Convolutional Neural Network

MET CS Generative AI

Reza Rawassizadeh

Popular Image Dataset

- There are some very popular datasets used to evaluate computer vision algorithms.

- One is a handwritten number dataset called MNIST (stayed for Modified National Institute of Standards and Technology database) that to date is the most popular image dataset in use for experimenting with different types of machine learning algorithms, especially image recognition ones.

- There are a few popular datasets including ImageNet [Deng ‘09], mtCars (vehicle data), IRIS (flower shape), CIFAR-10/CIFAR100 (tiny images of different objects), and Fashion MNIST (cloths images), which are used for experimenting or benchmarking machine learning algorithm.

Outline

- Convolutional Neural Network
- Convolution and Cross-correlation
- How to Build CNN
- Different Types of Convolutions
- CNN Models for Classification

Outline

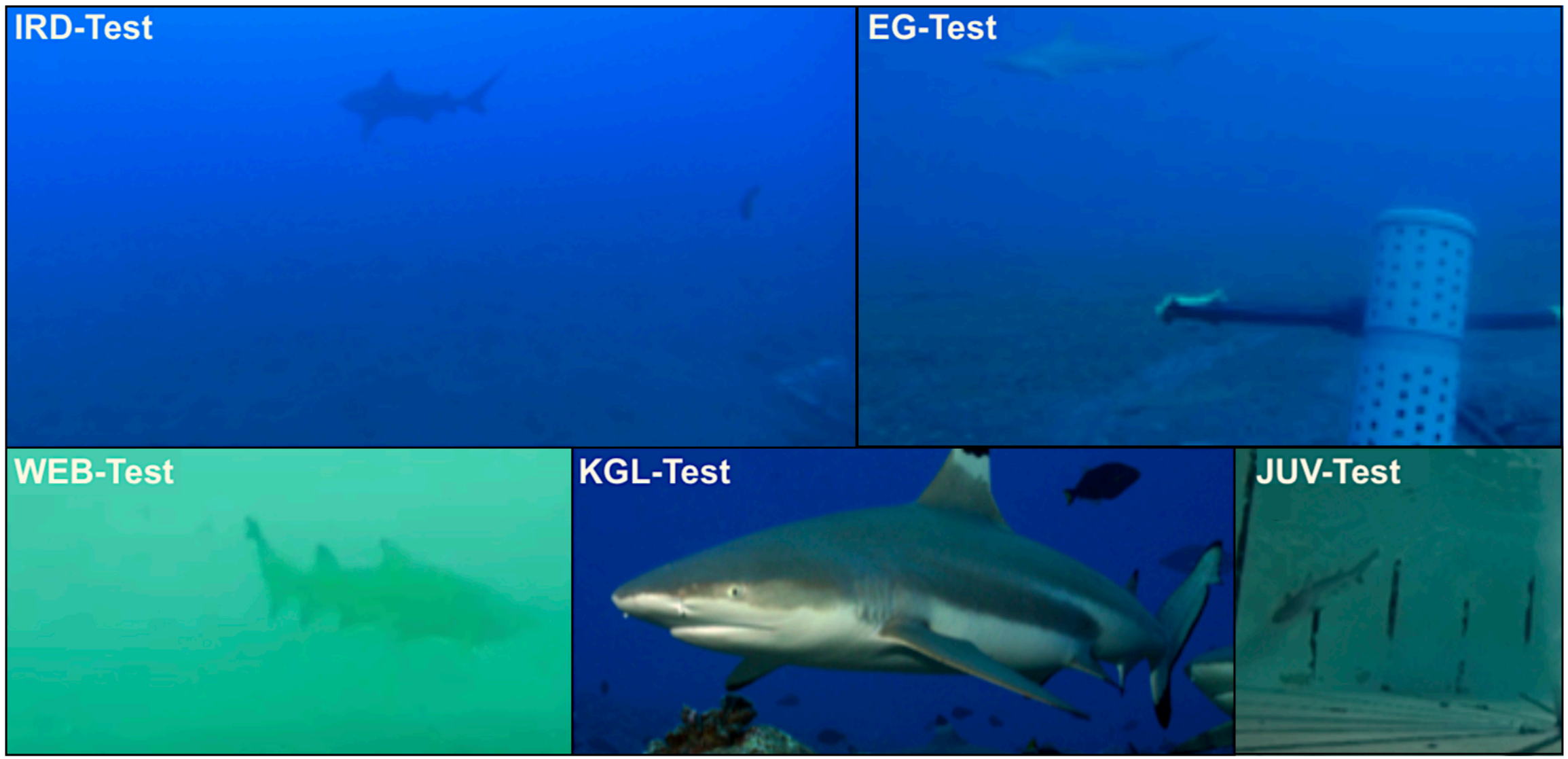
- **Convolutional Neural Network**
- Convolution and Cross-correlation
- How to Build CNN
- Different Types of Convolutions
- CNN Models for Classification

Convolutional Neural Network (CNN)

- CNN architecture preserves the spatial structure of the input data, and preserving the spatial structure of data makes CNN very popular (probably the most popular) for computer vision applications. However, CNNs are not scaling or rotation invariant.

- CNN operates by assigning multiple image filters to an input image and these filters enable the algorithm to accurately classify the image.

Scale-invariant Example

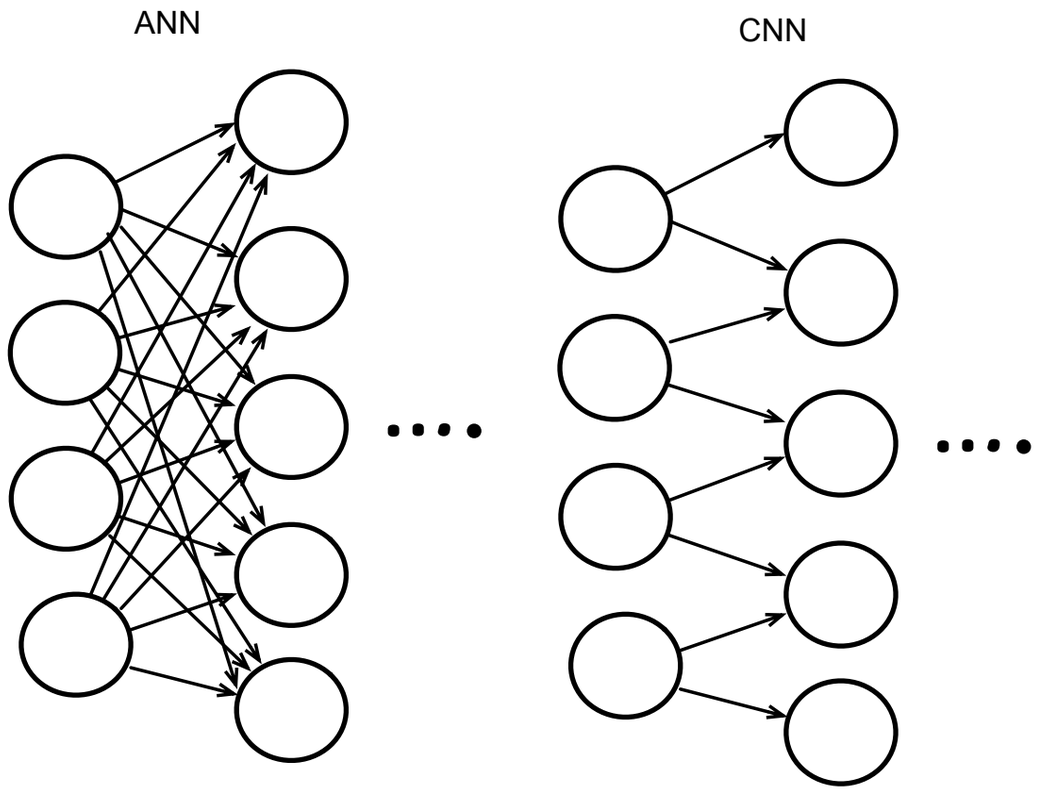


CNN Advantage over ANN

1. First, ANN uses densely connected layers; in CNN, each neuron is connected to a limited number of neurons in the neighbor layer.

- Having all neurons connected to all other neurons of the neighbor layer causes ANN to learn only **global patterns** in the image (the result of having densely connected layers where every output neuron is connected to every input neuron).

- However, CNN does not use densely connected layers, and by using a window over the image, it can learn the **local patterns**.



CNN Advantage over ANN

2. Second CNN is learning the *hierarchical features of the image*.

- For example, it separates a chicken image into a peak, eyes, wings, etc.

- This allows the CNN to learn complex structures and be able to classify similar images, which is not possible with traditional algorithms. In simple words, if two images are taken from a different angles and in different light settings, still a CNN algorithm can classify them.

CNN Input

- Typically, a CNN network takes an input of a tensor, which includes image height, width, and channels (three data for red, green, and blue in RGB mode or three data for hue, saturation, and luminance in HSL mode).

Convolution

- Convolution is the process of combining two functions. As a result getting a new function, which has a different function shape than the two input functions.

- In the context of neural networks, convolution is the process of multiplying matrices (or tensors), and the result will be a new matrix (or tensor).

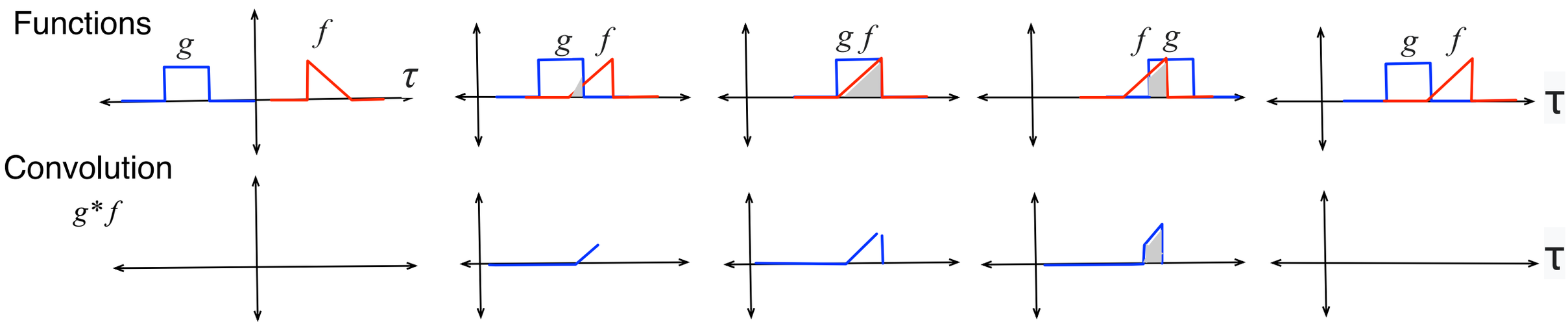
Convolution

- Mathematically convolution is written as *the integral product of two functions, but one of these functions is reversed and shifted*.

$$(g * f)(t) = \int_{-\infty}^{\infty} g(\tau) f(t - \tau) d_{\tau}$$

- In simple words, the integral specifies the amount of overlap of one function as it is shifted over another function.

- t is the current time and τ is used to specify the size of the shift.



Cross Correlation

- **Cross correlation** is similar to convolution, but it measures how similar are two different functions. In other words, it is similar to the convolution with the difference that it does not flip one of the functions, convolution function flips one function (flipping means negative all of its variables e.g. $f(x) = x$ and flipping it will be $f'(x) = -x$)

$$(g \otimes f)(t) = \int_{-\infty}^{\infty} g(\tau) f(t + \tau) d_{\tau}$$

2D Convolution and 2D Cross Correlation

When we work with image, the data is considered discrete and multidimensional. Assuming f is one function (image), and g is the other function (kernel), the convolution equation for 2D data can be written as follows:

