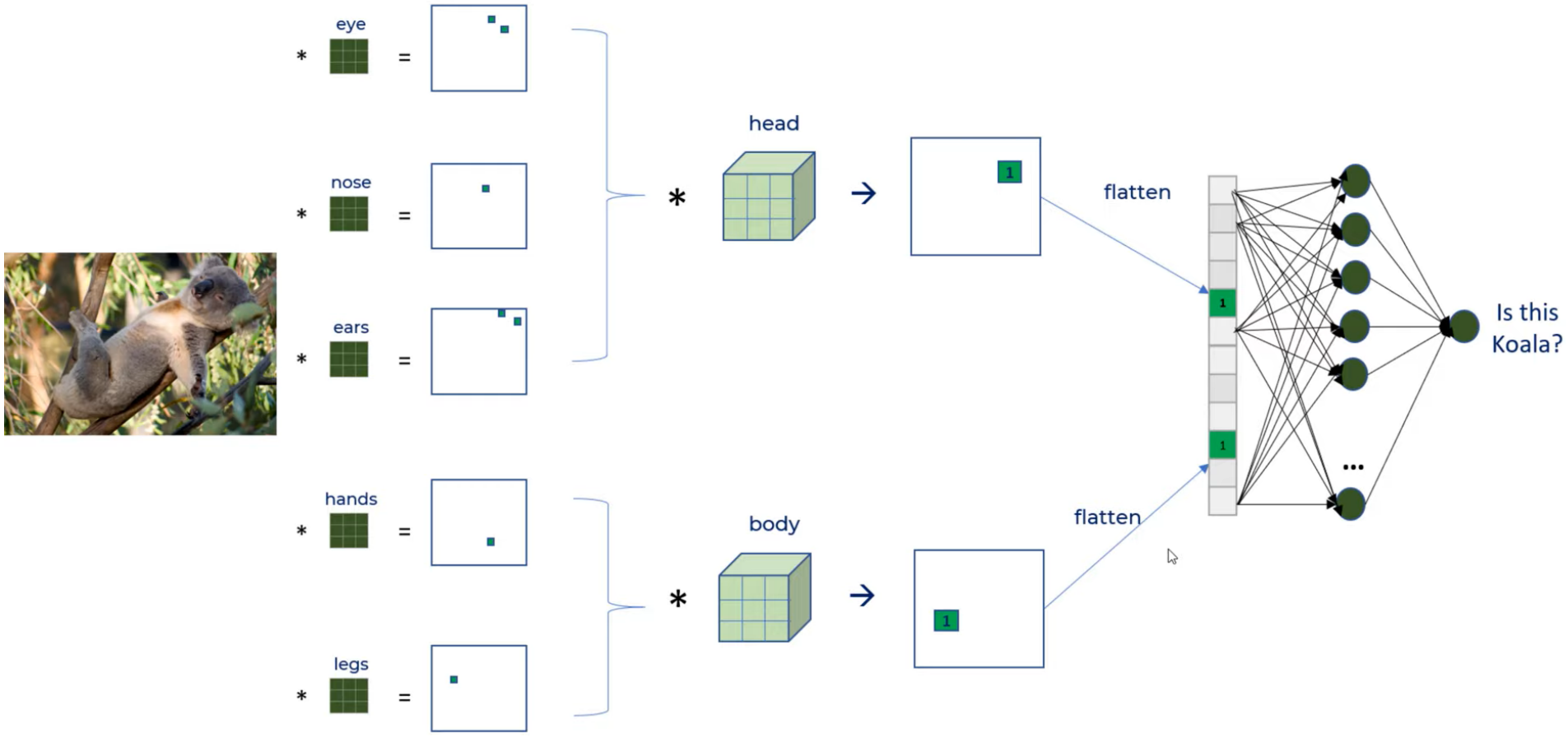
CNN Operations (Examples)



CNN Operations (Examples)



CNN Operations (Examples)



CNN Operations (Examples)

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

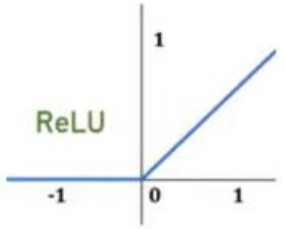
*

Loopy pattern filter

1	1	1
1	-1	1
1	1	1

→

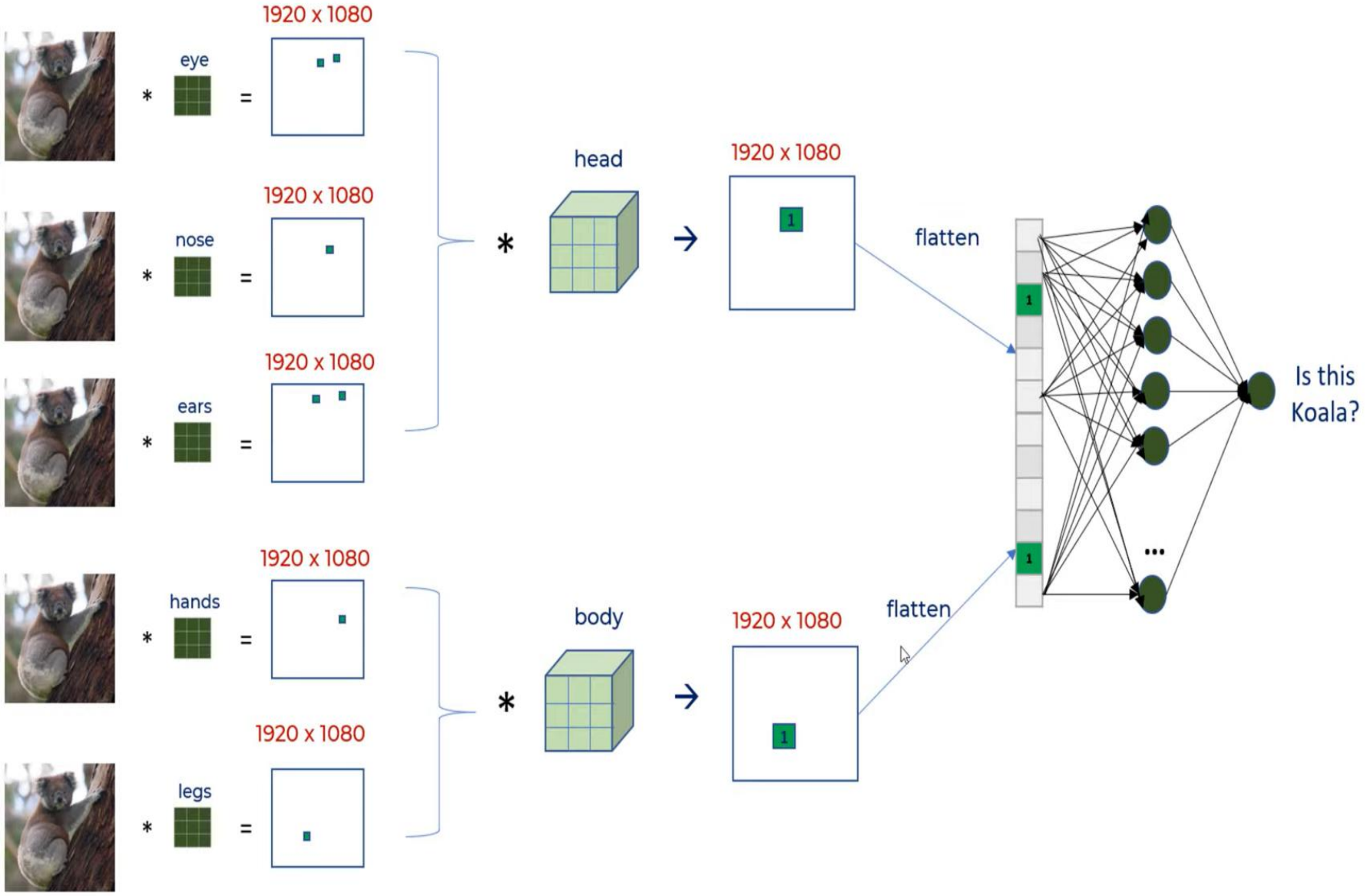
-0.11	**1**	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33



→

0	**1**	0
0	0.11	0
0	0.33	0
0	0	0
0	0	0

CNN Operations (Examples)



CNN Operations (Examples)

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

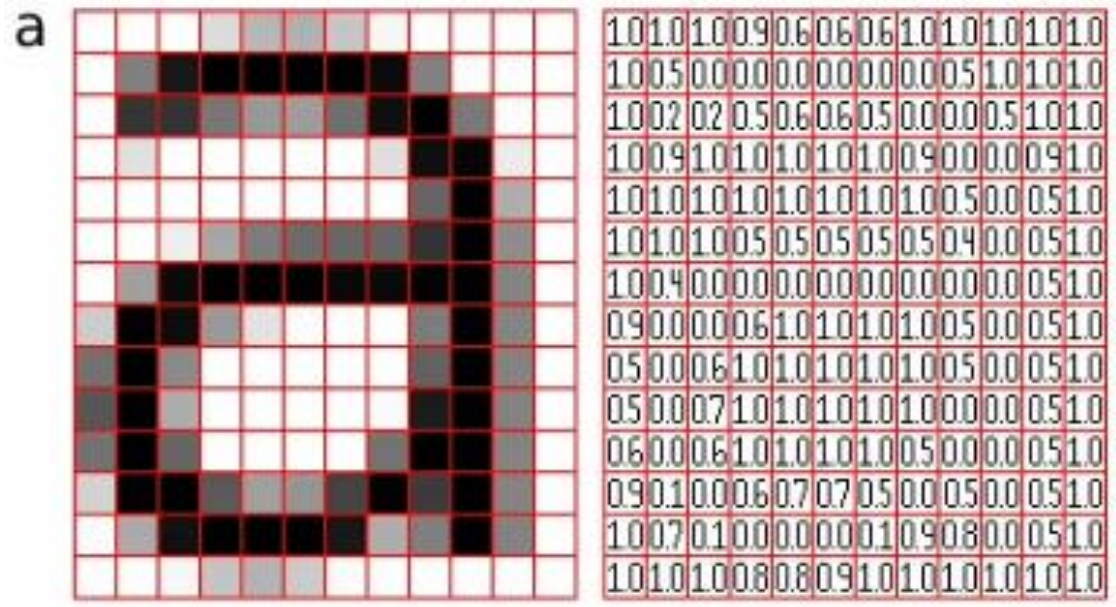
8	**9**
3	2

2 by 2 filter with stride = 2

CNN Architecture

❑A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image.

❑A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.

