Convolution Neural Network – Better Understanding

Premanand S
Last Updated : 20 Nov, 2023

15 min read 5

This article was published as a part of the Data Science Blogathon

Introduction:

In the world of Deep Learning (DL), there are many trending and advanced models are emerging, for making our life (researchers!) easy. One among them is Convolution Neural Network (CNN), in this article we are going to see in detail this model.

Table of contents

Deep Learning -The Evolution:

In today's scenario, many people irrespective of their domain (degree qualification!) opt for the Data Science domain. There may be much reason but the ground reality is most of the IT giants and big organization, they are shifting towards DATA SCIENCE side with a small part (means along with the current speciality, they also include data science) or as a full time. One of the main reasons is smartness and also its robustness irrespective of any domain. If you look into the history behind Deep Learning it is not the technology that happened before some five or ten years old. In 1943, Walter Pitts and Warren McCulloch created a computer model based on the human brain's neural networks. They used algorithms and mathematics, called "threshold logic" to mimic the thought process. But the popularity gets faded in the middle years, you know because of two simple reasons,

1. Availability of data (more and more data!)

2. Processor power (to run the loads and loads of data!)

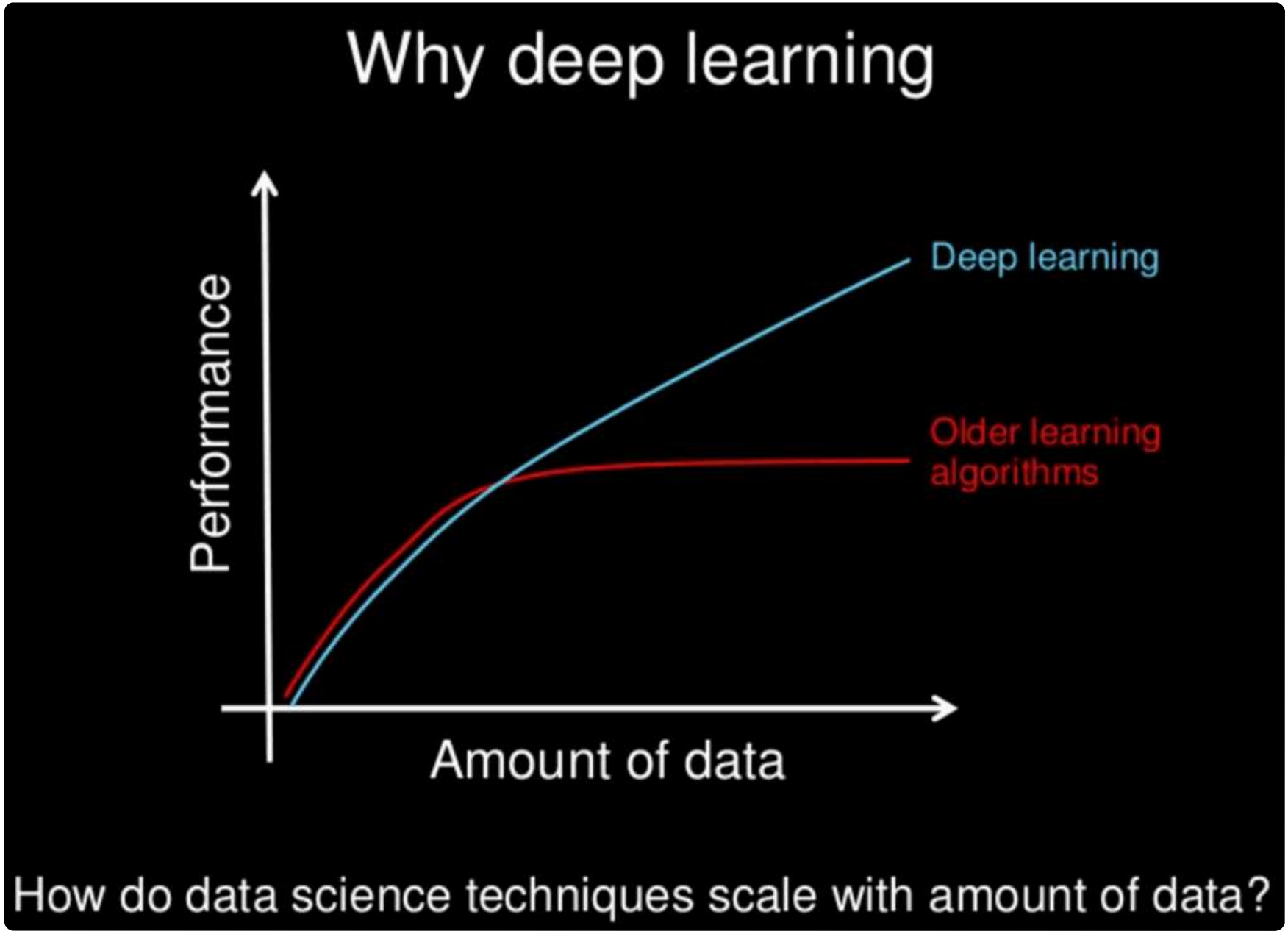


Image Source: https://machinelearningmastery.com/

Geoffrey Hinton – Father of Deep Learning

In simple terms, Deep Learning is used to mimic our very own Human Brain (On Processing! but not fully!) and to replicate, how well our human brain learns things faster and thinks with sense.