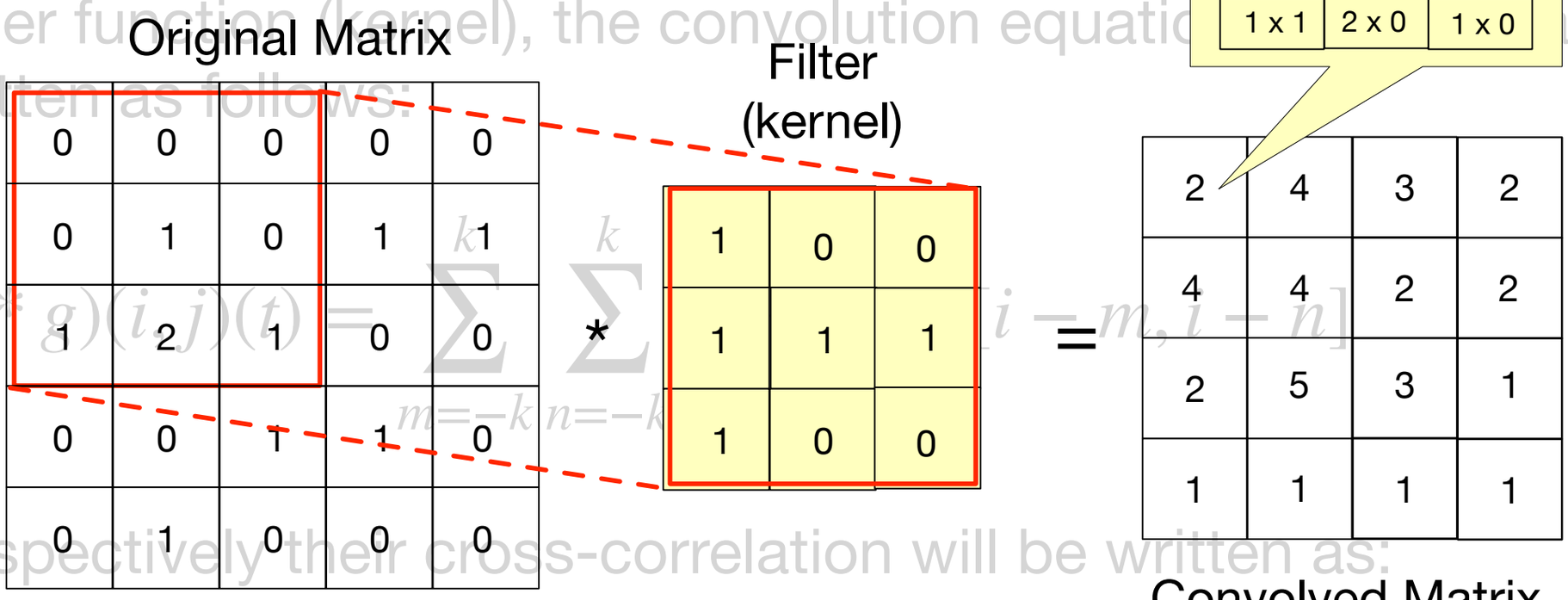
$$(f * g)(i,j)(t) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} f[m,n] \,.\, g[i-m, i-n]$$

Respectively their cross-correlation will be written as:

$$(f \otimes g)(i,j)(t) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} f[m,n] \,.\, g[i+m, i+n]$$

2D Convolution and 2D Cross Correlation

When we work with image, the data is considered discrete and multidimensional. Assuming f is one function and g is the other function (kernel), the convolution equation can be written as follows:



$$(f * g)(i,j)(t) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} f[m,n] \cdot g[i-m, i-n]$$

Respectively their cross-correlation will be written as:

$$(f \otimes g)(i,j)(t) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} f[m,n] \cdot g[i+m, i+n]$$

Outline

- Convolutional Neural Network
- Convolution and Cross-correlation
- **How to Build CNN**
- Different Types of Convolutions
- CNN Models for Classification

CNN Algorithm Step by Step

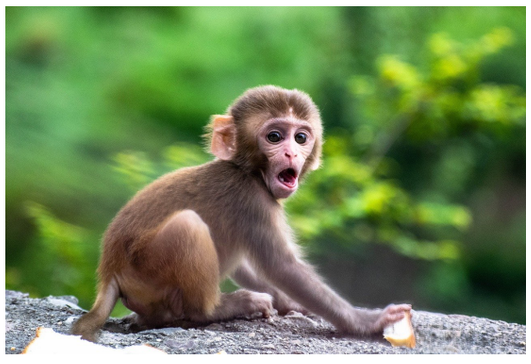
1. Convolution: Using kernels (filters) to construct feature maps
2. Non-Linear transformation
3. Pooling
4. Flattening
5. Fully Connected layer (Dense Layer)

Step 1- Feature Map Construction

The task of kernels or filters in the CNN is feature detection. Kernels are similar to the image filter that we are using on photo editing software such as blurring an image, sharpening an image, etc.

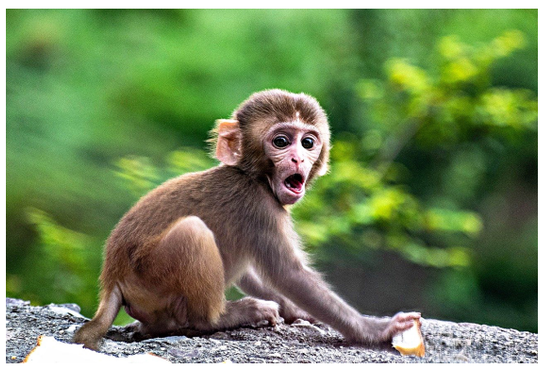
Filters are matrices (tensors for colored images) that are multiplied by a given image and the resulting image will be the transformed image that is blurred, sharpen, etc.

Original

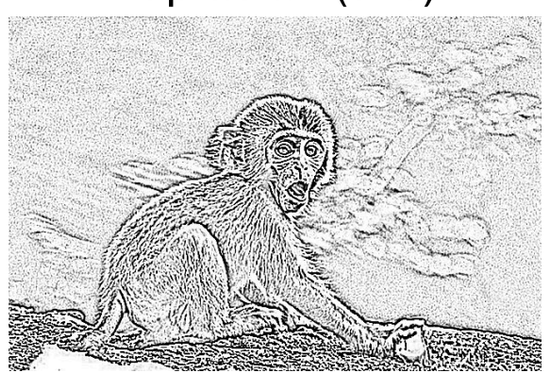


Blur

Sharpen



Sobel
Laplacian (5x5)

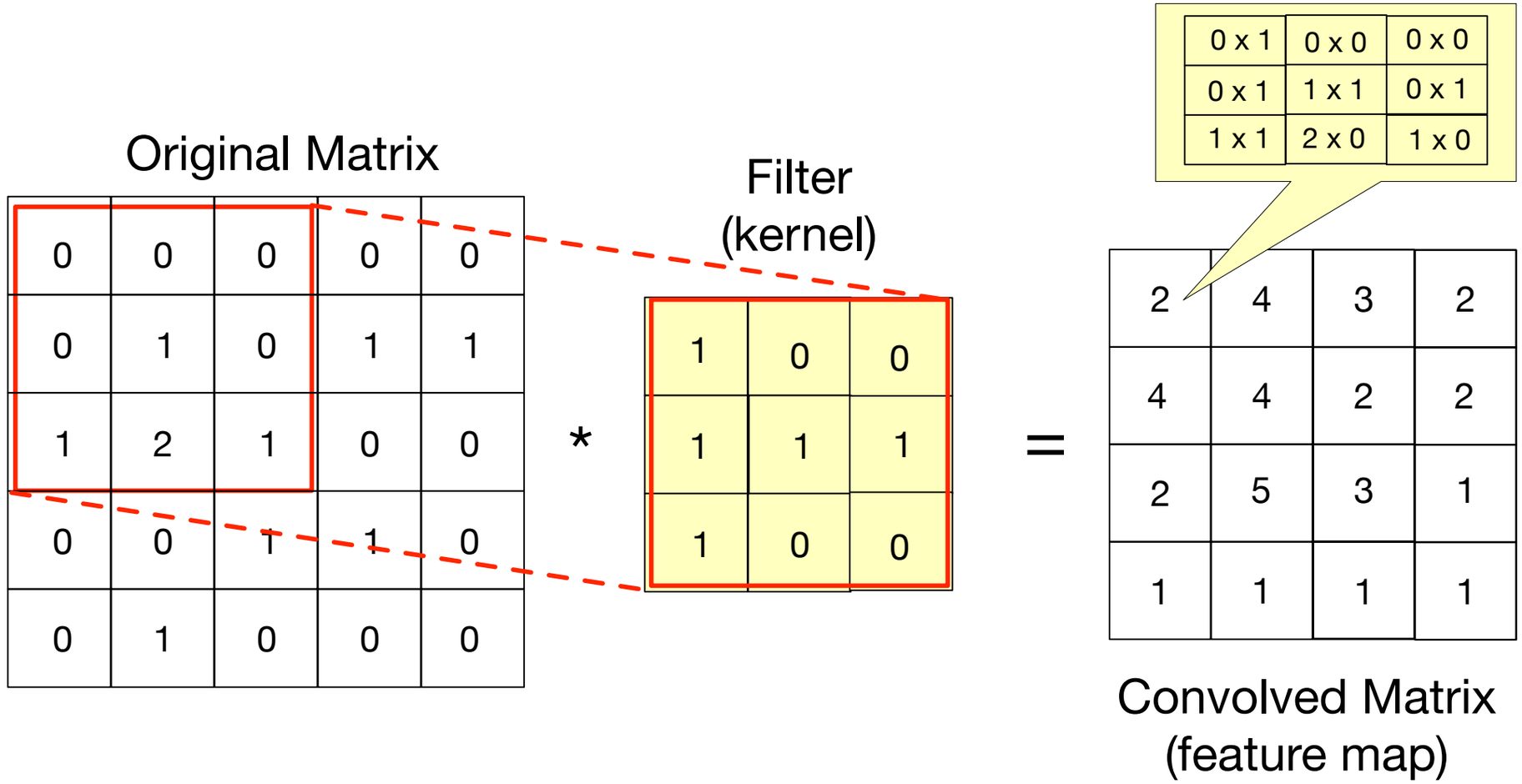


Step 1- Feature Map Construction

In the context of CNN, we call these filters **convolutional kernels** (or filters) and the resulting image is a **convoluted image**, thus we can write the following:

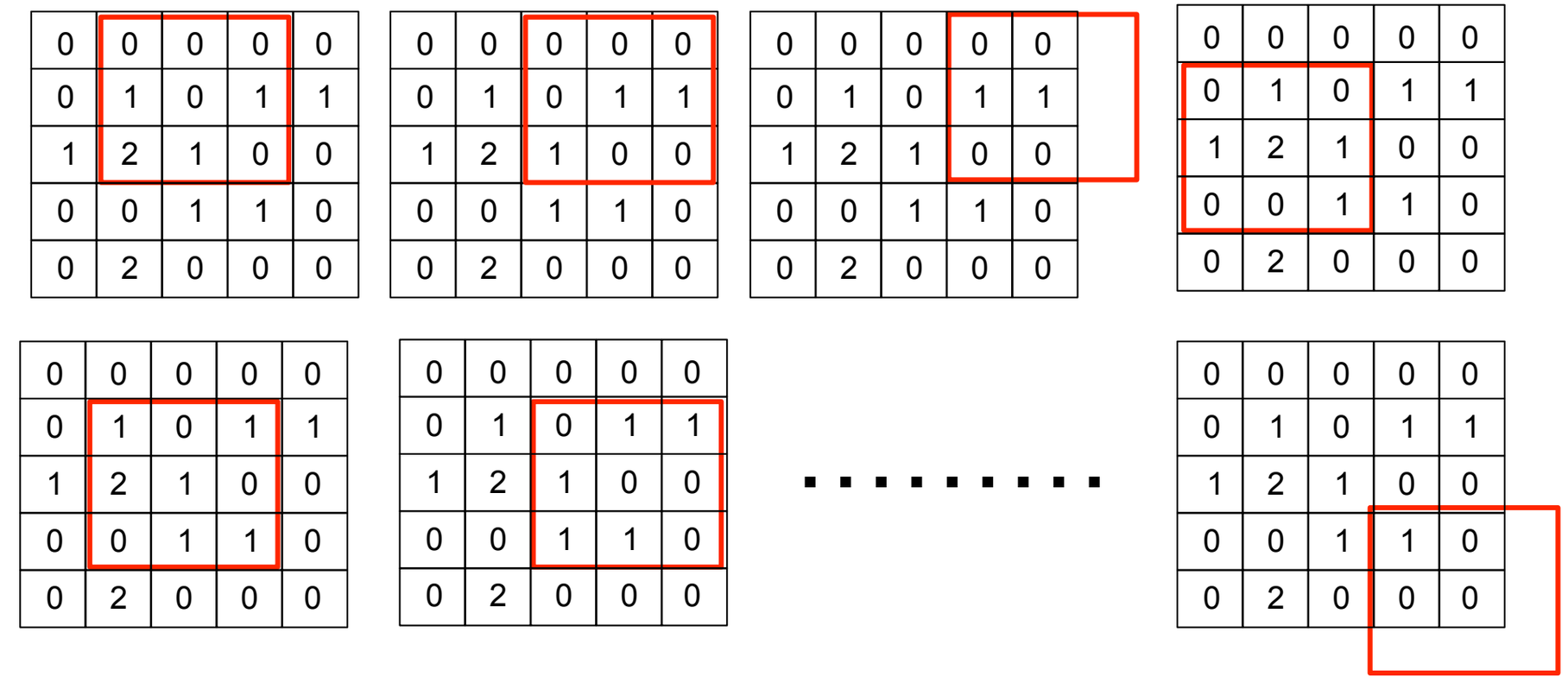
*input image * kernel (feature detector) = convolved image (feature map)*

The * operator here is referred to as “element-wise” multiplication.



Step 1- Feature Map Construction (Stride and Window size)

- The number of pixels that a window move is called **stride.** In this figure we move the window one pixel in the X and then Y direction. Therefore, we say the stride is equal to one. Usually, the stride is set to two pixels, and the larger the stride the convolved image is getting smaller.