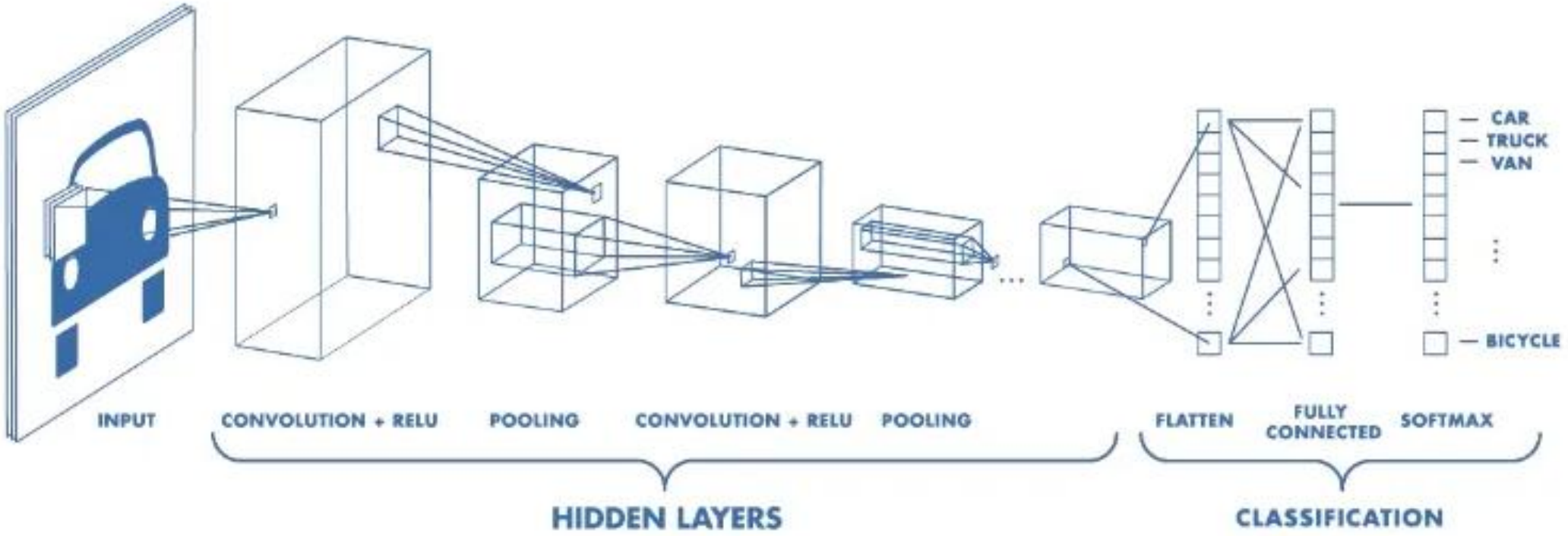


CNN Architecture

- The human brain processes a huge amount of information the second we see an image.
- Each neuron works in its own receptive field and is connected to other neurons in a way that they cover the entire visual field.
- Just as each neuron responds to stimuli only in the restricted region of the visual field called the receptive field in the biological vision system, each neuron in a CNN processes data only in its receptive field as well.
- The layers are arranged in such a way so that they detect simpler patterns first (lines, curves, etc.) and more complex patterns (faces, objects, etc.) further along.
- By using a CNN, one can enable sight to computers.

CNN Architecture

❑**Convolutional Neural Network Architecture:** A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.



CNN Architecture

❑Convolutional Neural Network Architecture: Convolution Layer

❑The convolution layer is the core building block of the CNN. It carries the main portion of the network's computational load.

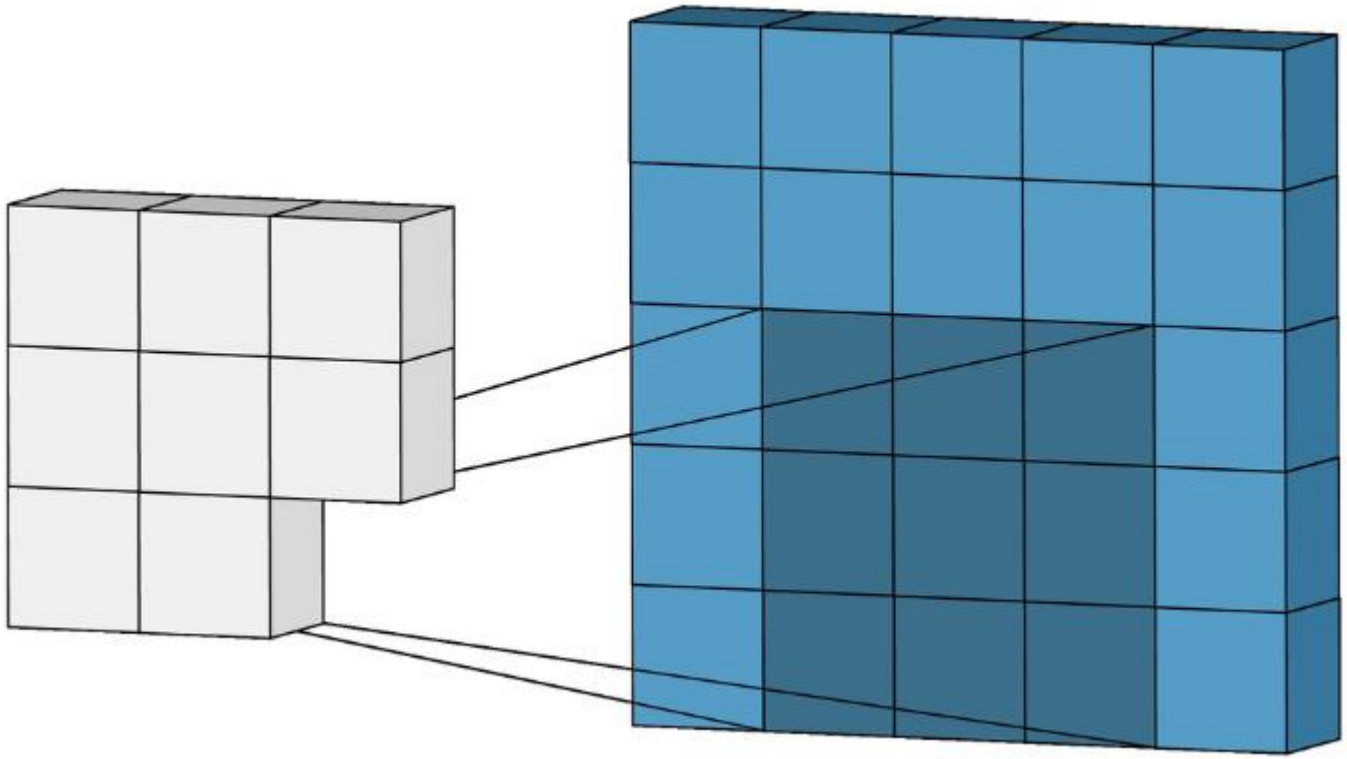
❑This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field.

❑The kernel is spatially smaller than an image but is more in-depth.

❑This means that, if the image is composed of three (RGB) channels, the kernel height and width will be spatially small, but the depth extends up to all three channels.

CNN Architecture

❑Convolutional Neural Network Architecture: Convolution Layer



❑Convolutional Neural Network Architecture: Convolution Layer

❑During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region.

❑This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image.

❑The sliding size of the kernel is called a stride.

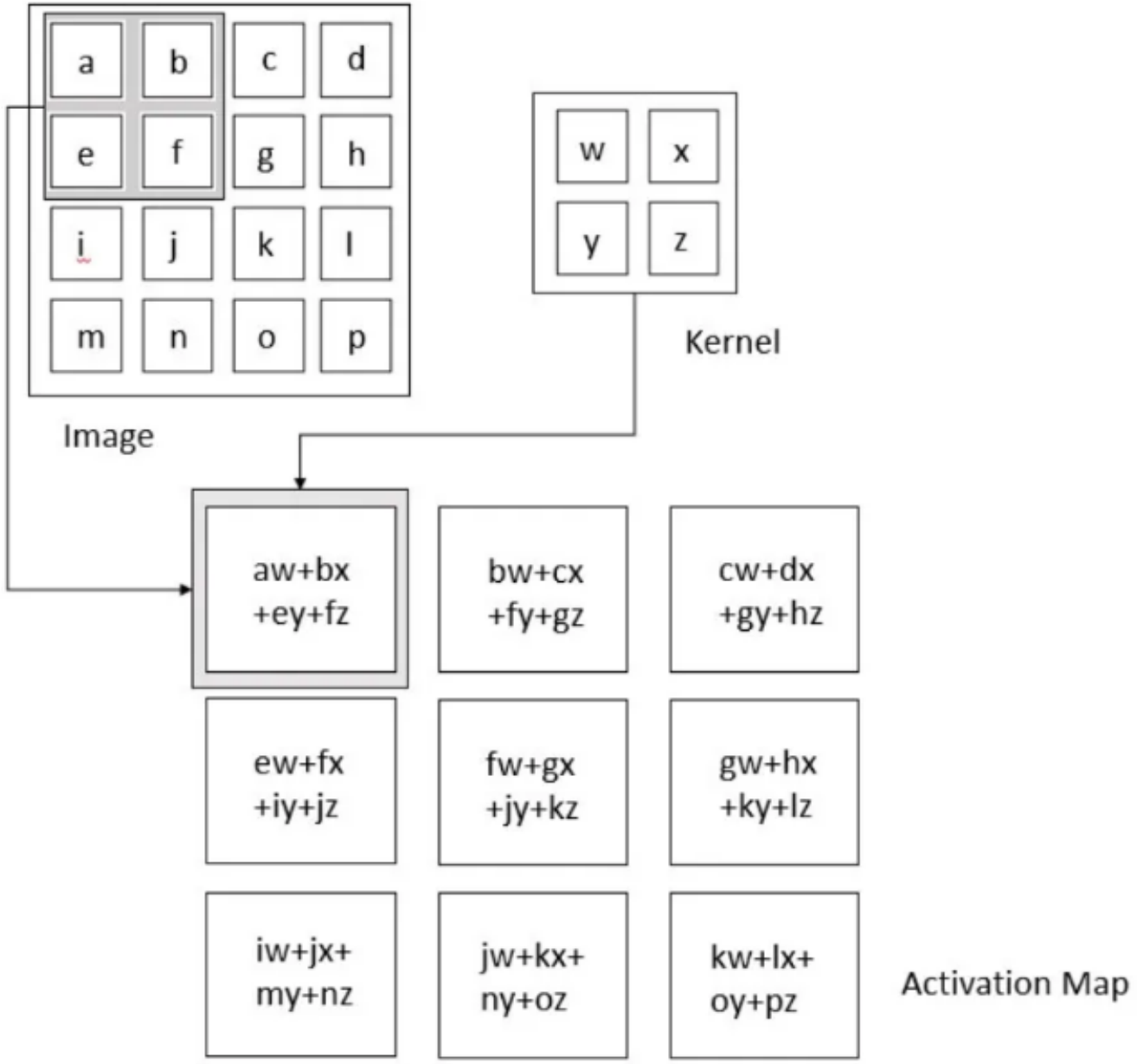
❑If we have an input of size W x W x D and Dout number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

❑This will yield an output volume of size ***Wout* x *Wout* x *Dout***.

CNN Architecture

❑Convolutional Neural Network Architecture: Convolution Layer



CNN Architecture

❑Convolutional Neural Network Architecture: Pooling Layer

❑The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs.

❑This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights.