Still need to read more about the DL approach, (https://www.tango-learning.com/post/deep-learning-an-intuition-behind-the-technology)

Types of Deep Learning:

There are many different types of DL as per different sources prevailing in our internet ocean, but predominately, DL is classified as,

1. Artificial Neural Network (ANN)

2. Convolution Neural Network (CNN)

3. Recurrent Neural Network (RNN)

Artificial Neural Network:

We discussed already Artificial Neural Network (ANN) in detail in the following article with python code. Please find the link for better understanding, (https://www.analyticsvidhya.com/blog/2021/06/artificial-neural-networks-better-understanding/)

Convolution Neural Network – Understanding:

Before getting into the main topic, Convolution Neural Network (CNN). First, let us understand some basic clarity, so look into the below picture,

Image Source: Google Image

What we inferred from the above diagram? Some say it as Men and some say it as Women and some say it as both!.. Because of the features that get picked by our brain while viewing the drawing. If you would easily differentiate the above picture, then view the next one, the below-mentioned picture,

Image Source: Google Image

It's actually a fusion of the male and female picture while viewing some thought as male and female. So above two examples illustrate to us how the brain works, that it processes certain features on these examples or whatever we see in real-life and it classifies that as such. So we are going to do the same process with Neural Network with CNN is almost very similar. It's like how we are getting features from the above pictures, in the same way, the computers also going to process, sounds interesting right!

Yann Lecun – Grandfather of Convolution Neural Network!

Convolution Neural Network – Layman Understanding:

As human psychology, when we sit or lie on a bed or in any position (when we are awake – condition applied!), we tend to observe and look at our surroundings with or without knowledge, our brain will be in an active state only and it keeps on processing and sometimes it keeps predicting too right!.

Image Source: Author

Just look at the above picture, what we identified from the photo is some persons and some projects are present. After that what we analyzed are plates with some food not in particular and tumbler with water, then ID card, etc…Added in the emotional side if we observe means it's a happy moment with friends and randomly enjoying some poses. Whatever mentioned above, are predictions by seeing the pictures, so how is possible to predict? Then how you can find the objects and emotion predictions? Our eyes and our brain work in perfect harmony to create such beautiful visual experiences. The system which makes this possible for us is the eye, our visual pathway, and the visual cortex inside our brain.