Step 1- Feature Map Construction (Filter Example)

Sharpen

0	-1	0
-1	5	-1
0	-1	0

Blur

0.05	0.1	0.05
0.1	0.25	0.1
0.05	0.1	0.05

Left Sobel

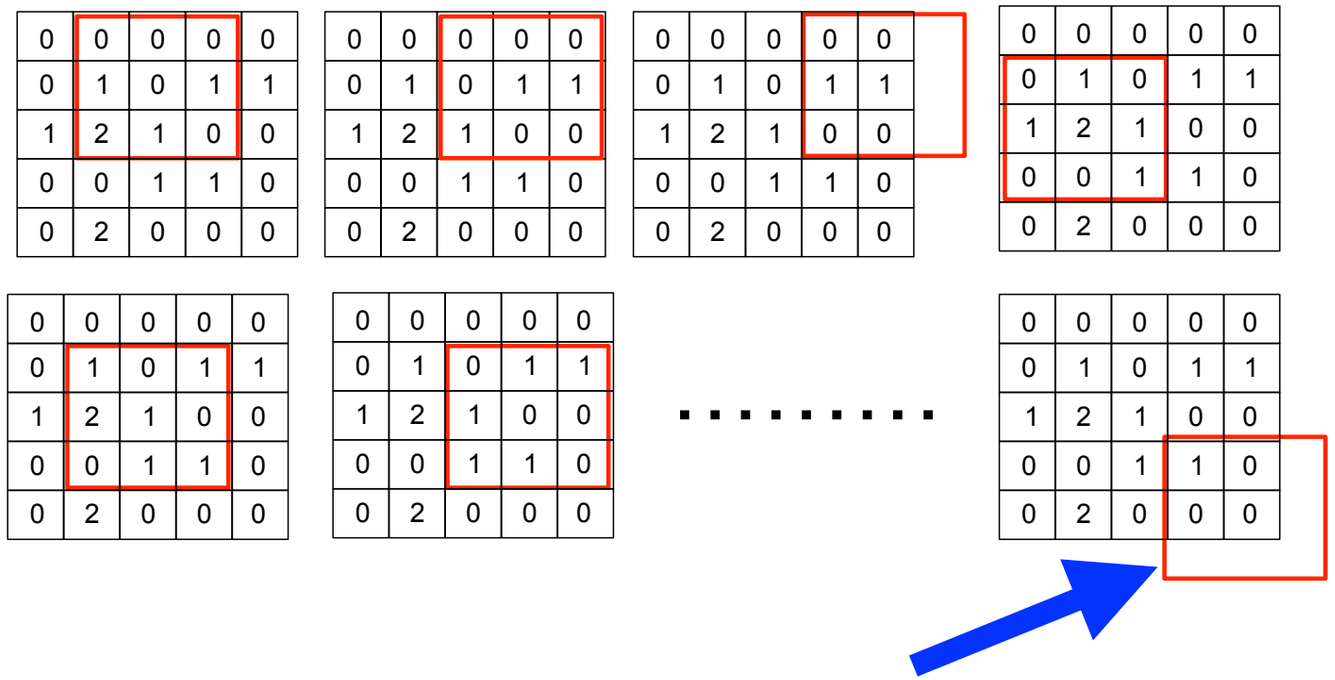
1	0	-1
1.5	0	-1.5
1	0	-1

Emboss

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

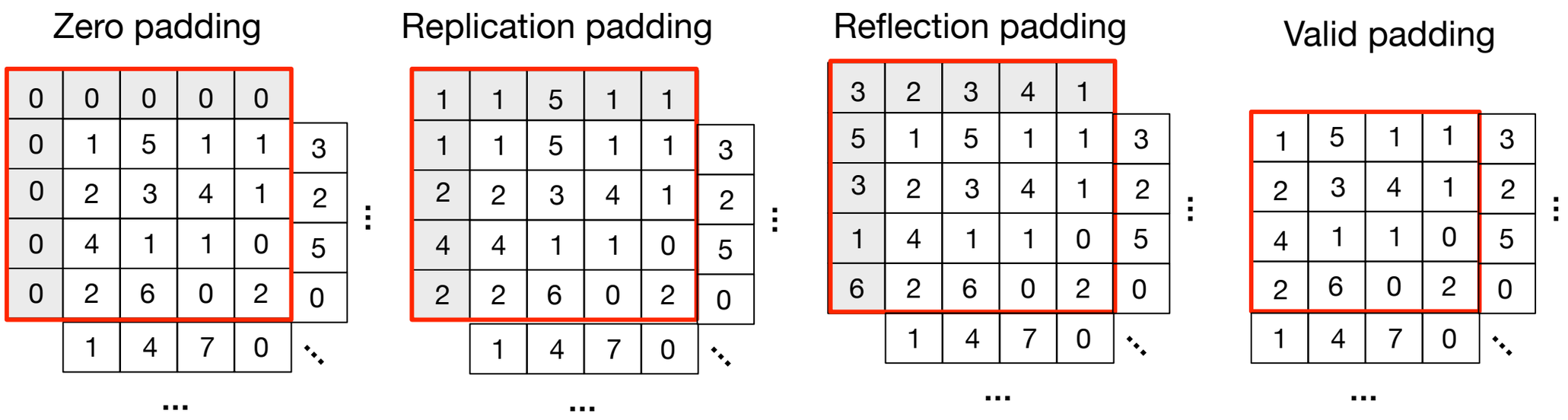
- Some sample image filters. Sharpen and Blur are clear, Left Sobel is used to showing differences between each pixel and its adjacent pixel on the left. Emboss creates an illusion of depth for the viewer of the image.

Step 1- Feature Map Construction (Padding)



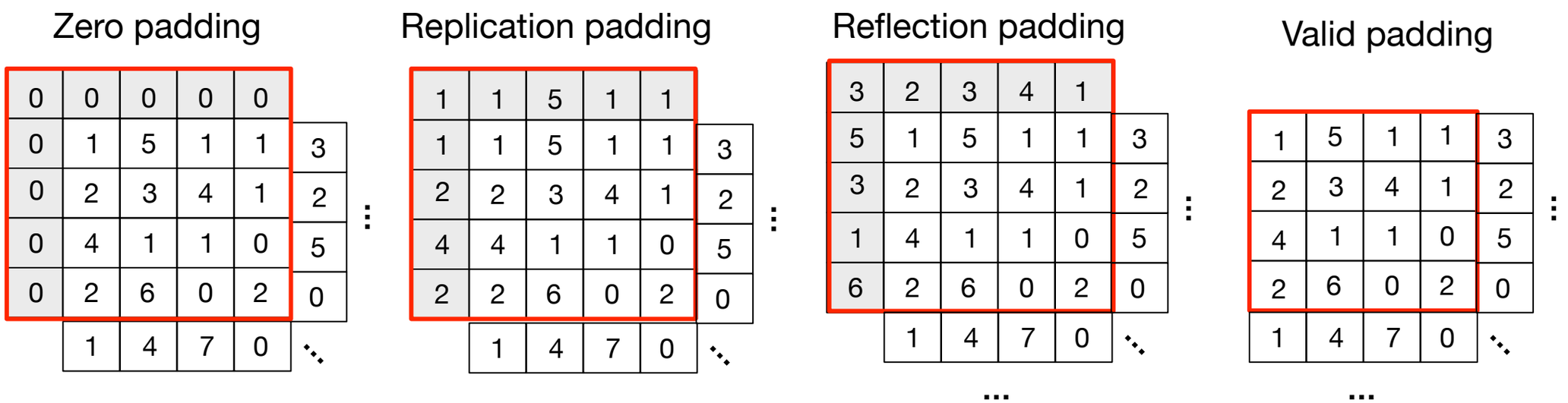
- In the above example, the window is fallen outside the image matrix, and thus there is no pixel there, which causes losing the border of the image on the convolved image. To avoid losing the borders, we can use a process called **padding**. Padding is the process of substituting the pixels on the edges.

Step 1- Feature Map Construction (Padding)



- There are different approaches to performing padding, including **reflection padding**, **replication padding,** and **zero padding**. Otherwise, the convolutional filter does not go outside the image border its padding is referred to as **valid padding**. If the output of the convolution has the same size as the input image, its padding is referred to as **same padding**.

Step 1- Feature Map Construction (Padding)



- Zero padding (masking) is just adding a zero in the padding area.
- Replication padding is using the same value for the last pixel in the padding area and substituting it in the padding area.
- Reflection padding is using the neighbor value of the last pixel on the opposite side of this pixel, and substituting it in the padding area.
- Valid padding happens when the filter window stays inside the image and does not go outside.

CNN Algorithm Step by Step

1. Convolution: Using kernels (filters) to construct feature maps
2. **Non-Linear transformation**
3. Pooling
4. Flattening
5. Fully Connection (Dense Layer)

(Step 2) Perform non-linear transformation

- Filters create convolved images, which are **linear transformation** of the original image.

- However, an image includes a set of non-linear objects. To increase the non-linearity of feature maps (convoluted images), usually, the feature map will be fed into a non-linear activation function, such as ReLU, to get a non-linear transformation of the image.

- Not having a non-linear transformation after the convolution layer results in lower classification accuracy.

- Keep in mind that at the end output layer should be something related to the task that we expect from the neural network. For example, if it is a classification task for more than two classes, then the activation function of the output layer should be Softmax, if it is for two classes, then it should be Sigmoid.

CNN Algorithm Step by Step

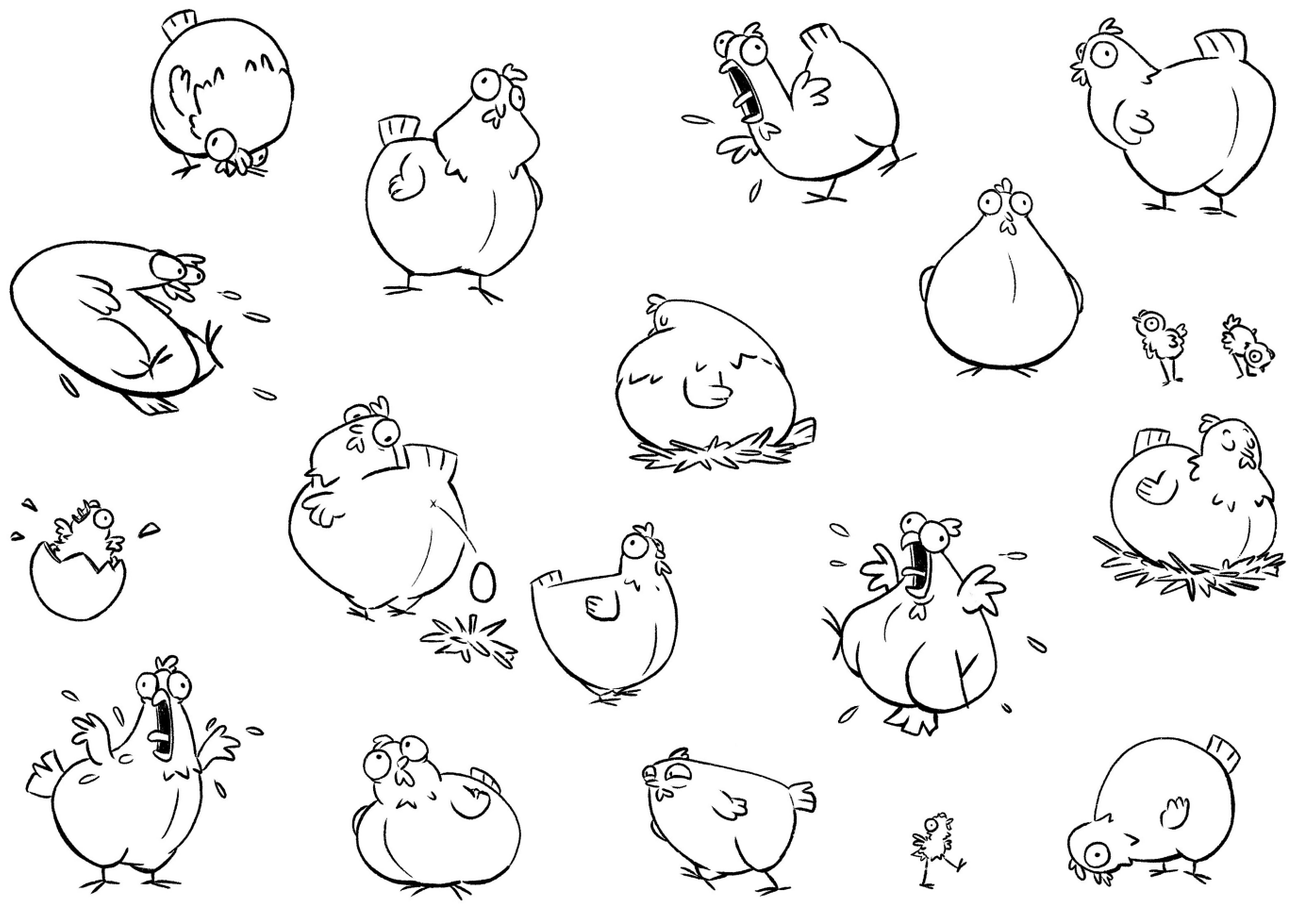
1. Convolution: Using kernels (filters) to construct feature maps
2. Non-Linear transformation
3. **Pooling**
4. Flattening
5. Fully Connection (Dense Layer)

(Step 3) Pooling

- The feature extraction which is done automatically by the CNN is spatial invariant. Spatial invariance is the feature that makes CNN algorithms a very accurate and flexible algorithm for image classification.