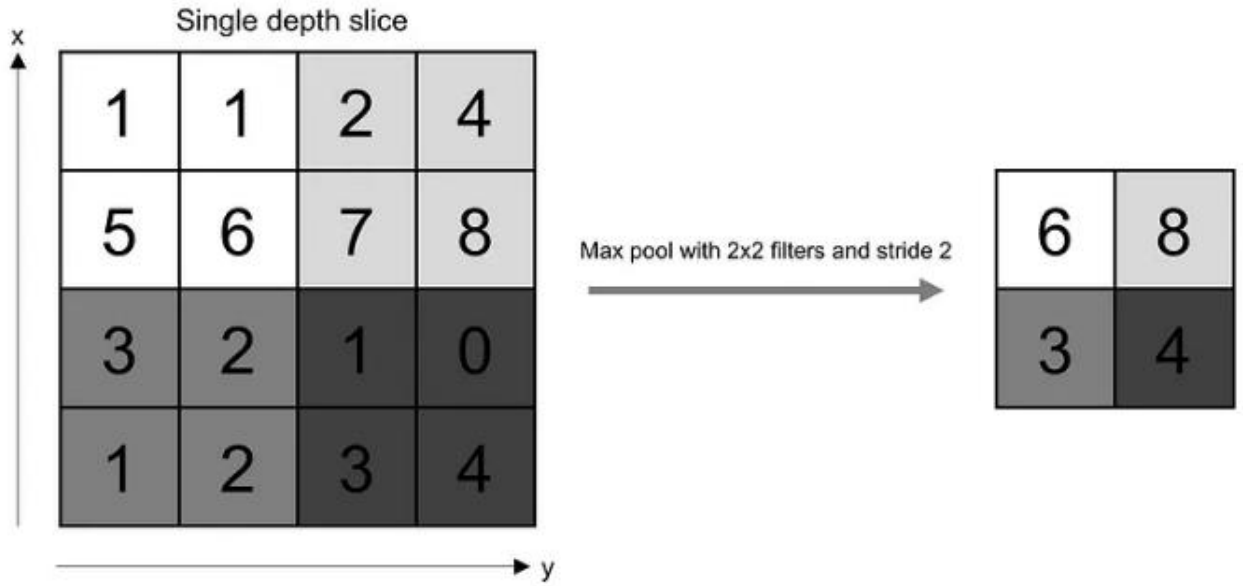
❑The pooling operation is processed on every slice of the representation individually.

❑There are several pooling functions such as the average of the rectangular neighborhood, L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel.

❑However, the most popular process is max pooling, which reports the maximum output from the neighborhood.

❑Convolutional Neural Network Architecture: Pooling Layer

❑If we have an activation map of size *W* x *W* x *D*, a pooling kernel of spatial size *F*, and stride *S*, then the size of output volume can be determined by the following formula:



$$W_{out} = \frac{W - F}{S} + 1$$

❑This will yield an output volume of size *Wout* x *Wout* x *D*. In all cases, pooling provides some translation invariance which means that an object would be recognizable regardless of where it appears on the frame.

CNN Architecture

❑Convolutional Neural Network Architecture: Fully Connected Layer

❑Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN.

❑This is why it can be computed as usual by a matrix multiplication followed by a bias effect.

❑The FC layer helps to map the representation between the input and the output.

CNN Architecture

❑**Convolutional Neural Network Architecture: Non-Linearity Layers (Activation Function)**

❑Since convolution is a linear operation and images are far from linear, non-linearity layers are often placed directly after the convolutional layer to introduce non-linearity to the activation map.

❑An Activation Function decides whether a neuron should be activated or not.

❑This means that it will decide whether the neuron's input to the network is important or not in the process of prediction.

❑There are several commonly used activation functions such as the ReLU, Softmax, tanH, and the Sigmoid functions. Each of these functions has a specific usage.

❑**Sigmoid:** For a binary classification in the CNN model

❑**tanH:** The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values, in this case, is from -1 to 1.

❑**Softmax:** It is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes.

❑**RelU:** the main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.

CNN Architecture

❑Designing a Convolutional Neural Network:

❑Now that we understand the various components, we can build a convolutional neural network.

❑We will be using Fashion-MNIST, which is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples.

❑Each example is a 28x28 grayscale image, associated with a label from 10 classes.

❑Our convolutional neural network has architecture as follows:

[INPUT]

→[CONV 1] → [BATCH NORM] → [ReLU] → [POOL 1]

→[CONV 2] → [BATCH NORM] → [ReLU] → [POOL 2]

→[FC LAYER] → [RESULT]

❑For both conv layers, we will use kernel of spatial size 5 x 5 with stride size 1 and padding of 2.

❑For both pooling layers, we will use max pool operation with kernel size 2, stride 2, and zero padding.

CNN Architecture

❑Designing a Convolutional Neural Network:

CONV 1
Input Size (W_1 x H_1 x D_1) = 28 x 28 x 1
• Requires four hyperparameter: ○ Number of kernels, k = 16 ○ Spatial extend of each one, F = 5 ○ Stride Size, S = 1 ○ Amount of zero padding, P = 2
• Outputting volume of W_2 x H_2 x D_2 ○ W_2 = (28 – 5 + 2 (2)) / 1 + 1 = 28 ○ H_2 = (28 – 5 + 2 (2)) / 1 + 1 = 28 ○ D_2 = k
Output of Conv 1 (W_2 x H_2 x D_2) = 28 x 28 x 16

CNN Architecture

❑Designing a Convolutional Neural Network:

POOL 1
Input Size (W_2 x H_2 x D_2) = 28 x 28 x 16
• Requires two hyperparameter: ○ Spatial extend of each one, F = 2 ○ Stride Size, S = 2
• Outputting volume of W_3 x H_3 x D_2 ○ W_3 = (28 – 2) / 2 + 1 = 14 ○ H_3 = (28 – 2) / 2 + 1 = 14
Output of Pool 1 (W_3 x H_3 x D_2) = 14 x 14 x 16

CNN Architecture

❑Designing a Convolutional Neural Network:

CONV 2
Input Size (W_3 x H_3 x D_2) = 14 x 14 x 16
• Requires four hyperparameter: o Number of kernels, k = 32 o Spatial extend of each one, F = 5 o Stride Size, S = 1 o Amount of zero padding, P = 2
• Outputting volume of W_4 x H_4 x D_3 o W_4 = (14 – 5 + 2 (2)) / 1 + 1 = 14 o H_4 = (14 – 5 + 2 (2)) / 1 + 1 = 14 o D_3 = k
Output of Conv 2 (W_4 x H_4 x D_3) = 14 x 14 x 32

CNN Architecture

❑Designing a Convolutional Neural Network:

POOL 2
Input Size (W_4 x H_4 x D_3) = 14 x 14 x 32
• Requires two hyperparameter: ○ Spatial extend of each one, F = 2 ○ Stride Size, S = 2
• Outputting volume of W_5 x H_5 x D_3 ○ W_5 = (14 – 2) / 2 + 1 = 7 ○ H_5 = (14 – 2) / 2 + 1 = 7
Output of Pool 2 (W_5 x H_5 x D_3) = 7 x 7 x 32

CNN Architecture

❑Designing a Convolutional Neural Network:

FC Layer
Input Size (W_5 x H_5 x D_3) = 7 x 7 x 32
Output Size (Number of Classes) = 10

Simple Convolutional Network

- **Experiential Learning:** Students need to complete the topic with reference to previous slides and other available material.

ResNet

- Deep residual networks were a breakthrough idea which enabled the development of much deeper networks (hundreds of layers as opposed to tens of layers).
- Its a generally accepted principle that deeper networks are capable of learning more complex functions and representations of the input which should lead to better performance.
- However, many researchers observed that adding more layers eventually had a negative effect on the final performance.
- This behavior was not intuitively expected, as explained further.

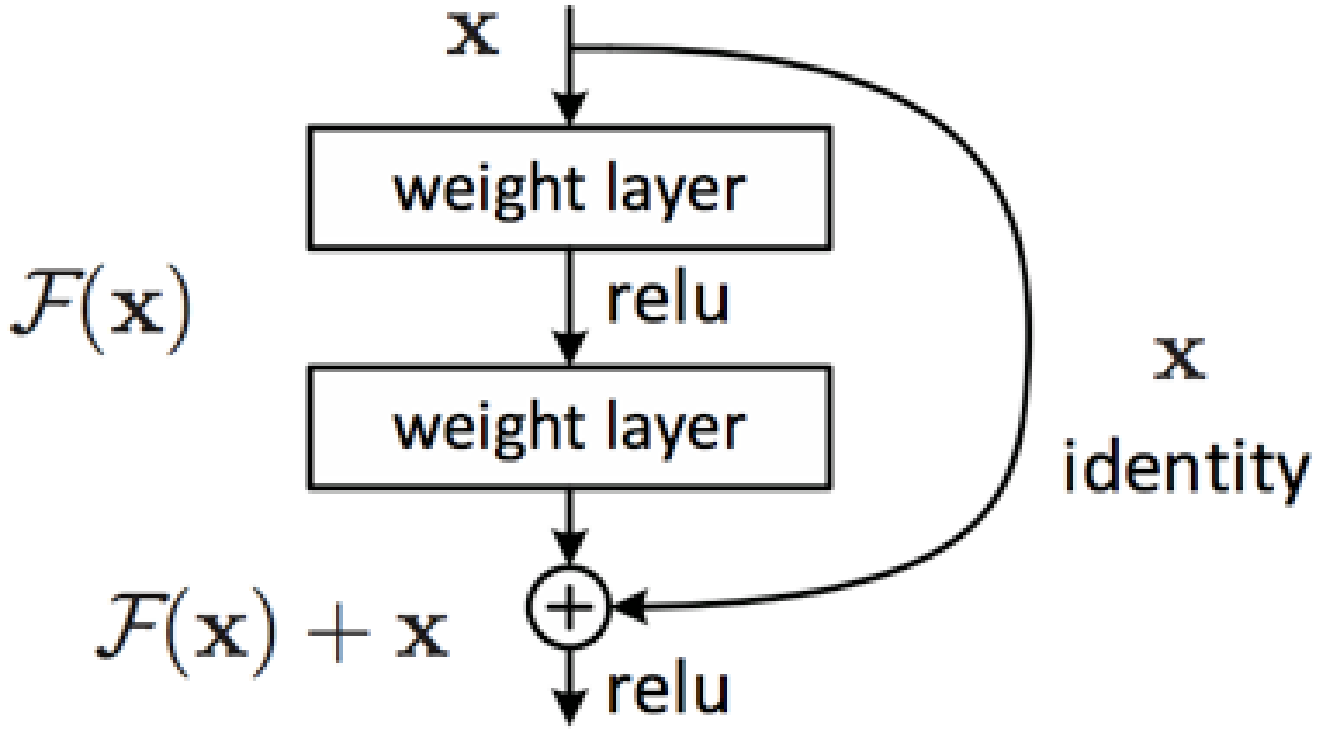
ResNet

- Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it.
- *There exists a solution by construction to the deeper model: the added layers are identity mapping, and the other layers are copied from the learned shallower model.*
- *The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart.*
- *But experiments show that our current solvers on hand are unable to find solutions that are comparably good or better than the constructed solution.*

ResNet

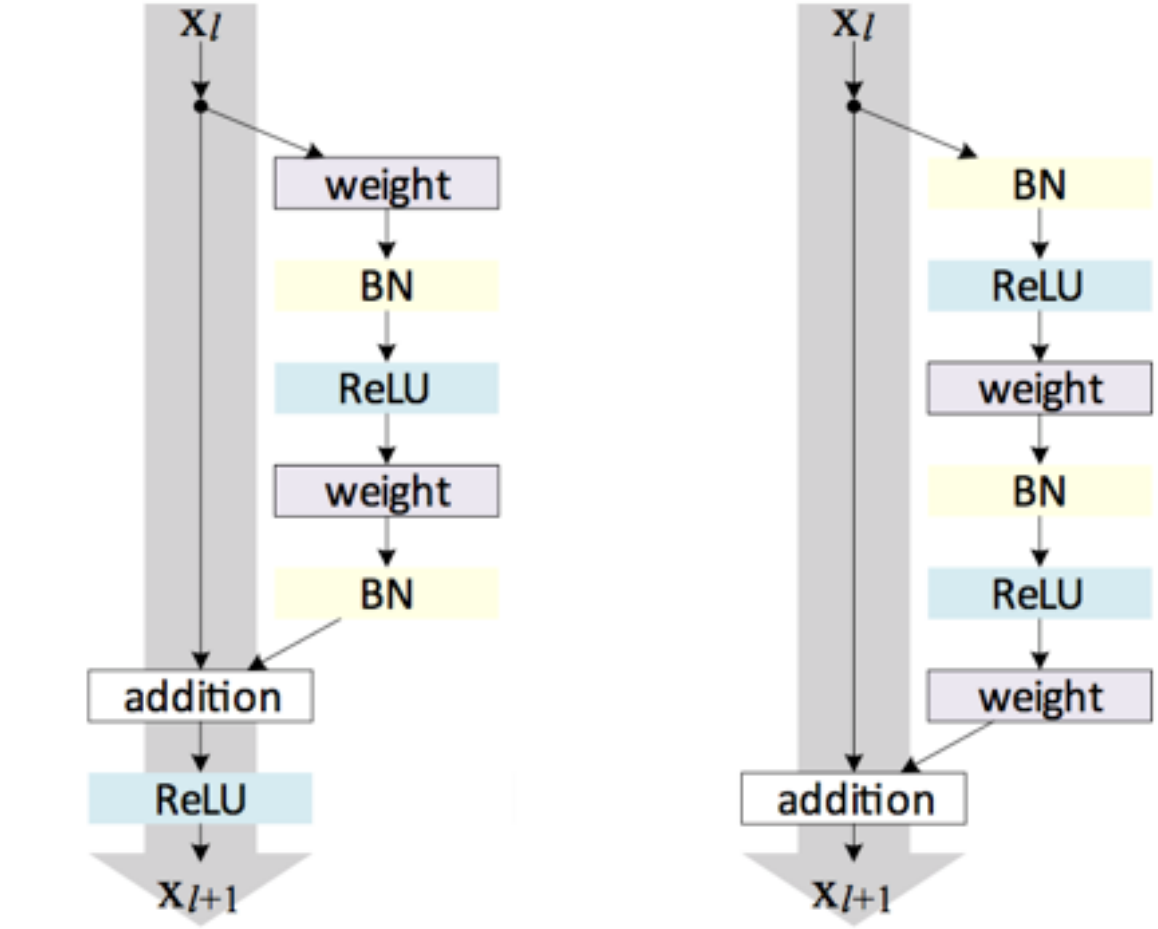
- This phenomenon is referred as the *degradation* problem:
- Alluding to the fact that although better parameter initialization techniques and batch normalization allow for deeper networks to *converge*, they often converge at a higher error rate than their shallower counterparts.
- In the limit, simply stacking more layers degrades the model's ultimate performance.
- The researcher propose a remedy to this degradation problem by introducing *residual blocks* in which intermediate layers of a block learn a residual function with reference to the block input.
- You can think of this residual function as a refinement step in which we learn how to adjust the input feature map for higher quality features.
- This compares with a "plain" network in which each layer is expected to learn new and distinct feature maps.
- In the event that no refinement is needed, the intermediate layers can learn to gradually adjust their weights toward zero such that the residual block represents an identity function.

ResNet



❑It was later discovered that a slight modification to the original proposed unit offers better performance by more efficiently allowing gradients to propagate through the network during training.

ResNet

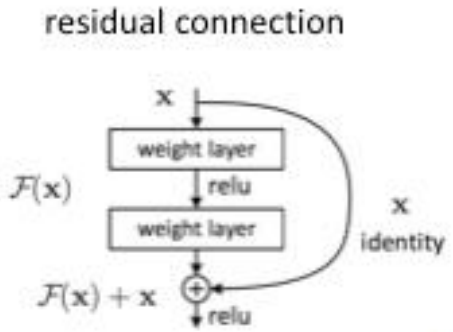


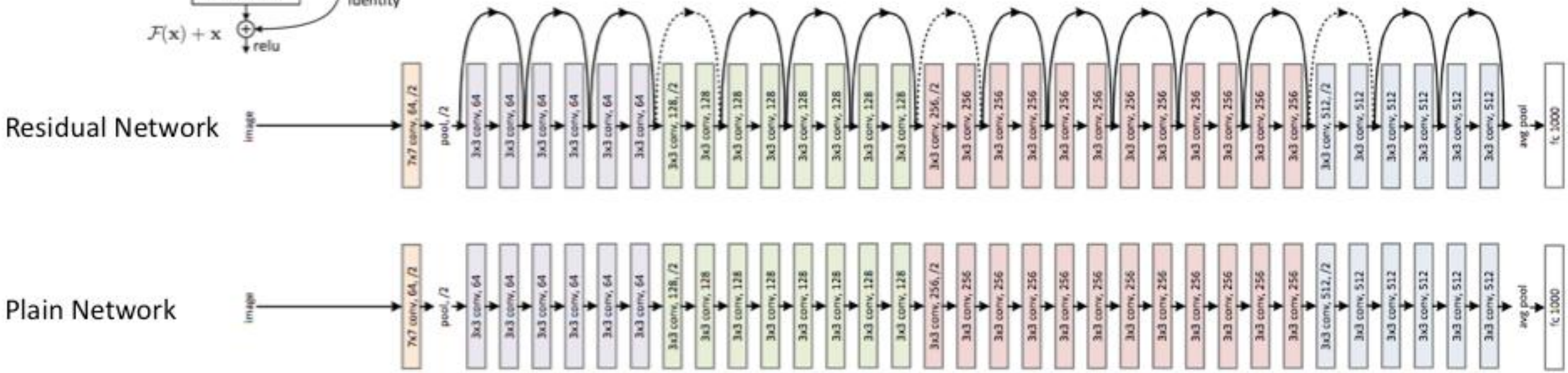
original residual unit

proposed revision

ResNet

❑**Wide residual networks:** Although the original ResNet paper focused on creating a network architecture to enable deeper structures by alleviating the degradation problem, other researchers have since pointed out that increasing the network's width (channel depth) can be a more efficient way of expanding the overall capacity of the network.



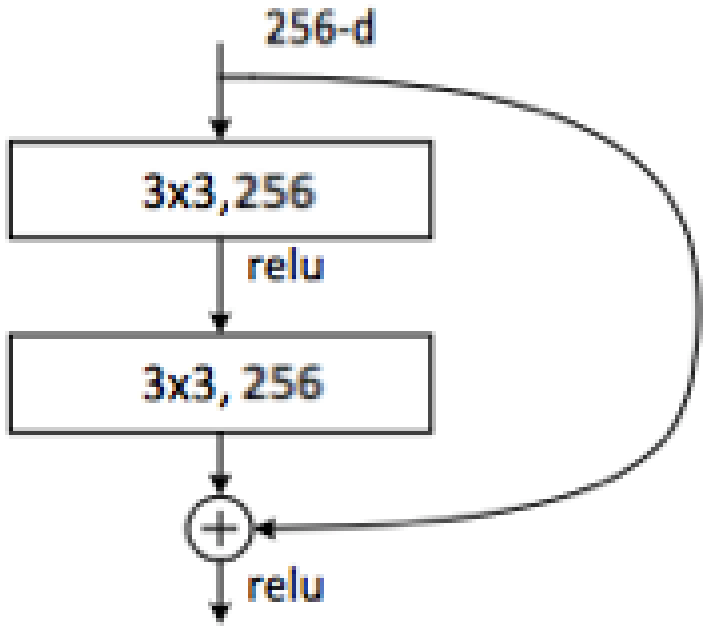


ResNet

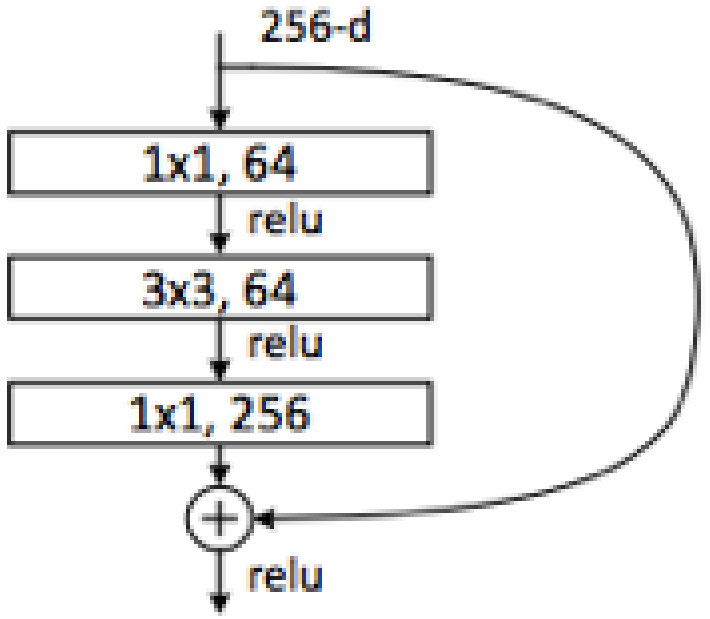
- Each colored block of layers represent a series of convolutions of the same dimension.
- The feature mapping is periodically downsampled by strided convolution accompanied by an increase in channel depth to preserve the time complexity per layer.
- Dotted lines denote residual connections in which we project the input via a 1x1 convolution to match the dimensions of the new block.
- The diagram above visualizes the ResNet 34 architecture.
- For the ResNet 50 model, we simply replace each two layer residual block with a three layer bottleneck block which uses 1x1 convolutions to reduce and subsequently restore the channel depth, allowing for a reduced computational load when calculating the 3x3 convolution.