- A traditional image feature extraction cannot recognize that these are chickens, because the shape, camera angle, etc. in all chickens are different.

- However, a CNN that has trained on many comic chickens can recognize the given chicken image, despite their different visual shapes



(Step 3) Pooling-Too many filters

- Each convolutional layer applies different filters to the input image for feature extraction. The number of filters is a hyperparameter, and it grows between convolutional layers. For example, the first layer has 32 filters, the next layers 64 filters, and so forth.

- Therefore, a CNN creates many convolved images (feature maps). Dealing with such a large number of data (results of applying many filters) is computationally very ineffective, or impossible.

- To handle this issue, *the CNN downsamples (reduce the size) of convolved images while maintaining the highlighted features*. The pooling functionality is used for this purpose (downsampling), similar to using windows and strides, which downsample the original image into smaller images.

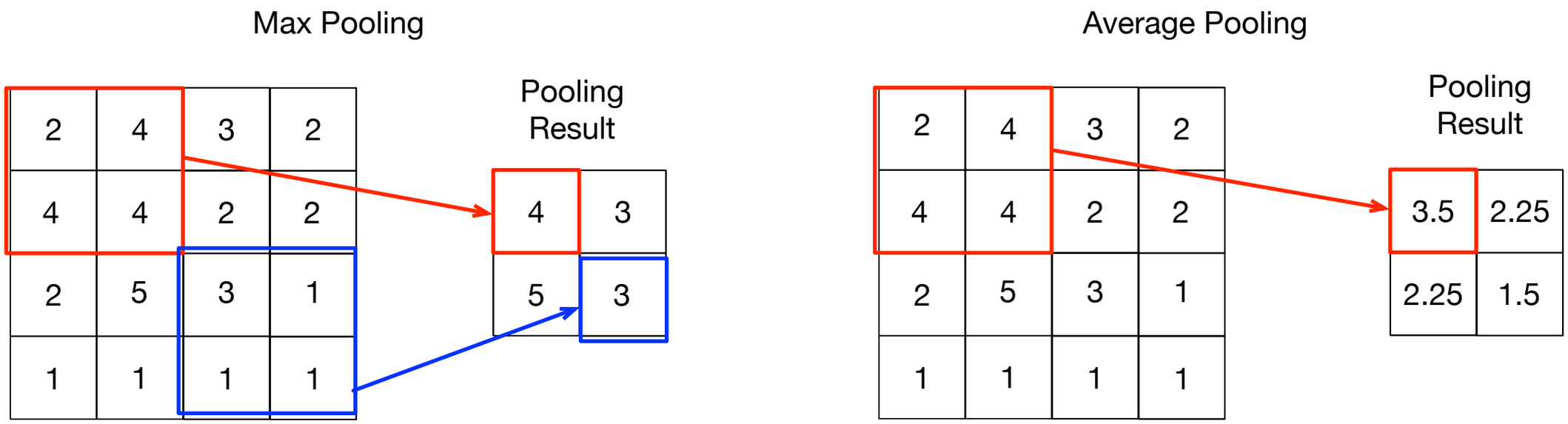
(Step 3) Pooling - Downsampling

- *The CNN downsamples (reduce the size) of convolved images while maintaining the highlighted features*. The pooling functionality is used for this purpose (downsampling), similar to using windows and strides, which downsample the original image into smaller images.

- The result of the convolutional layer, i.e., feature maps (after they have been transformed to non-linear feature maps), are sent to a pooling layer, and the pooling layer downsamples on feature maps by using *tensor operation*.

- Downsampling in CNN refers to lowering the resolution while still maintaining the features.

(Step 3) Pooling-Methods



Max pooling and average pooling examples, which are applied on the convoluted image (feature map) and creates the pooled feature map. In these two examples we use stride of two pixels in each direction.

(Step 3) Pooling-Advantage

- There are other pooling methods as well, such as calculating L_2 norm for all pixels inside a window, but they are not as popular as max pooling.

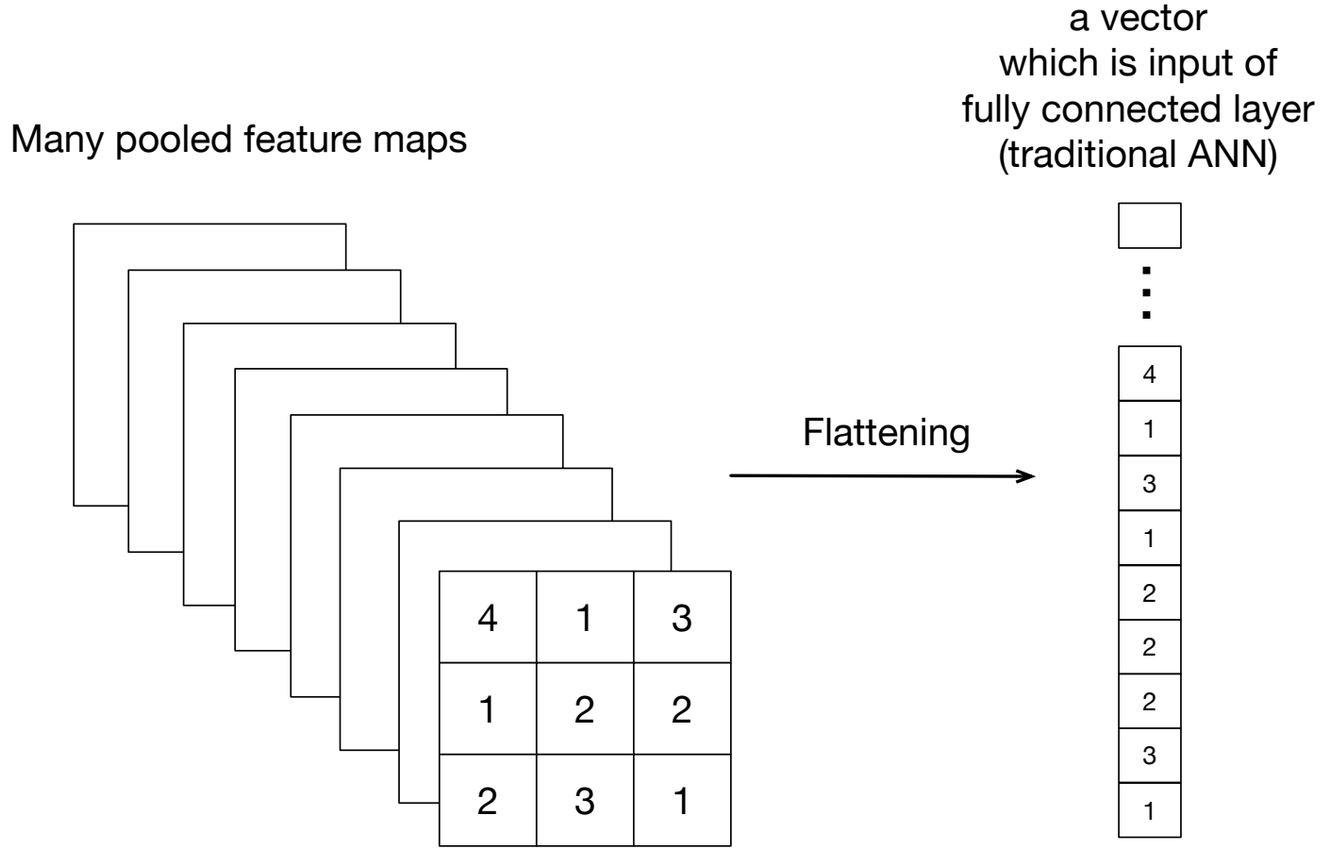
- Pooling has other advantages as well, it removes irrelevant information and unnecessary features. This reduces the chance of overfitting (because of reducing the number of parameters) in addition to making the input smaller and thus making the network more resource efficient.

CNN Algorithm Step by Step

1. Convolution: Using kernels (filters) to construct feature maps
2. MaxPooling: Perform non-Linear transformation
3. Pooling
4. **Flattening**
5. Fully Connection (Dense Layer)

(Step 4) Flattening

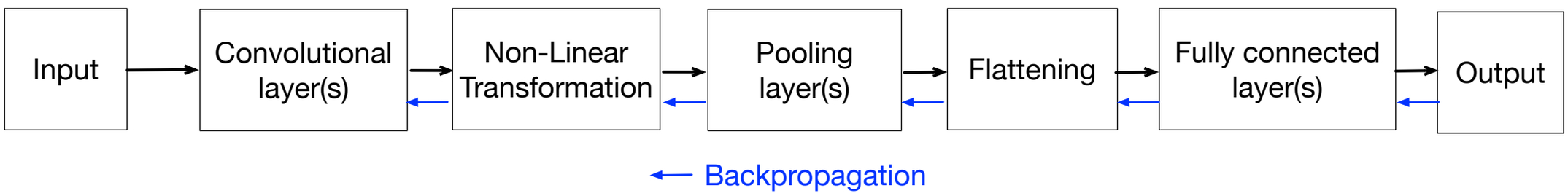
- After the end of the pooling layer usually, there will be a normal flat feedforward layer. Flattening is very simple, the result of the pooling layer is a matrix, and the flattening layer converts matrices to a vector.



(Step 5) Dense Layer

- After this very long vector has been created, then this will be fed into a traditional ANN layer. The input layer of this ANN is the result of flattening, its hidden layers are called **fully connected layer**, and the output, specifies the result of the classification or regression.

CNN Summary



- A CNN architecture can be used for classification and regression both.

- CNN used for classification has a Softmax activation function (or other non-linear activation function) that outputs probabilities for a classification result. The Softmax function is a generalization of the logistic function and provides a vector of values with a range between zero to one. The Softmax function is usually used with cross entropy as a loss function.

- It is not common to use CNN for regression, but if you intend to use a CNN for regression, there will be one output neuron, and its activation function will be linear.

Outline

- Convolutional Neural Network
- Convolution and Cross-correlation
- How to Build CNN
- **Different Types of Convolutions**
- CNN Models for Classification

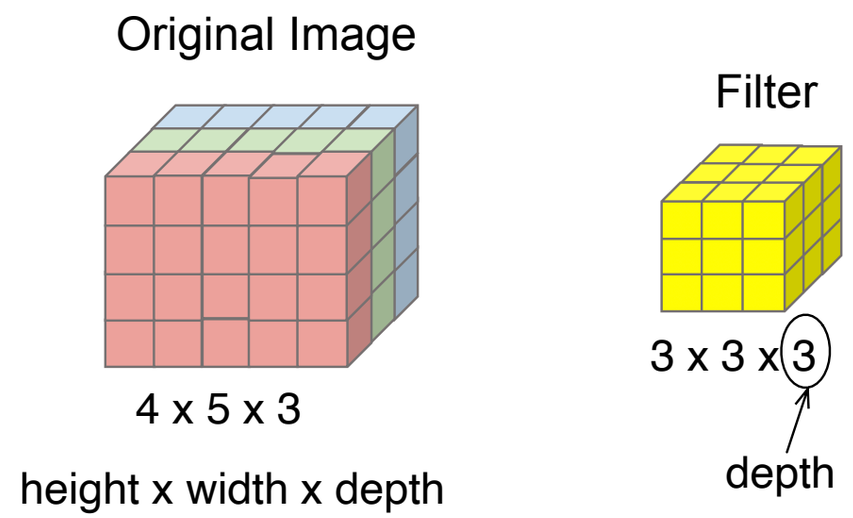
Different Types of Convolutions

- 3D Convolution
- Dilated Convolution
- Transposed Convolution

3D Convolution

- **3D Convolution:** Until now, we only used matrix and explained convolution on matrix. This convolution is called 2D Convolution. If the original data is in 3D format, we should use 3D Convolution.

- For example, a colored image has three color properties (Red, Green, and Blue) along with x and y coordinates. Therefore, we should deal with a 3D tensor (three matrices with the same width and height). The third dimension is referred to as **depth** or **channel,** and the size



3D Convolution

- The dot product of 3D filter to this 3D tensor will be a matrix, which you can calculate on your own, by setting arbitrary stride and padding sizes.