ResNet

ResNet 34
residual block



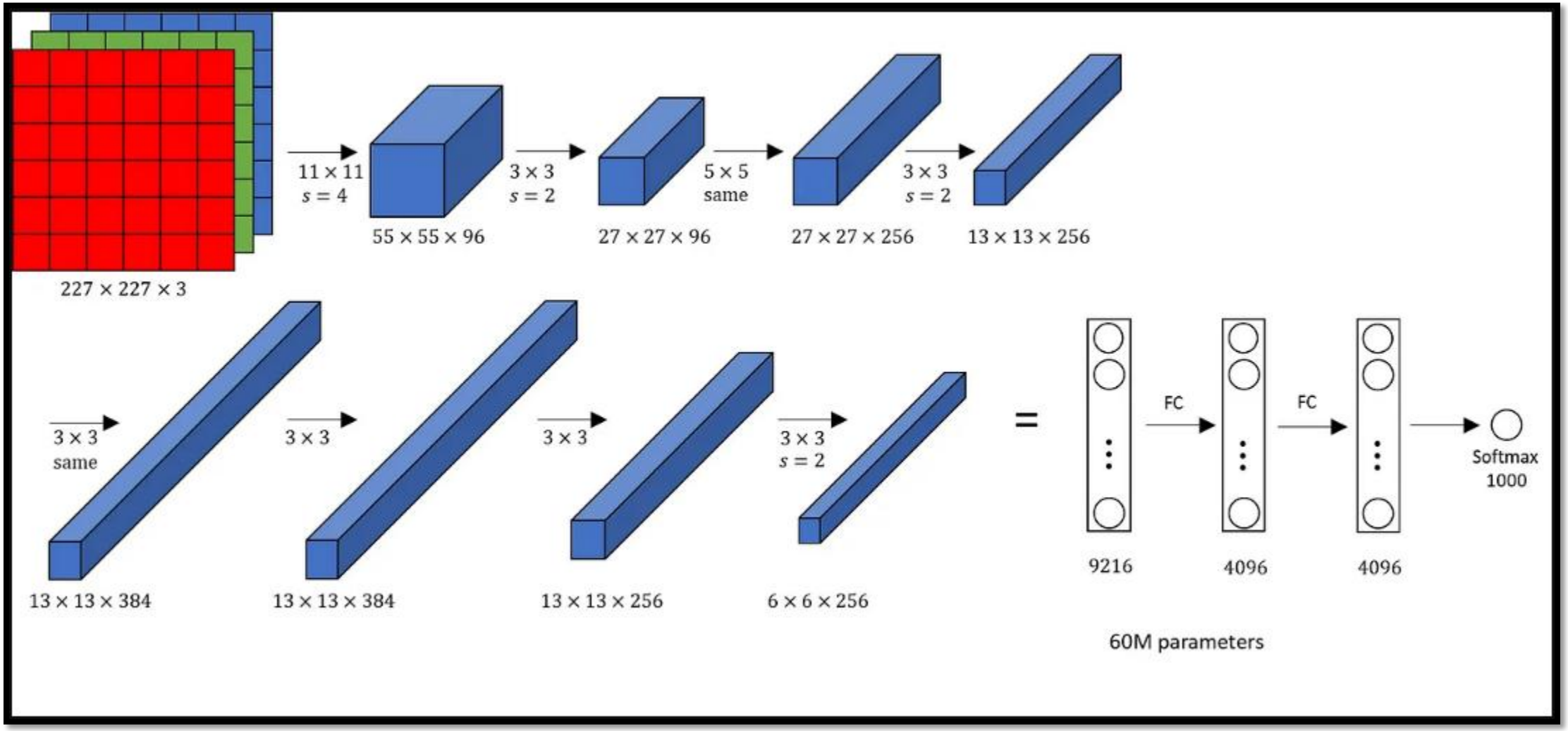
ResNet 50
residual block



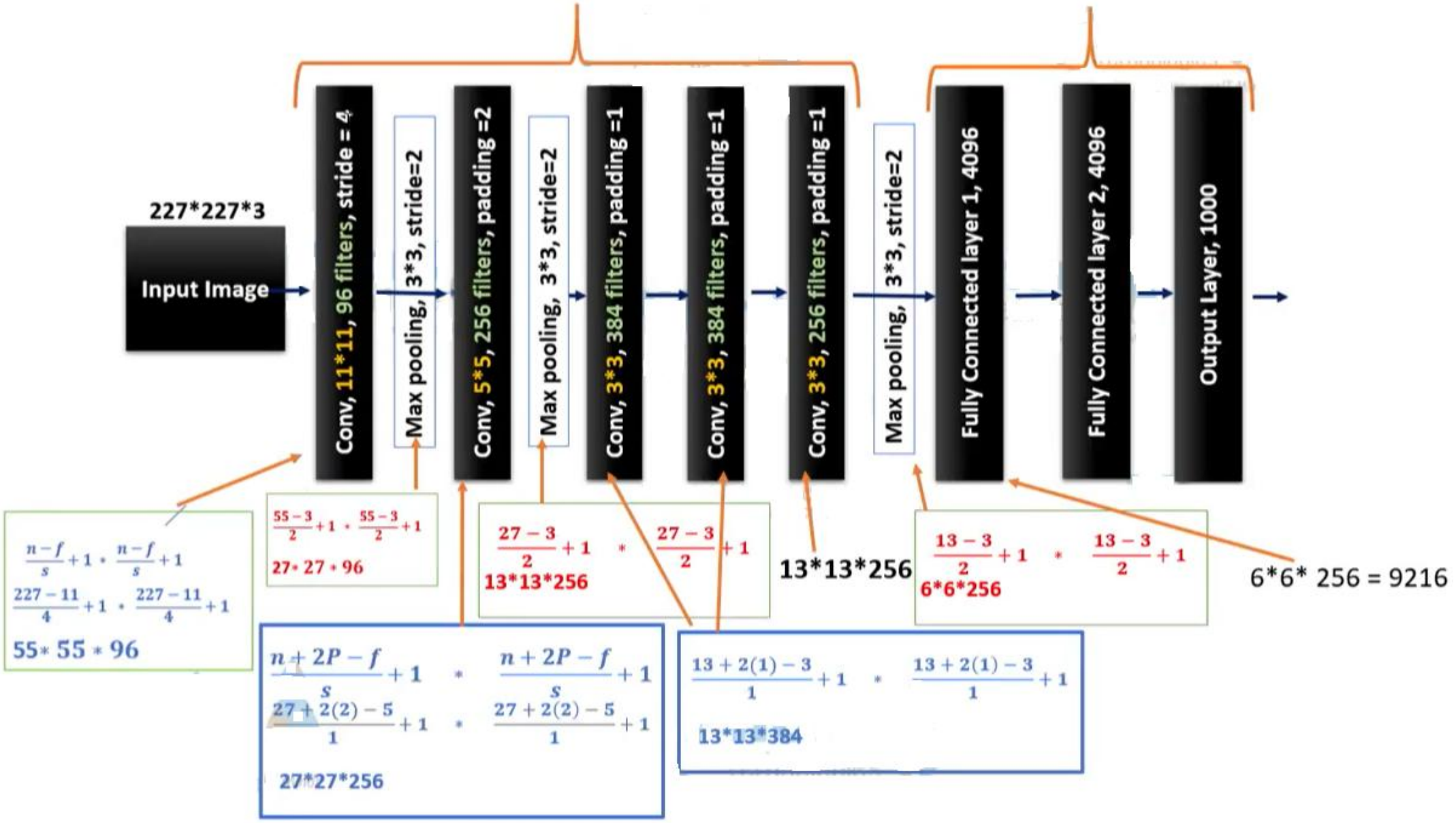
AlexNet

- This was the first architecture that used GPU to boost the training performance.
- AlexNet consists of 5 convolution layers, 3 max-pooling layers, 2 Normalized layers, 2 fully connected layers and 1 SoftMax layer.
- Each convolution layer consists of a convolution filter and a non-linear activation function called “ReLU”.
- The pooling layers are used to perform the max-pooling function and the input size is fixed due to the presence of fully connected layers.
- The input size is mentioned at most of the places as 224x224x3 but due to some padding which happens it works out to be 227x227x3.
- Above all this AlexNet has over 60 million parameters.

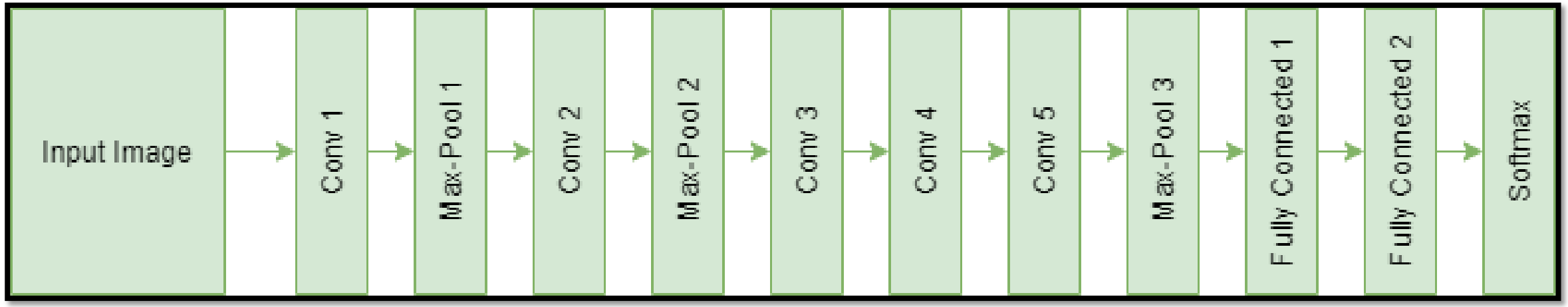
AlexNet


227 × 227 × 3
11 × 11
s = 4
55 × 55 × 96
3 × 3
s = 2
27 × 27 × 96
5 × 5
same
27 × 27 × 256
3 × 3
s = 2
13 × 13 × 256
3 × 3
same
13 × 13 × 384
3 × 3
13 × 13 × 384
3 × 3
13 × 13 × 256
3 × 3
s = 2
6 × 6 × 256
=
9216
FC
4096
FC
4096
Softmax
1000
60M parameters

AlexNet


227*227*3
Input Image
Conv, 11*11, 96 filters, stride = 4
Max pooling, 3*3, stride=2
Conv, 5*5, 256 filters, padding =2
Max pooling, 3*3, stride=2
Conv, 3*3, 384 filters, padding =1
Conv, 3*3, 384 filters, padding =1
Conv, 3*3, 256 filters, padding =1
Max pooling, 3*3, stride=2
Fully Connected layer 1, 4096
Fully Connected layer 2, 4096
Output Layer, 1000
55* 55 * 96
27* 27 * 96
13*13*256
6*6*256
6*6* 256 = 9216
27*27*256
13*13*384

AlexNet


Input Image
Conv 1
Max-Pool 1
Conv 2
Max-Pool 2
Conv 3
Conv 4
Conv 5
Max-Pool 3
Fully Connected 1
Fully Connected 2
Softmax

AlexNet

❑**Key Features:**

❑‘ReLU’ is used as an activation function rather than ‘tanh’

❑Batch size of 128

❑SGD Momentum is used as a learning algorithm

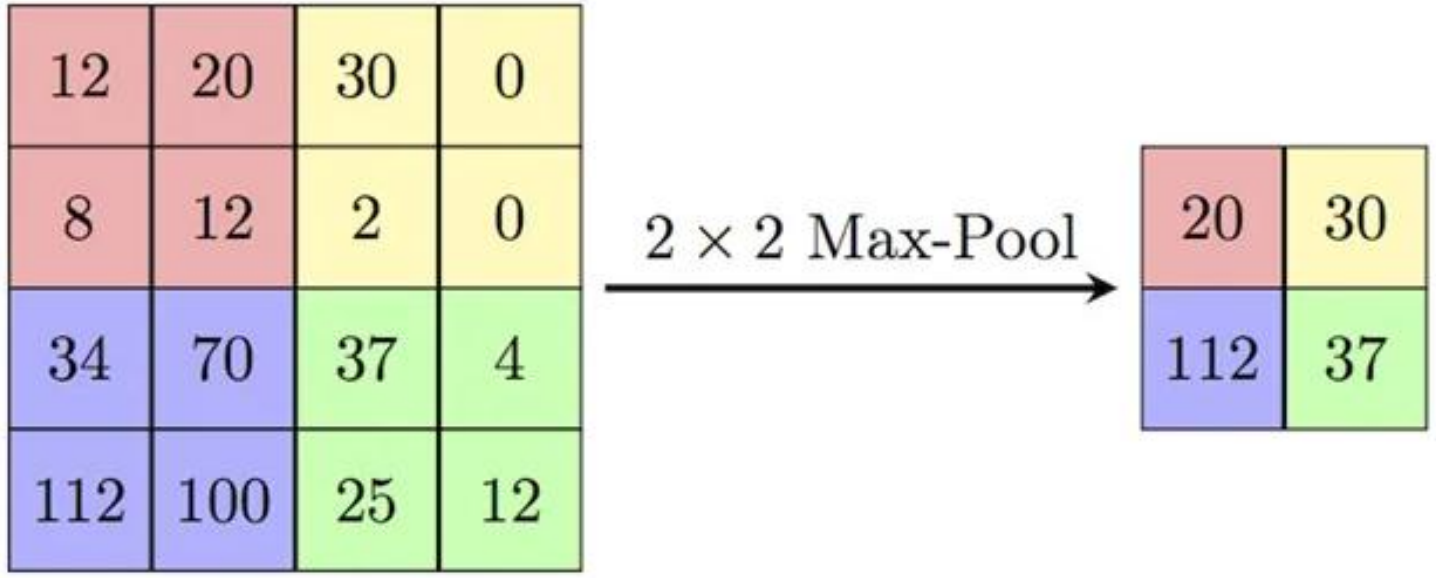
❑Data Augmentation is been carried out like flipping, jittering, cropping, colour normalization, etc.

❑AlexNet was trained on a GTX 580 GPU with only 3 GB of memory which couldn’t fit the entire network.

❑So the network was split across 2 GPUs, with half of the neurons(feature maps) on each GPU.

AlexNet

- **Max Pooling** is a feature commonly imbibed into Convolutional Neural Network (CNN) architectures.
- The main idea behind a pooling layer is to “accumulate” features from maps generated by convolving a filter over an image.
- Formally, its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computations in the network.
- The most common form of pooling is max pooling.

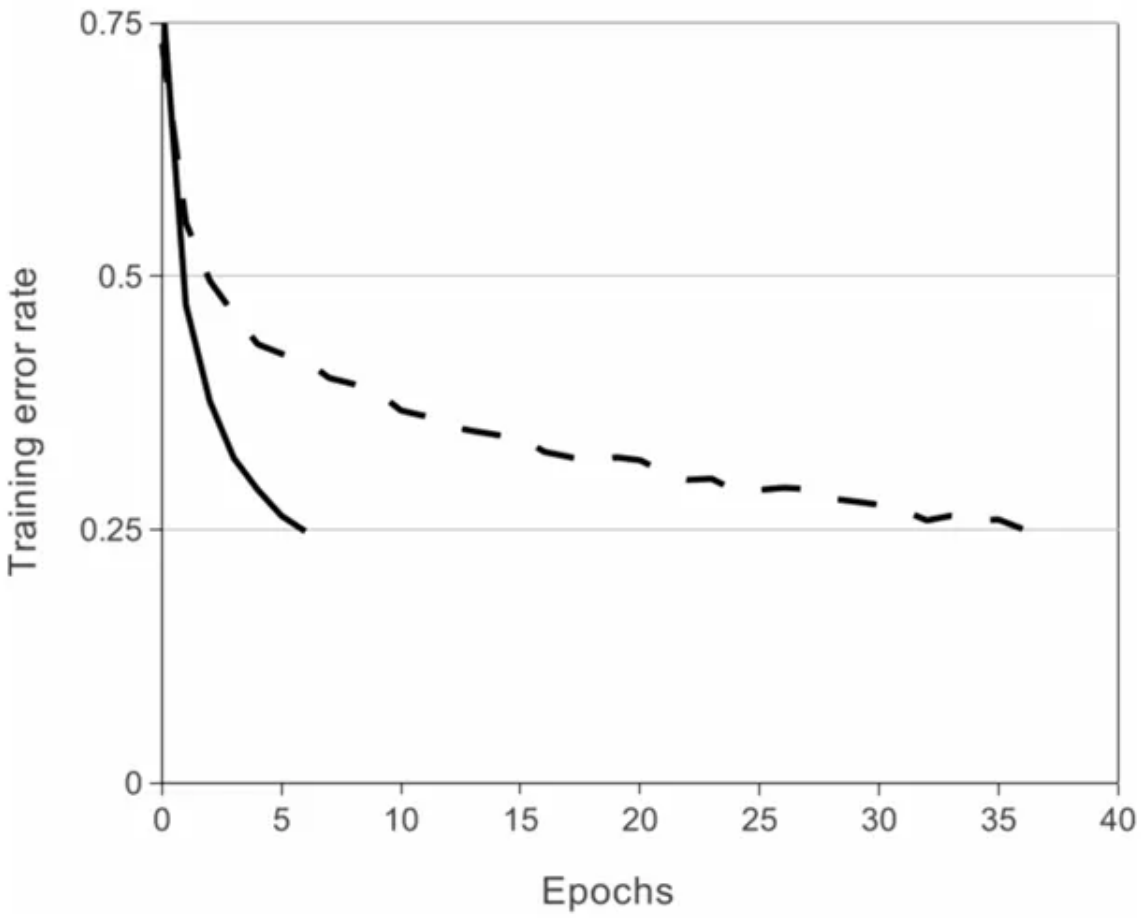


AlexNet

❑**ReLU Non-Linearity:** AlexNet demonstrates that saturating activation functions like Tanh or Sigmoid can be used to train deep CNNs much more quickly.

❑The image below demonstrates that AlexNet can achieve a training error rate of 25% with the aid of ReLUs (solid curve). Compared to a network using tanh, this is six times faster (dotted curve).

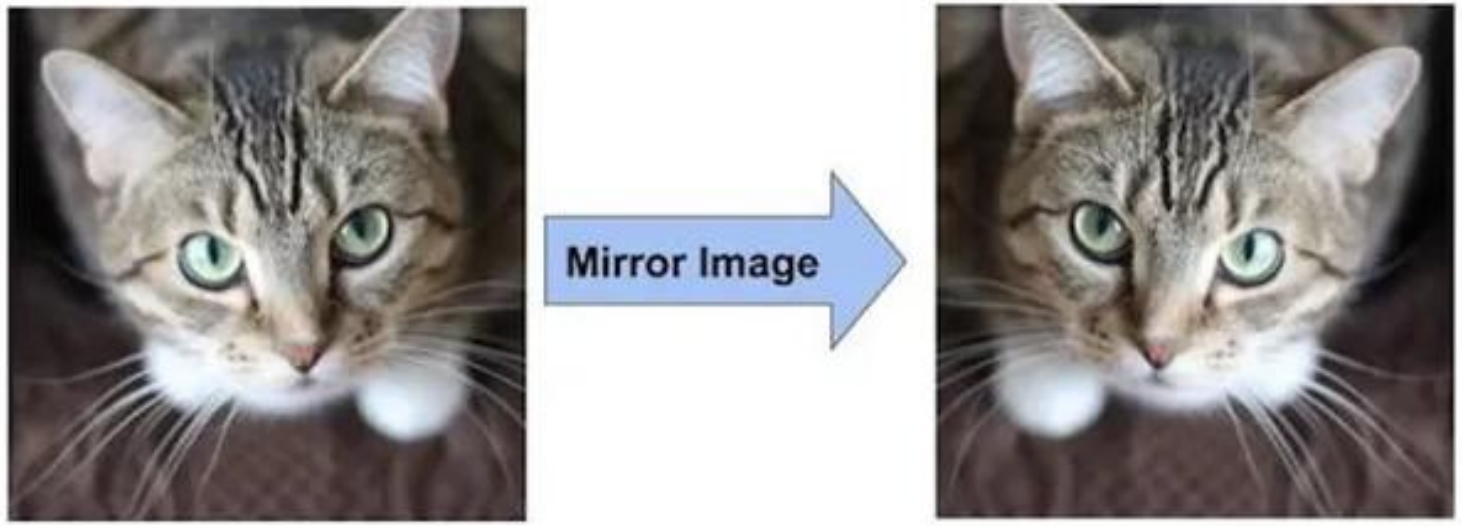
❑On the CIFAR-10 dataset, this was evaluated.



AlexNet

❑**Data Augmentation:** Overfitting can be avoided by showing Neural Net various iterations of the same image. Additionally, it assists in producing more data and compels the Neural Net to memorise the main qualities.

❑**Augmentation by Mirroring:** Consider that our training set contains a picture of a cat. A cat can also be seen as its mirror image. This indicates that by just flipping the image above the vertical axis, we may double the size of the training datasets.



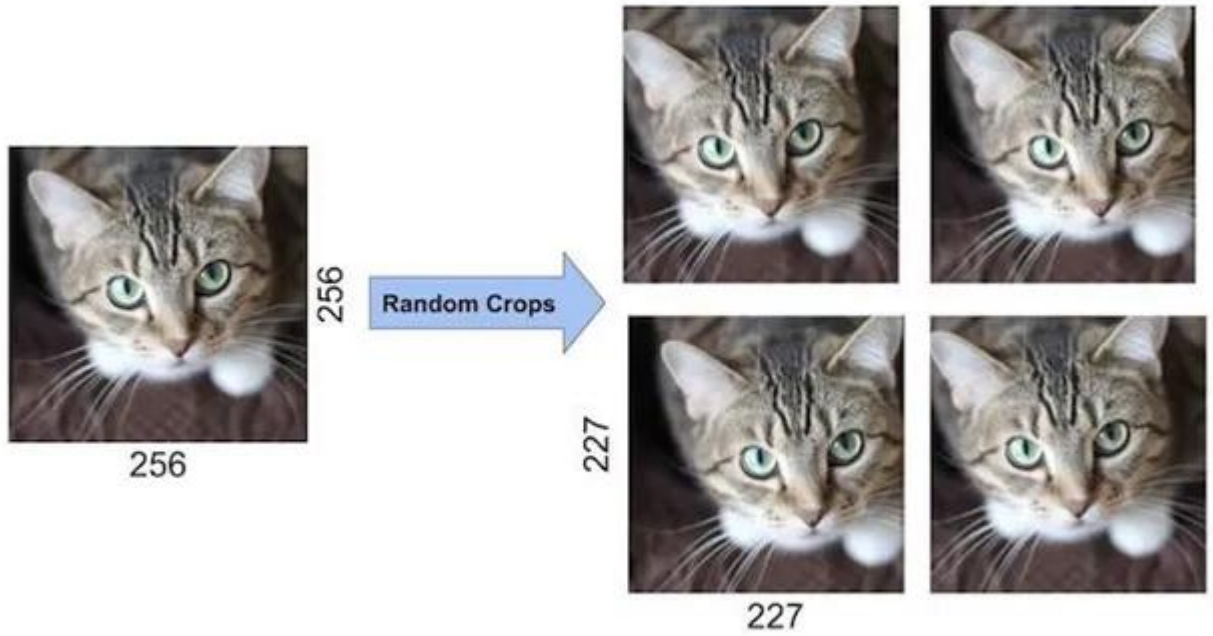
Data Augmentation by Mirroring

AlexNet

❑**Data Augmentation:**

❑**Augmentation by Random Cropping of Images:** Randomly cropping the original image will also produce additional data that is simply the original data shifted.