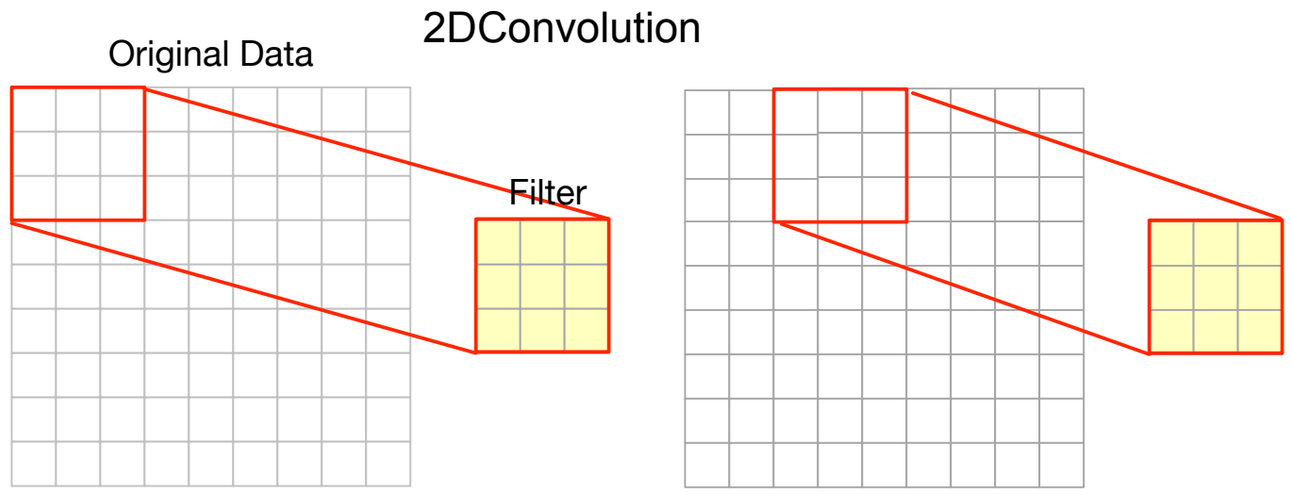
- 3D convolutions are used for common 2D images and also volumetric images (e.g. MRI, CT images), which also include voxel (a pixel with one additional dimension that is depth) or video files (the third dimension is time).

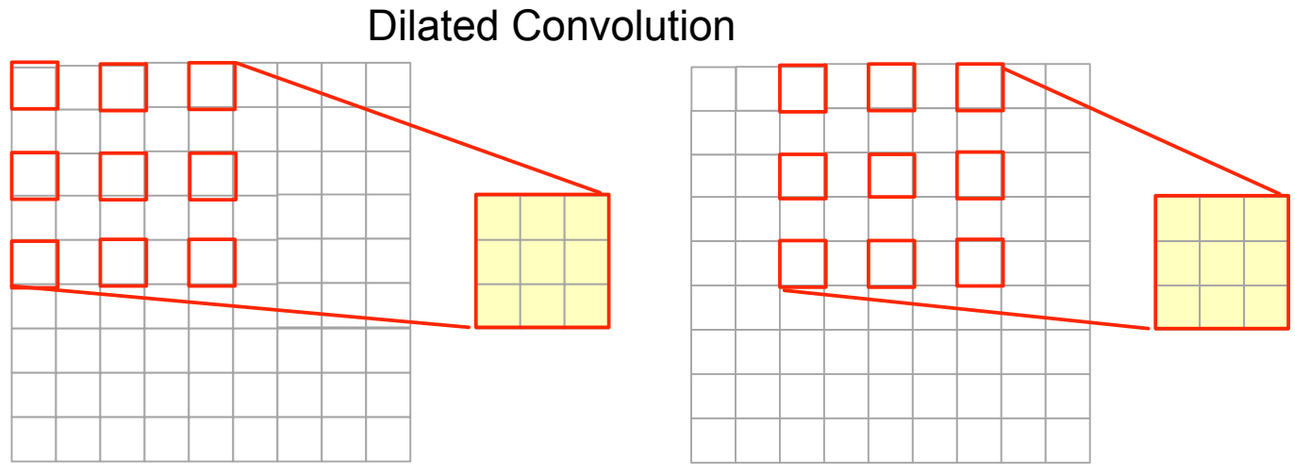
Dilated (Atrous) Convolution

- Dilated Convolution adds a gap between pixels that feds into the filter.
- A parameter called *dilation rate (l)* specifies the size of gaps between matrix elements (e.g. pixels). In other words, parameter l indicates *how much the kernel is widened*.

Dilated (Atrous) Convolution

- The top figure has a dilation rate of $l = 2$, 2D Convolution shown on top of this figure has $l = 1$ (no gap between pixels). As the dilation rate increases, the patch sizes increase as well because it skips several pixels.



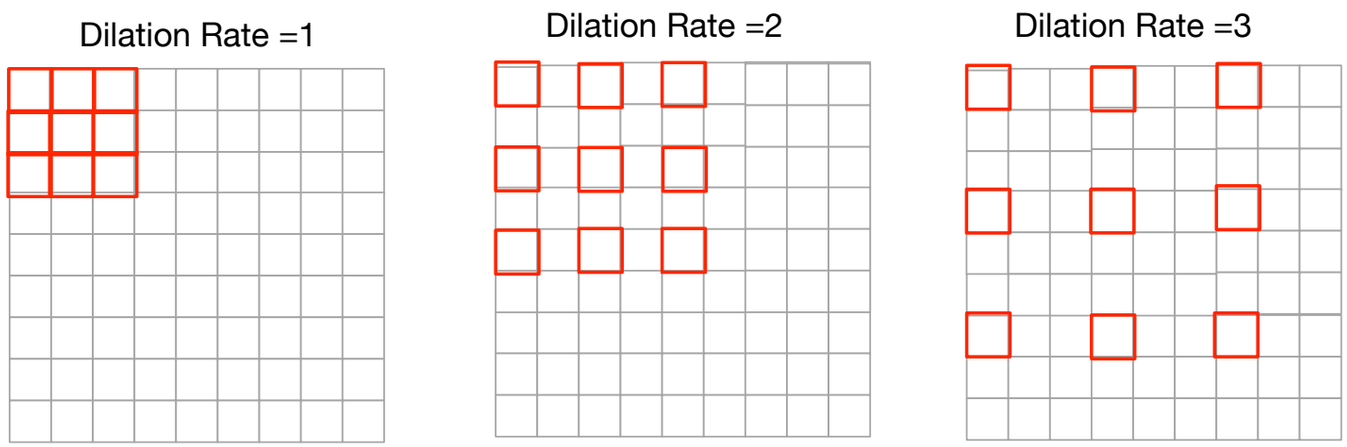


- Dilation convolutions are popular approaches for image segmentation [Chen ’17].

Dilated (Atrous) Convolution

- Second line figures has a dilation rate of $l = 2$, 2D Convolution shown on top of this figure has $l = 1$ (no gap between pixels).

- As the dilation rate increases the patch sizes increase as well, because it skips several pixels.



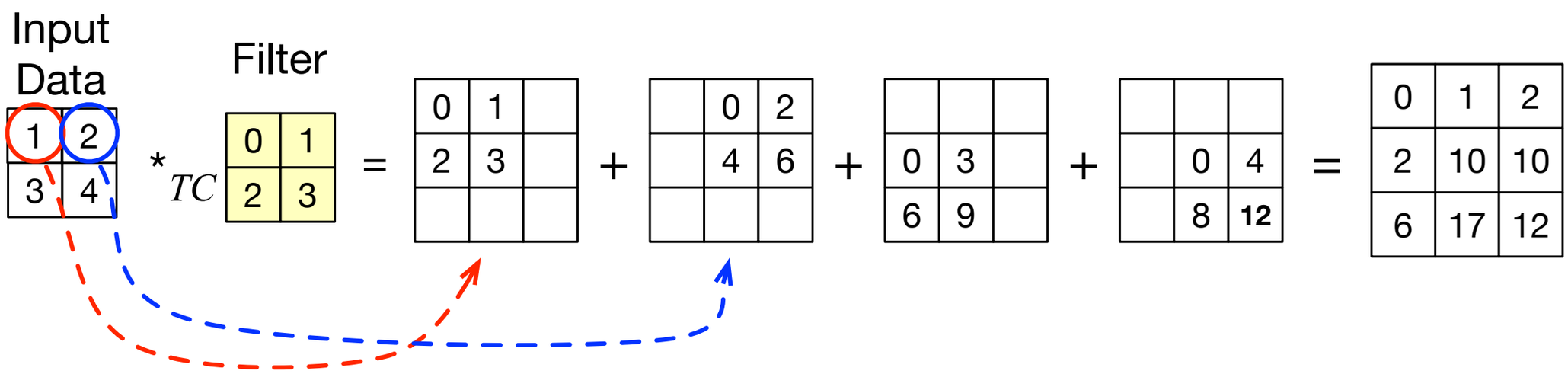
Transposed Convolution

- Convolutions we explained (except dilated convolution) either keep the size of the original data or reduce its size after convolution, which is called “downsampling”.

- The transpose convolution is usually used to increase the size of the output feature. This technique is referred to as an upsampling process, and it has applications in image processing, such as increasing the resolution of an image, reducing blurriness in an image, etc.

Transposed Convolution

- A simple example of transposed convolution, with stride size 1 and padding size 0.



- The inverse of convolution is referred to as deconvolution and it is recommended to not call transposed convolution deconvolution.

CNN Example with Pytorch

https://colab.research.google.com/drive/17o6CvTvGwGlstTl24cD4lwF0tYZDwurf?usp=sharing

Some Notes about CNN (1/2)

- The CNN will learn the kernel (filter) itself, and we do not need to give the kernel manually. In particular, the algorithm will decide which kernel will get meaningful information from an image.

- Edge detection kernels (Sobel filters) are very common to be used by CNN algorithms.

- Often in real-world cases, we stack several convolutional layers on top of each other. This means that the output of one convolutional layer will be an input of another convolutional layer and so forth. This causes the CNN to discover more complex patterns as it goes deeper into convolutional layers. In simple words, for example, a CNN with five convolutional layers could recognize more patterns than a CNN with three convolutional layers.

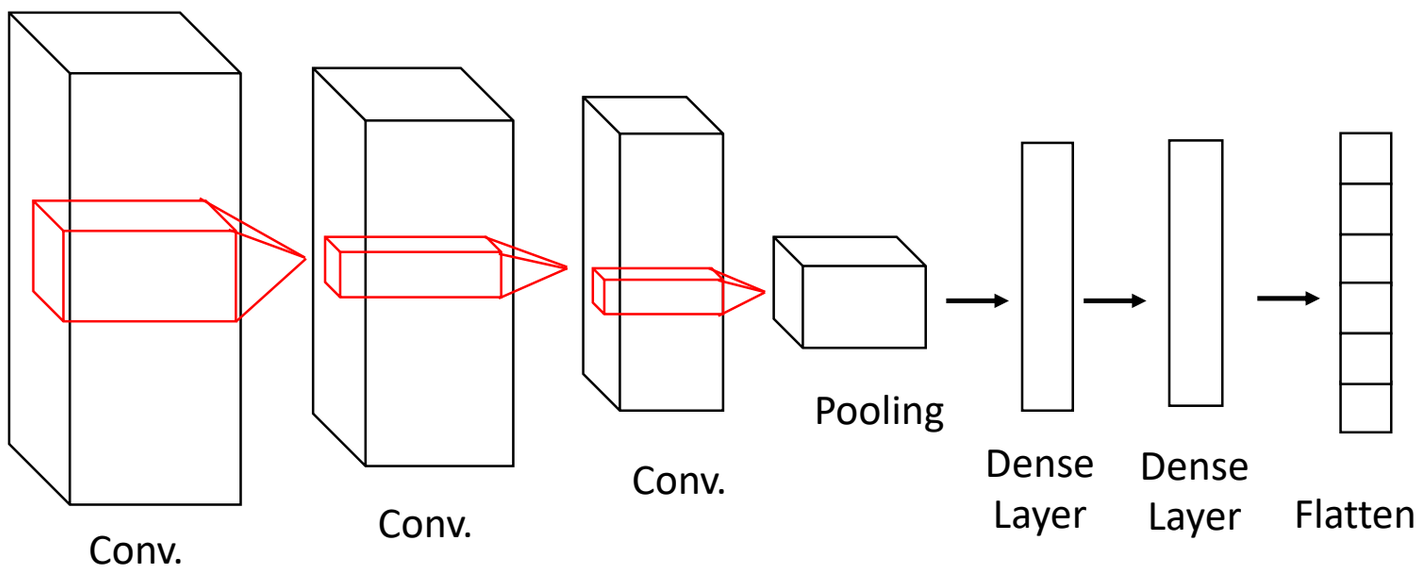
- We use cross entropy cost for classification tasks, because classifications are use softmax function. If the task is regression we can use means square error.

Some Notes about CNN (2/2)

- **Nowadays, we rarely need to build a CNN on our own and most of the time we are using an existing model to define the model that fulfill our demand, i.e., transfer learning.**

* If we intend to classify images with different sizes, the pooling layer can handle this by constructing a fixed-size feature map and thus making it easier for the classification.

* AlexNet visualization [Krizhevsky ’12] is shown in the Figure below. Here, both convolutional and pooling layers are presented as cubes, the red cubes inside convolutional layers are presenting filters.



Outline

- Convolutional Neural Network
- Convolution and Cross-correlation
- How to Build CNN
- Different Types of Convolutions
- **CNN Models for Classification**

Popular CNN models

- The earliest CNN architecture was introduced by Fukushima in 1980 [Fukushima ’80]. Next, one of the earliest use of CNN for image classification was introduced by LeCun and Boser in 1989, and it is called LeNet [LeCun ’89].

- Next to LeNet, there is a large gap in the neural network until the introduction of DanNet [Ciresan ’11] and AlexNet [Krizhevsky ’12].