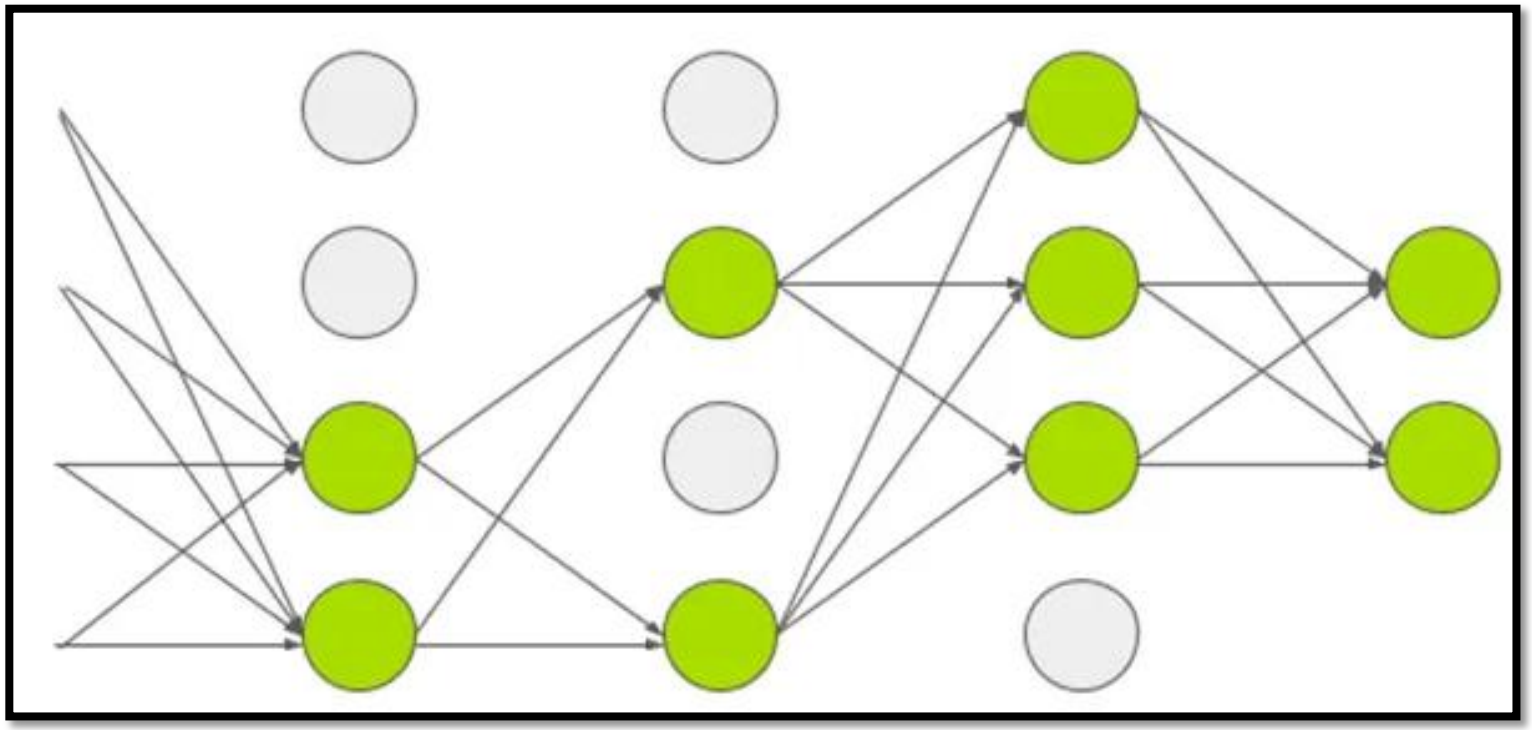
❑For the network's inputs, the creators of AlexNet selected random crops with dimensions of 227 by 227 from within the 256 by 256 image boundary. They multiplied the size of the data by 2048 using this technique.



Data Augmentation by Random Cropping

AlexNet

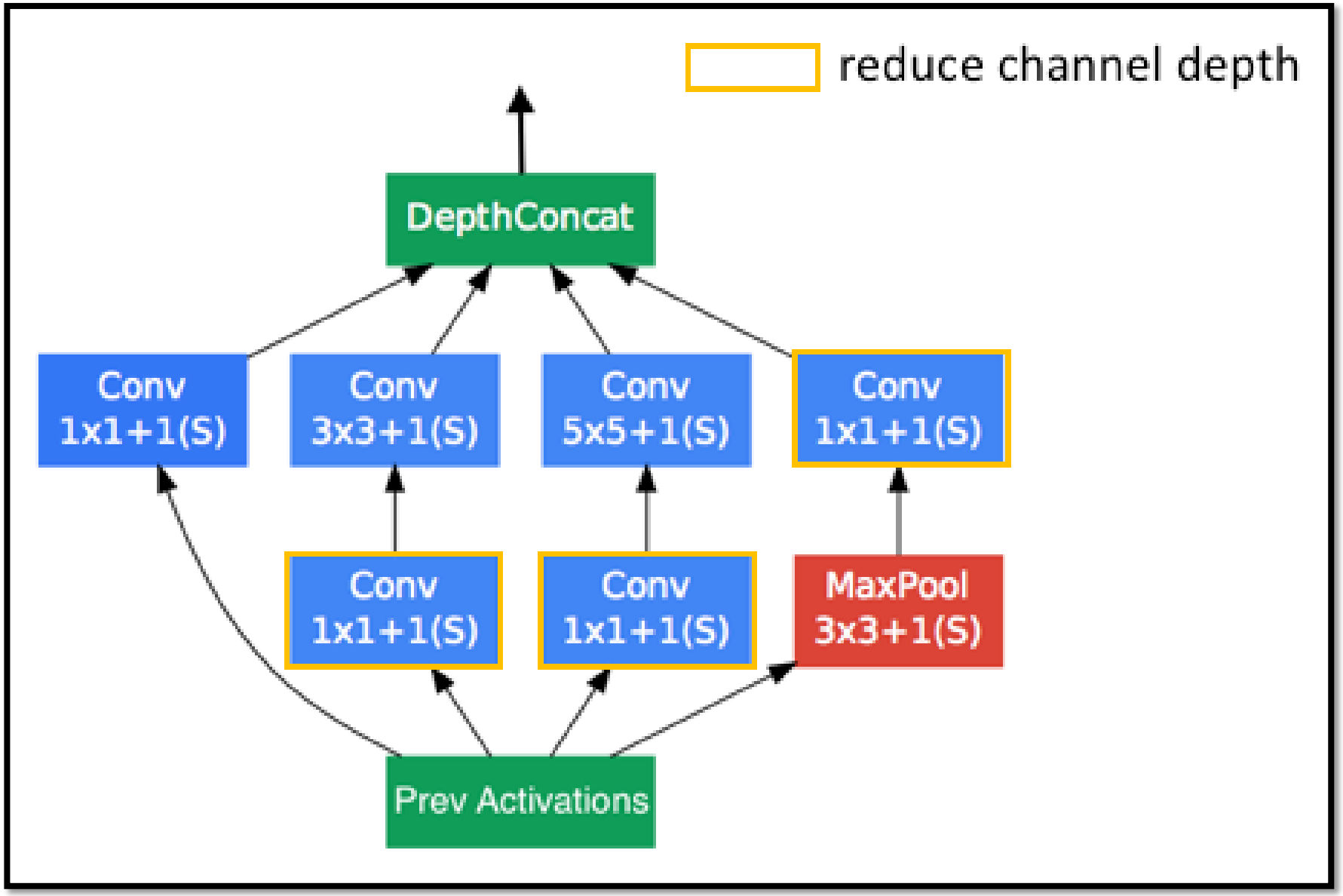
❑**Dropout:** A neuron is removed from the neural network during dropout with a probability of 0.5. A neuron that is dropped does not make any contribution to either forward or backward propagation. As seen in the graphic below, each input is processed by a separate Neural Network design. The acquired weight parameters are therefore more reliable and less prone to overfitting.



InceptionNet (GoogleNet)

- In 2014, researchers at Google introduced the Inception network which took first place in the 2014 ImageNet competition for classification and detection challenges.
- The model is comprised of a basic unit referred to as an "Inception cell" in which we perform a series of convolutions at different scales and subsequently aggregate the results.
- In order to save computation, 1x1 convolutions are used to reduce the input channel depth.
- For each cell, we learn a set of 1x1, 3x3, and 5x5 filters which can learn to extract features at different scales from the input.
- Max pooling is also used, albeit with "same" padding to preserve the dimensions so that the output can be properly concatenated.