- AlexNet, VGG, InceptionNet, ResNet, Inception-ResNet

AlexNet

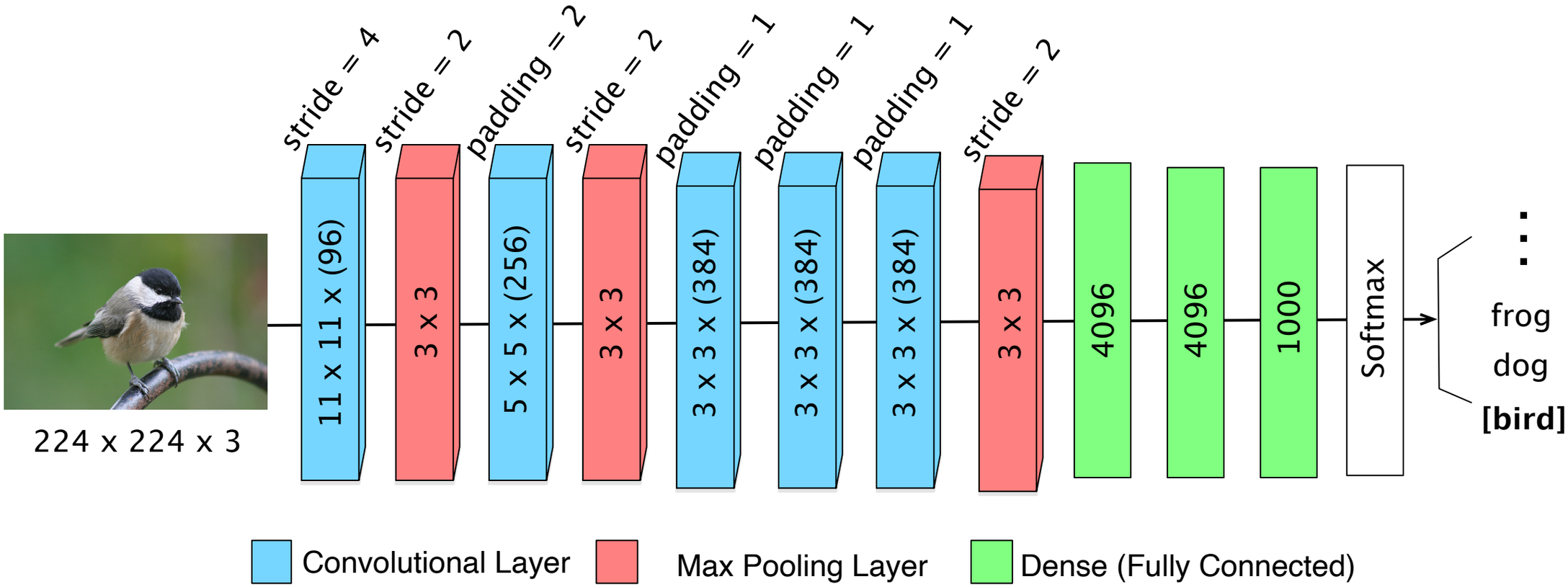
- **AlexNet:** After LeNet was introduced, due to its hardware limitations and advances of other algorithms such as SVM, neural networks did not receive much attention until the success of AlexNet and DanNet in 2012.

- We have described that AlexNet is one of the main fuels of the deep learning revolution and it won the ImageNet.

- Before AlexNet, the best ImagNet error rate was 26.2%, AlexNet achieved an error rate of 15.3%, which was 10.9% improvement.

AlexNet

- AlexNet architecture is similar to LeNet, but it has eight layers and it has ten times more convolution channels than LeNet.



- It also uses ReLU as an activation function instead of Sigmoid or Gaussian activation function, and max pooling instead of average pooling.

VGG

- Two years after AlexNet in 2014 VGG [Simonyan ’14] has been introduced and won the ImageNet’s ILSVRC award. The VGG-16 net achieved an error rate of 7.3%.

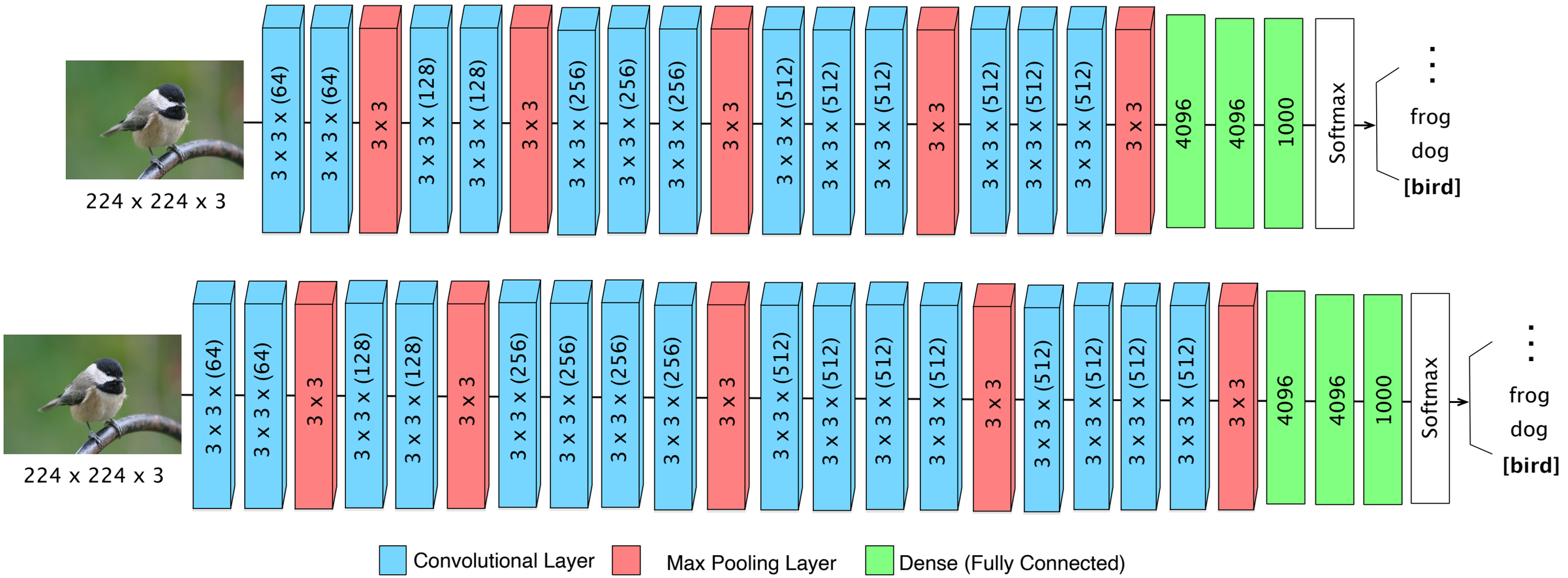
- VGG or VGGNet main difference from AlexNet is its deeper network architecture, i.e. 19 layers.

- Similar to AlexNet this model is also trained on ImageNet dataset. Therefore, its input image size is also the same ($224 \times 224 \times 3$), also same as AlexNet and LeNet it has two parts, its first parts are a set of convolutional layers.

- The second parts are a set of dense layers. Its second parts are exactly similar to AlexNet, with three dense layers (4096, 4096, and 1000 output). A set of convolutional layers that ends with a max pooling layer is referred to as a VGG block by authors.

VGG

- VGG16 on tope and VGG19 in the bottom.



VGG

- The filters in VGG are very small, 3 × 3 convolutions, with a stride size of 1 and padding size of 1.

- Its max pooling layers are 2 × 2 with a stride size of 2.

- Interestingly, the authors of VGG find that 3 × 3 convolution has the same “effective receptive field” as 7 × 7 convolution or larger convolutions.

Effective Receptive Field

- Huebel and Wiesel [Hubel ’59] performed a series of experiments on cats and realized that many neurons inside the visual part of the brain react only to a limited region of visual data, which is known as the *local receptive field*.

- On the other hand, some neurons react only to a larger region of visual data. These findings lead to the idea that the visual perception of mammals identifies complex visual patterns through higher-level neurons, which operate based on the output of lower-level neurons.

- The “Effective Receptive Field” refers to ***the region in the input image that produces an important feature.*** In other words, it is the output unit that contains pixels with a non-negligible impact (including visual stimuli) on that unit.

- If you still did not get it, In a biological context, **the receptive field is the portion of the sensory input space (e.g. image) that causes a neural response.**

- In computer vision, the receptive field refers to the region in the input data that produces *the feature*. Or the *smallest region that includes a feature of the image.* Its size is equal to the filter (kernel) size.

GoogLeNet / Inception Net

- GoogLeNet (Inception Net) [Szegedy ’15] is another successful architecture that has been introduced in 2014 and performs close to VGG.

- GoogLeNet achieved a error rate of 6.67% on the ImageNet Large Scale Visual Recognition Challenge in 2014.

- Authors of GoogLeNet investigate to find the best size of convolution kernels and they find it is better to have a combination of varied size convolutional kernels.

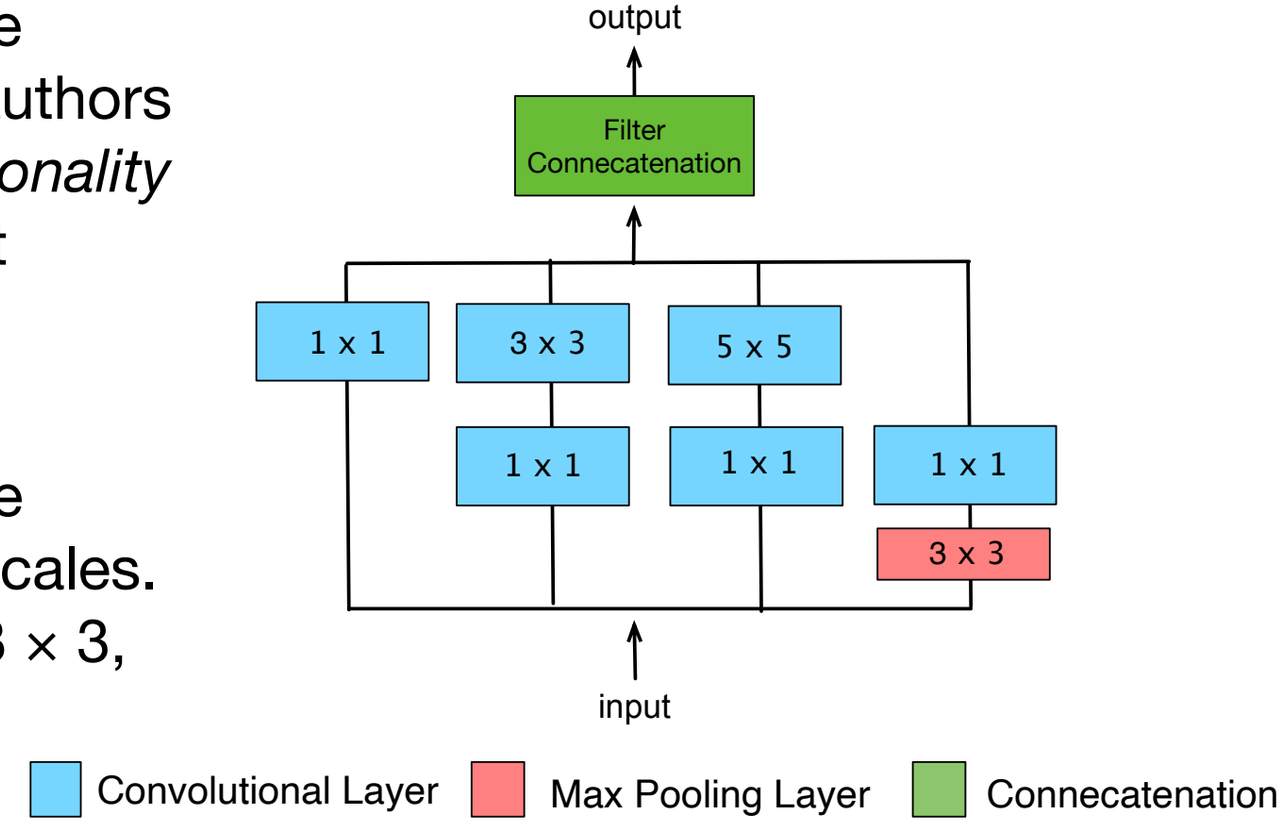
- Inspired by a quote in the movie Inception, “we need to go deeper”, the authors introduced the *inception module*. An inception module is a local network, and GoogLeNet operates by stacking inception modules on top of each other, which is a *network inside a network*.

Inception Module

- Each inception module takes input and sends it to four parallel paths of convolution to get into output. All convolutional layer has a ReLU activation function.
- At the first layer, the input will be fed into three 1 × 1 convolutional layers with a stride size of 1.

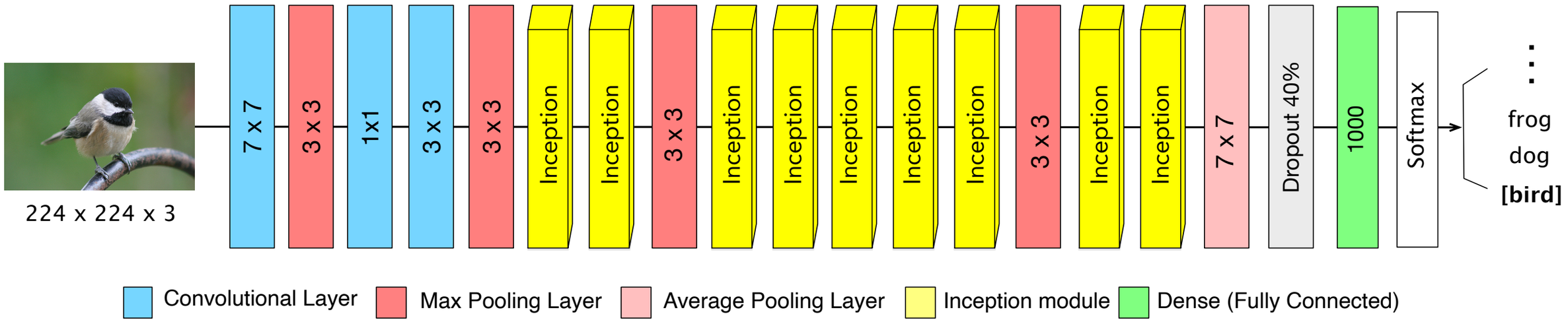
- These convolutions are too small to capture any specific pattern in the image, but they are outputting fewer feature maps, and the authors said they are used to *reduce the dimensionality* of the input data while keeping important features of the data.

- The second convolution layer enables the network to capture patterns in different scales. This layer has three convolutions 1 × 1, 3 × 3, and 5 × 5. In the end, all output will be concatenated together in the “filter concatenation” layer.



GoogLeNet Architecture

- Stacking inception modules on top of each other, and not having a fully connected layer reduces the number of parameters significantly and as result, it has about 6 million parameters.



InceptionNet versions

- After the first version of GoogLeNet (Inception v1), more versions of Inception have been introduced.

- Inception v2 leverages the use of Batch norm, uses ReLU as an activation function, and uses two 3×3 convolutions instead of a 5×5 convolution in the Inception module.

- Inception v3 introduces a "factorization", which reduces the number of parameters and also reduces the risk of overfitting. For example, a 5×5 convolutional layer can be substituted by 1×5 and 5×1 convolutional layers, which have fewer parameters.

- Before learning Inception v4, we should learn ResNet

Residual Networks (ResNet)

- One year after GoogLeNet won the ILSVRC award, in 2015 the Microsoft team propose Residual Network (ResNet) [He ’16] and won the 2015 ImageNet ILSVRC challenge.

- ResNet-50 achieved a top-5 error rate of 3.57% and ResNet-101 achieved a top-5 error rate of 3.25% in the ImageNet ILSVRC challenge.

- Increasing the number of convolutional layers does not increase the accuracy, and in contrast, causes ‘**degradation**’ problem, i.e., *as the network depth increases the accuracy gets saturated and then starts to decrease rapidly*.

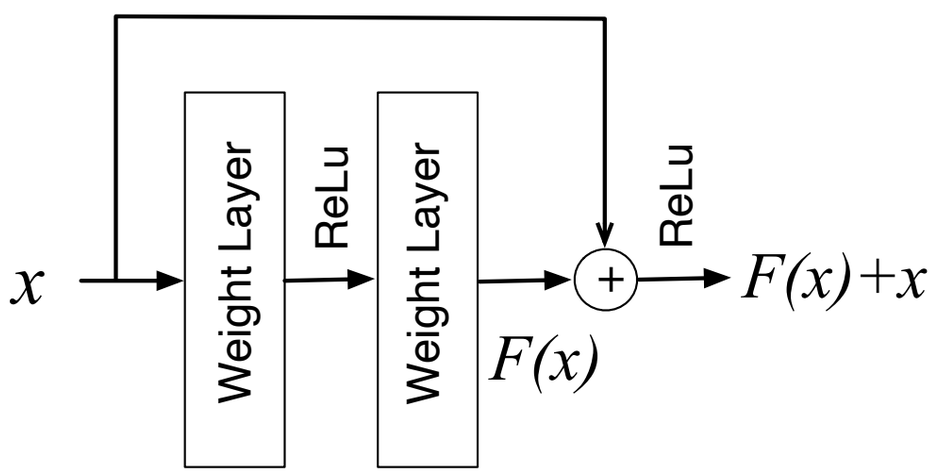
- Authors of ResNet claimed that optimization is the origin of the problem, because deeper models are harder to get optimized than shallower models, and thus their accuracy decreases. A major reason is a vanishing gradient caused by backpropagation in deep networks.

ResNet - Skip Connection

- ResNet resolves this issue by using the **Skip connections**.

- A skip connection adds the original input into the output. Therefore, the output will be the combination of the original input and the previous layer's output.

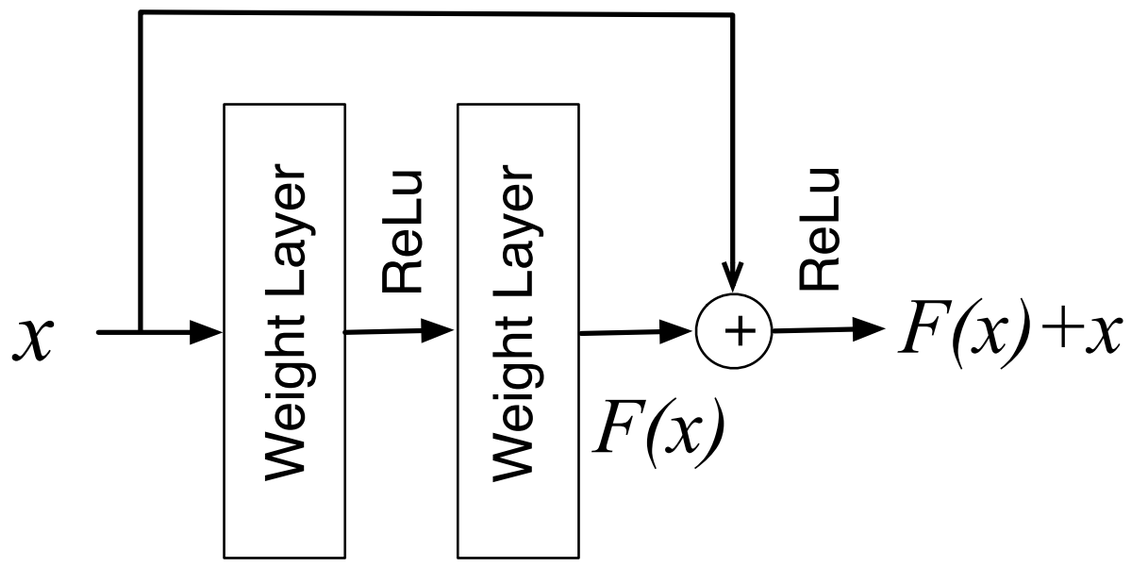
- ResNet calls a set of weight layers with skip connection 'Residual blocks', and by stacking several residual blocks it builds a deep neural network, which ensures high accuracy.



ResNet - Skip Connection

- Input x will pass weight layers and construct the $F(x)$, however, the next layer after the Residual block does not receive $F(x)$, instead, it receives the result of $F(x) + x$, that fed into a ReLU activation function.

- In summary, Skip connections (a.k.a. short-cut connections or residual connections), enable the network to go deep without getting into any optimization related challenges.

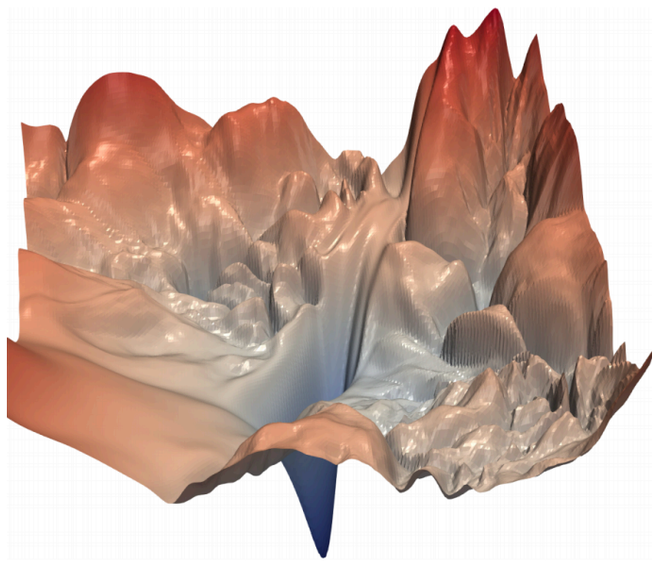


Skip Connection

- The following figure presents a visualization of a loss [Li ’17] in ResNet with and without skip connections. Here you can see how using the Skip connection flattens the complex curves and thus makes it easier for the gradient descent algorithm to reach the minima.

- We stated vanishing gradient problem resolved by Skip connection.

without skip connection



with skip connection

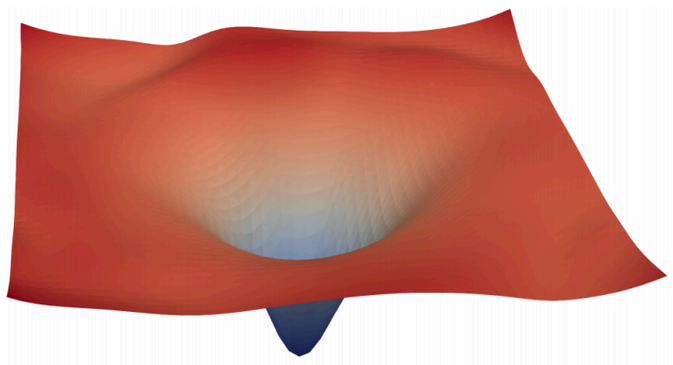
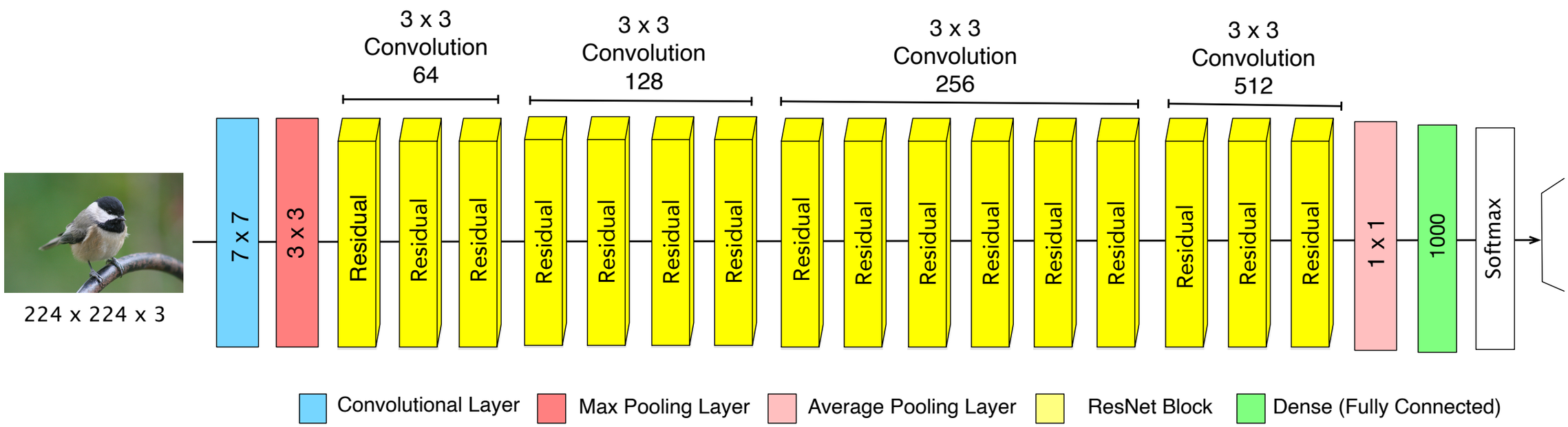


Image credit: Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018). Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, *31*.

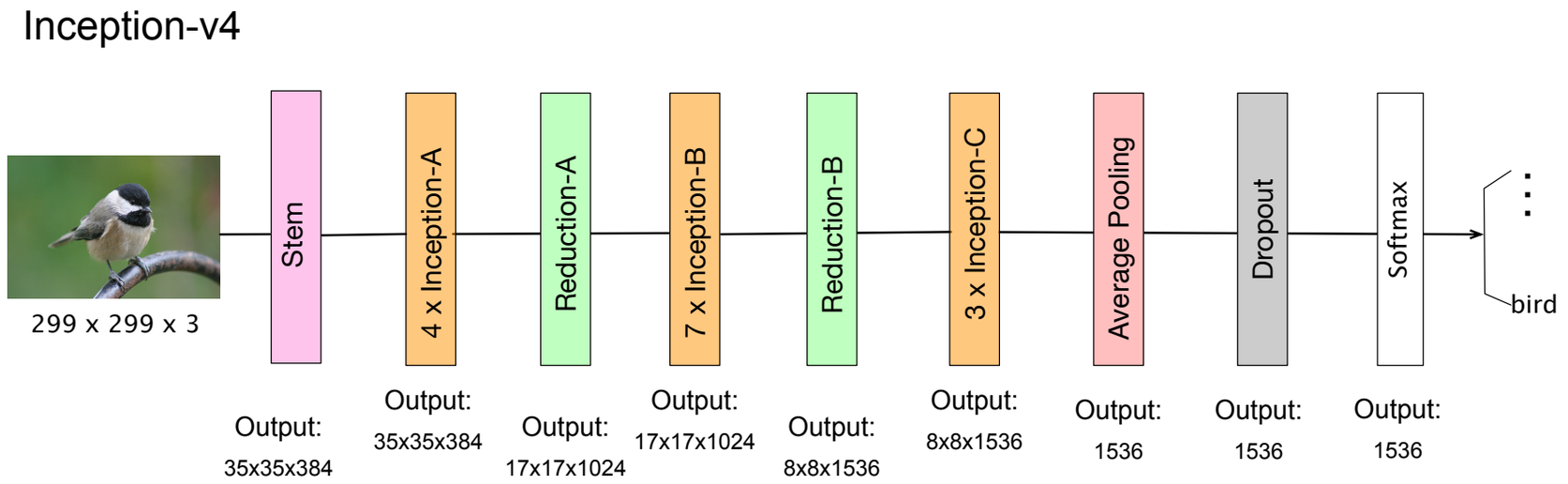
ResNet Architecture

ResNet uses He [He '15] initialization, its batch sizes are 256. After each layer of convolution, it uses a Batch normalization (we did not show it as a separate layer in the Figure). It has 3×3 convolutional layers (similar to VGG) and each convolutional layer is followed by a ReLU activation function.