InceptionNet (GoogleNet)

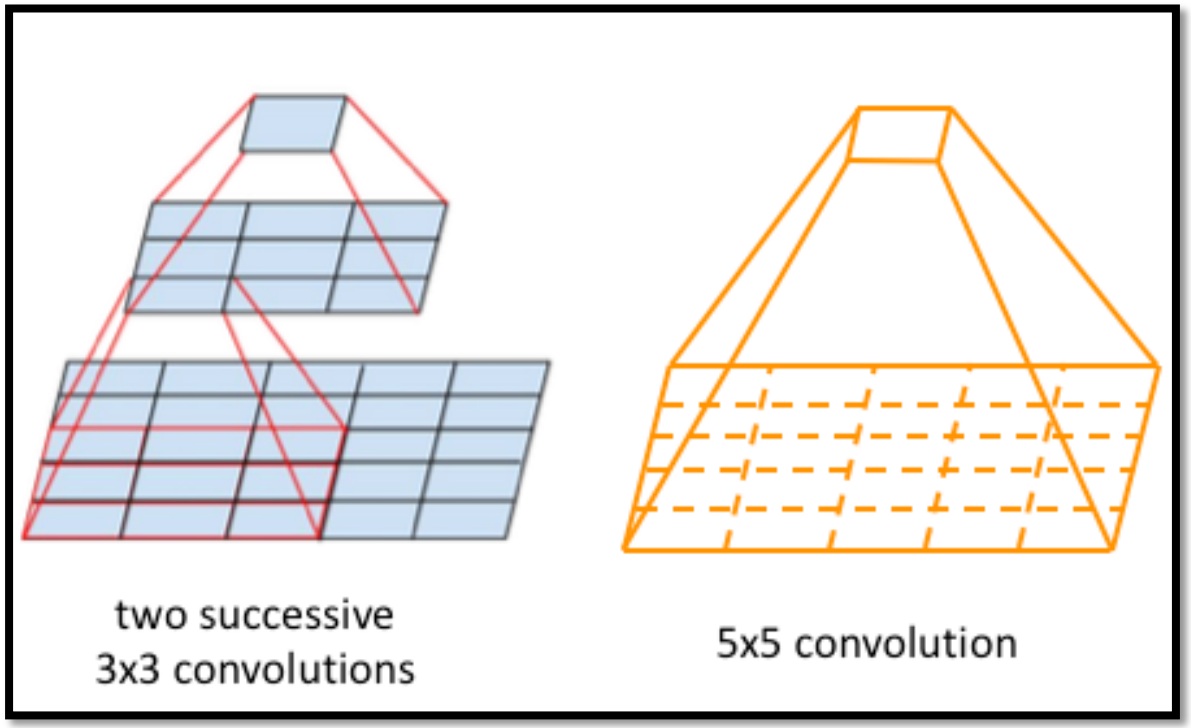


InceptionNet (GoogleNet)

❑These researchers published a follow-up paper which introduced more efficient alternatives to the original Inception cell.

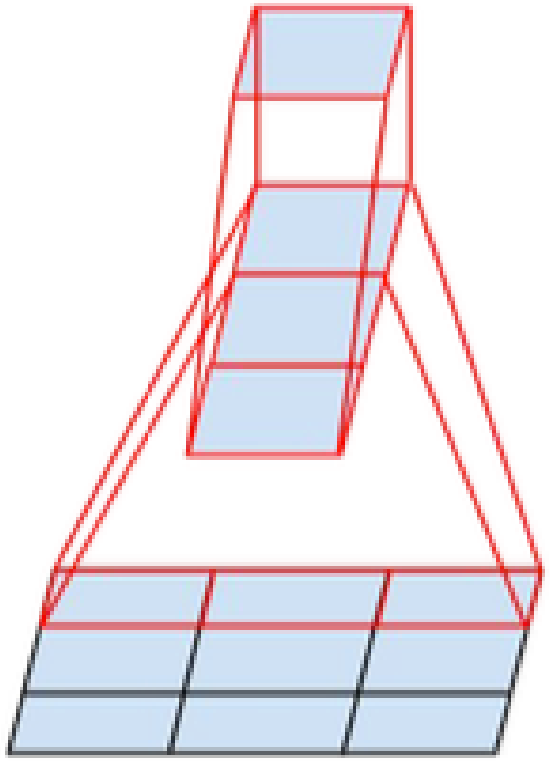
❑Convolutions with large spatial filters (such as 5x5 or 7x7) are beneficial in terms of their expressiveness and ability to extract features at a larger scale, but the computation is disproportionately expensive.

❑The researchers pointed out that a 5x5 convolution can be more cheaply represented by two stacked 3x3 filters.

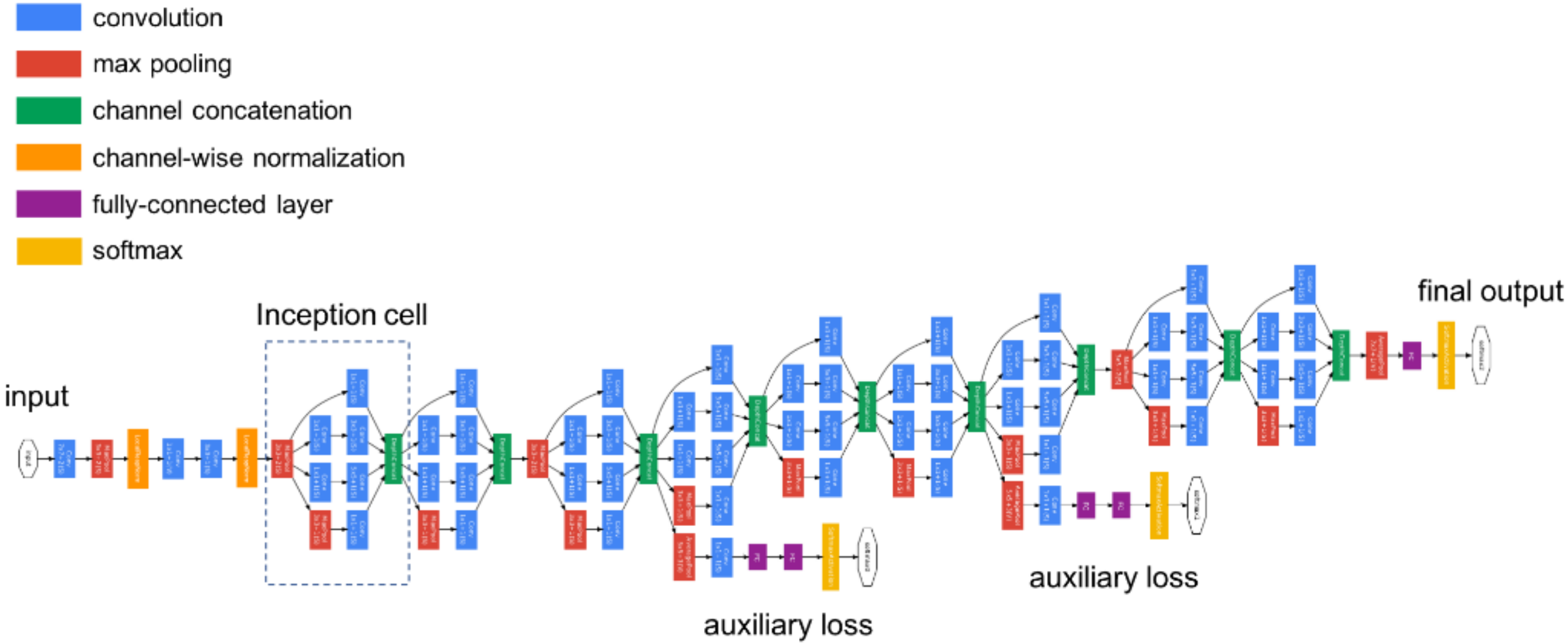


InceptionNet (GoogleNet)

- Whereas a 5×5×c filter requires 25c parameters, two 3×3×c filters only require 18 parameters.
- In order to most accurately represent a 5x5 filter, we shouldn't use any nonlinear activations between the two 3x3 layers.
- However, it was discovered that "linear activation was always inferior to using rectified linear units in all stages of the factorization.“
- It was also shown that 3x3 convolutions could be further deconstructed into successive 3x1 and 1x3 convolutions.
- Generalizing this insight, we can more efficiently compute an n×n convolution as a 1×n convolution followed by a n×1 convolution.



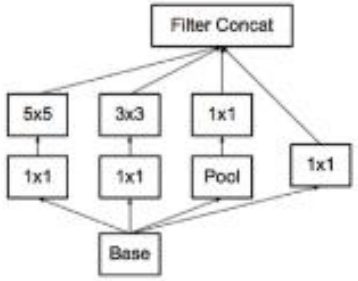
InceptionNet (GoogleNet)

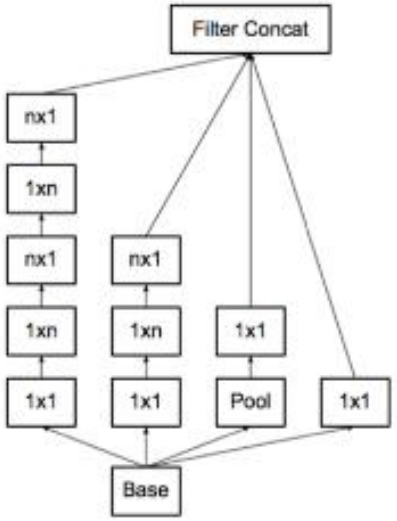


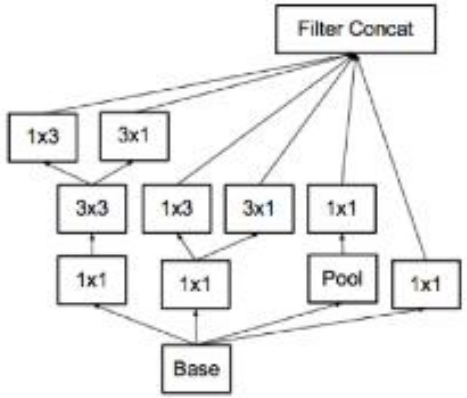
InceptionNet (GoogleNet)

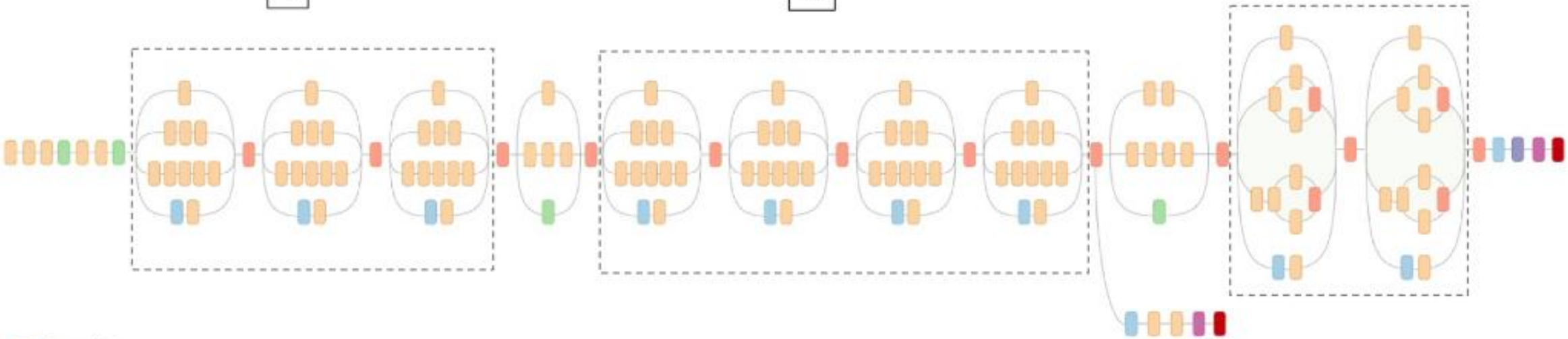
- In order to improve overall network performance, two auxiliary outputs are added throughout the network.
- It was later discovered that the earliest auxiliary output had no discernible effect on the final quality of the network.
- The addition of auxiliary outputs primarily benefited the end performance of the model, converging at a slightly better value than the same network architecture without an auxiliary branch.
- It is believed the addition of auxiliary outputs had a regularizing effect on the network.
- A revised, deeper version of the Inception network which takes advantage of the more efficient Inception cells is shown in next page.

InceptionNet (GoogleNet)









Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax

Note for Students

- This power point presentation is for lecture, therefore it is suggested that also utilize the text books and lecture notes.