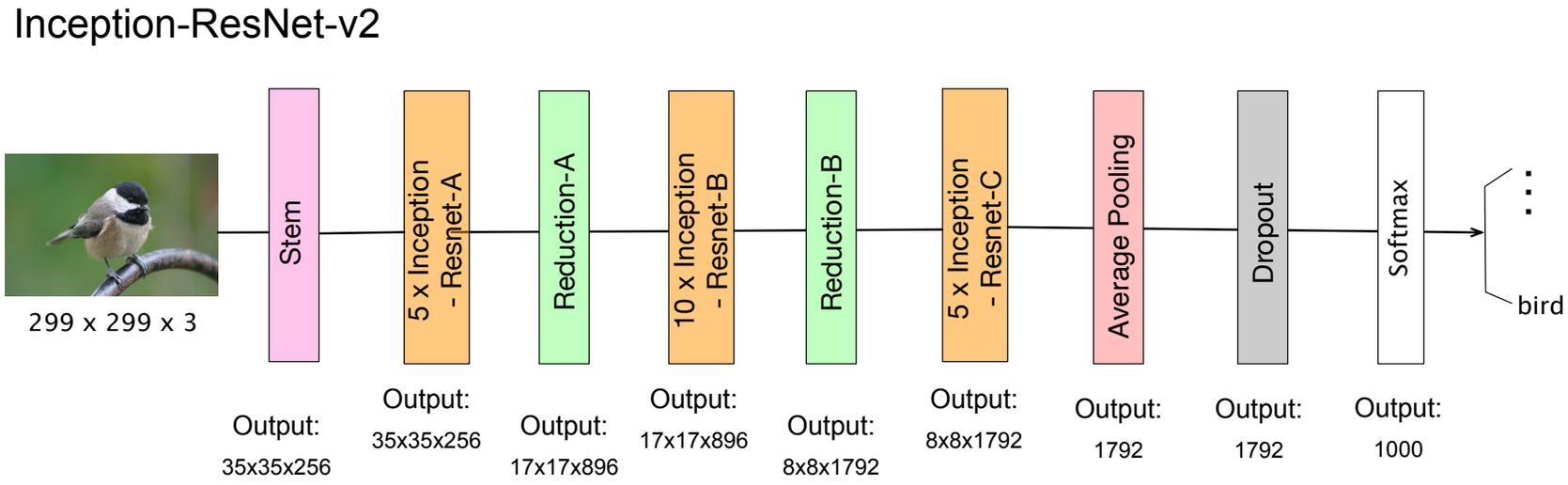


Inception-v4 and Inception-ResNet

- The inception model has been advanced until Inception-v4 and Inception-ResNet [Szegedy ’17] were released about one year after ResNet.



- Inception-ResNet combines the architecture of Inception with the Residual network. Authors report different combinations of inception and ResNet (Inception-ResNet-v2) and also Inception improvements alone, i.e., Inception-v4.

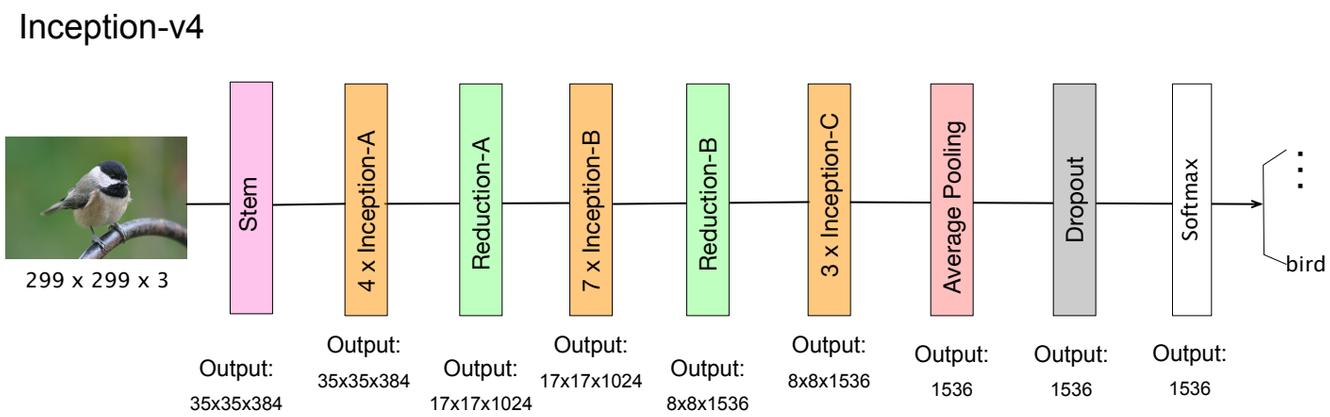


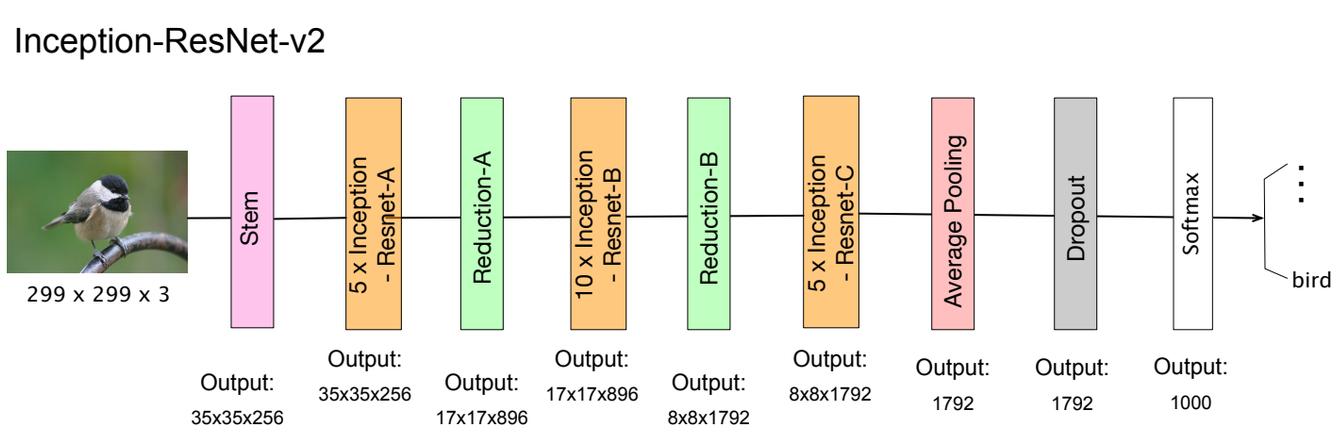
Inception-v4 and Inception-ResNet

- We don’t go into the detail of each layer and you don’t need to learn in such detail.

- However, there is one layer introduced in Inception models, as “Stem” and it is worth further exploring.

- Stem layer provides *parallel convolutions* that were concatenated at the end, this enables the model to have a different *Field of View (FoV)* on an image, which results in higher classification accuracy.





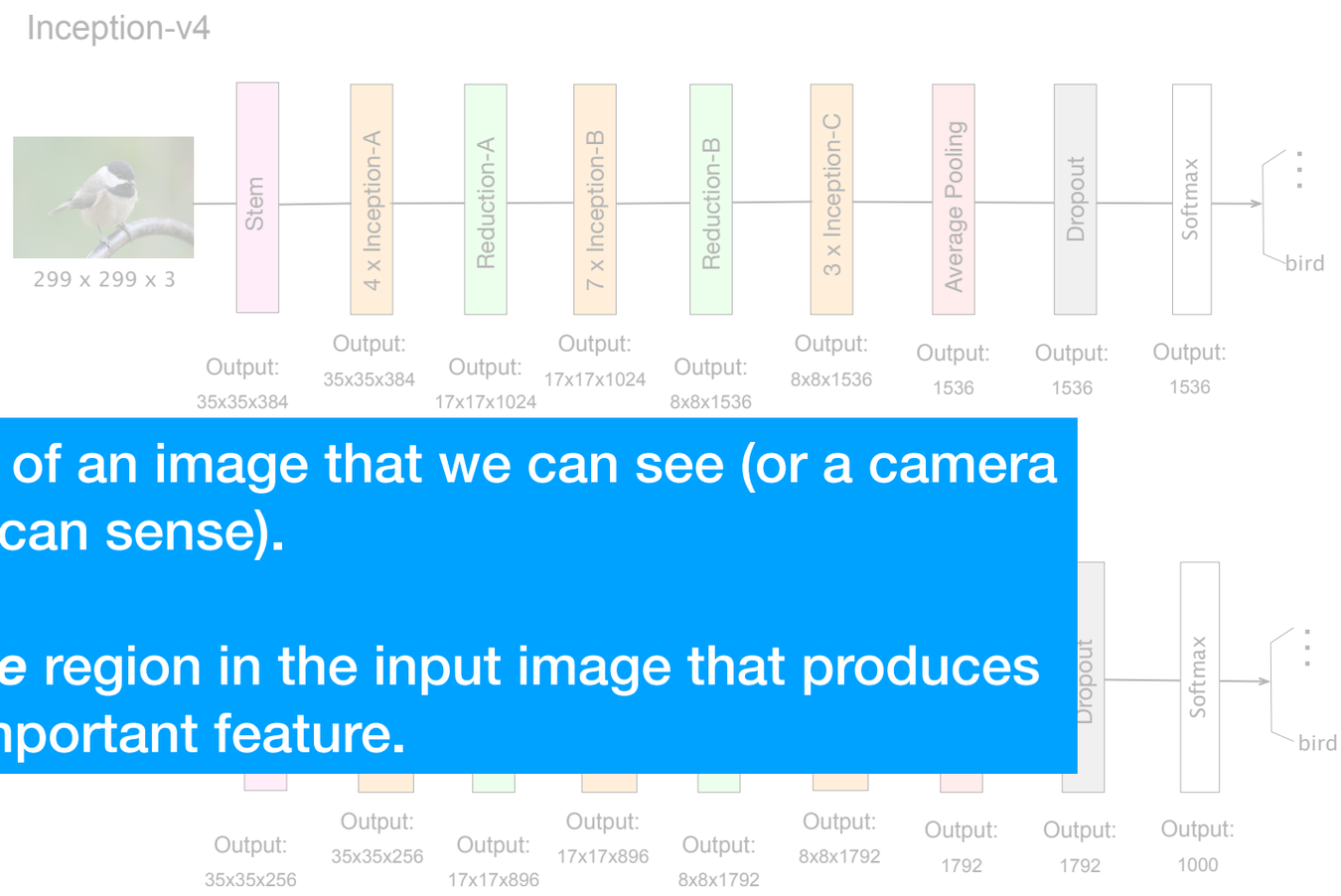
Reduction modules t reduces the spatial dimensions of the feature maps while increasing the number of channels. They reduce the network complexity, while increasing the depth of the network

Inception-v4 and Inception-ResNet

- We don’t go into the detail of each layer and you don’t need to learn in such detail.

- However, there is one layer introduced in Inception models as “Ste… further e…

- Stem layer provides *parallel convolutions* that were concatenated at the end, this enables the model to have a different *Field of View (FoV)* on an image, which results in higher classification accuracy.



Field of View refers to the area of an image that we can see (or a camera can sense).

Effective Receptive Field is *the* region in the input image that produces an important feature.

Reduction modules t reduces the spatial dimensions of the feature maps while increasing the number of channels. They reduce the network complexity, while increasing the depth of the network

Summary

- Convolutional Neural Network
  - Convolution and Cross-correlation
  - How to Build CNN: Conv. Filters, Non-Linear Transformation, Pooling, Flattening, Fully Connected Layers(FCN)
  - Different Types of Convolutions (2D Conv, 3D Conv, Dilated Convolution, Transposed Convolution)
  - CNN Models for Classification (AlexNet, VGG, GoogLenet/ Inception, ResNet)