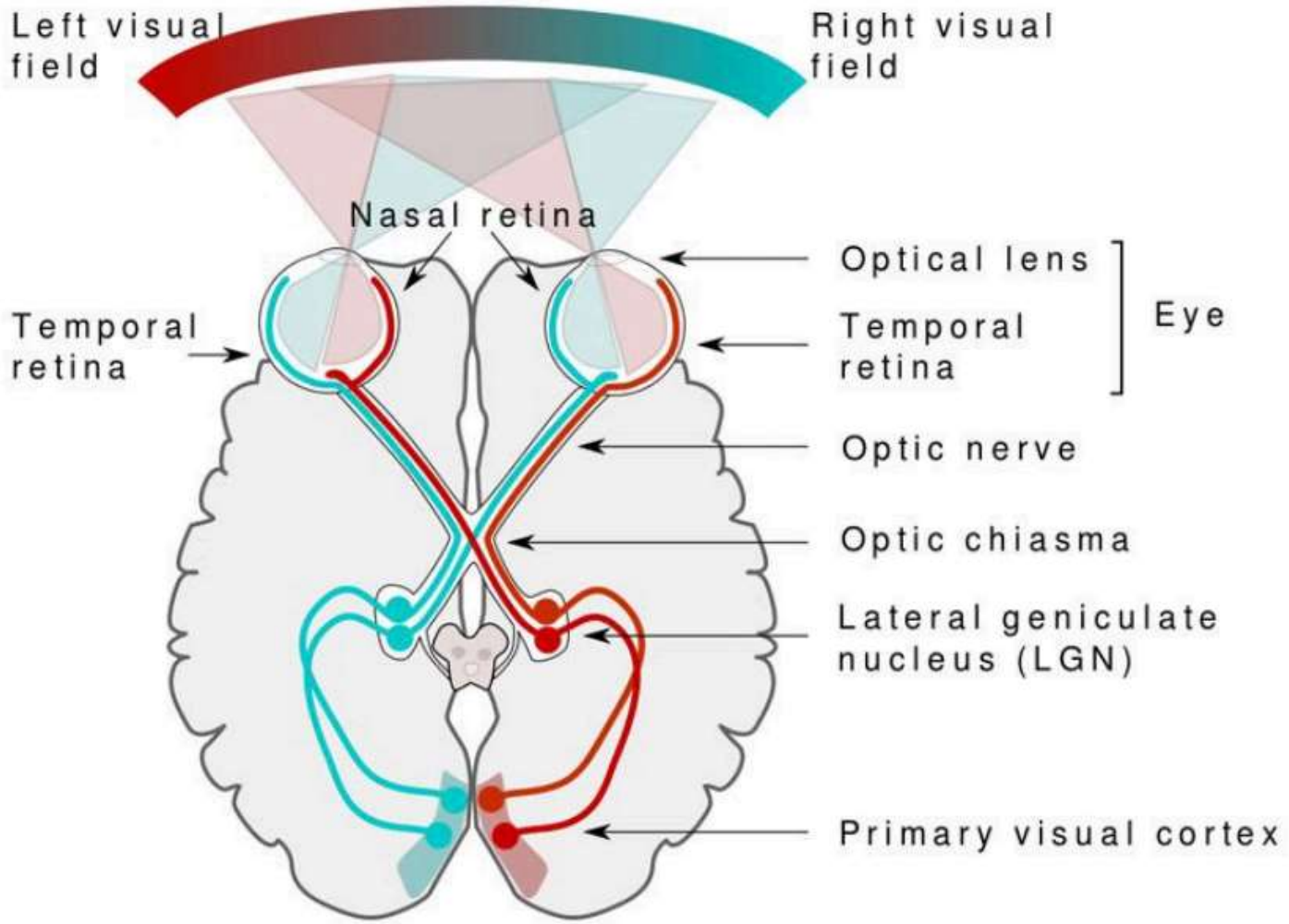


Image Source: http://mriquestions.com/visual.html

Lights and colours on the retina are captured in our eyes. These signals cross the receptors of the retina to the optical nerve and are transferred to the brain for this information to be meaningful. The eye and the visual brain are extremely complex and hierarchical. In perceiving and understanding what we around us view, the full visual path plays a vital part. It is this mechanism inside of us that allows

us to understand the image above, the text of this essay, and every other work we do daily. So these processes we are doing from our childhood and so we can predict and analyze through eyes and brain coordination.

So the same process, is that possible for computers? and the simple answer is YES!. computers can SEE the world but in a different way – through numbers. Like the picture below,

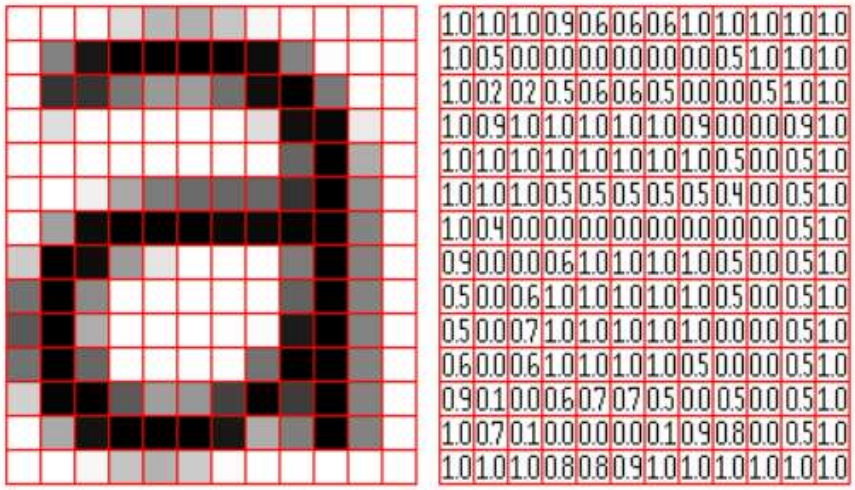


Image Source: http://pippin.gimp.org/image_processing/images/sample_grid_a_square.png

The importance of layers of neurons arranged in each algorithm is, each layer concentrates each pattern and its importance like firstly it observes simple patterns such as lines, curves, then some complex patterns like texts, objects, and face and more complex patterns present in the image or whatever inputs that we are analyzing. This process is almost like a human's point of view when we are randomly visualizing or seeing anything our brain analyzes any input like this only but in a matter of very few seconds.

So there are various steps to achieve this CNN algorithm, they are as follows,

1. Convolution layer

2. Activation function (ReLu layer)

3. Pooling

4. Flattening

5. Full connection

On the whole, the below animation will explain how CNN works,

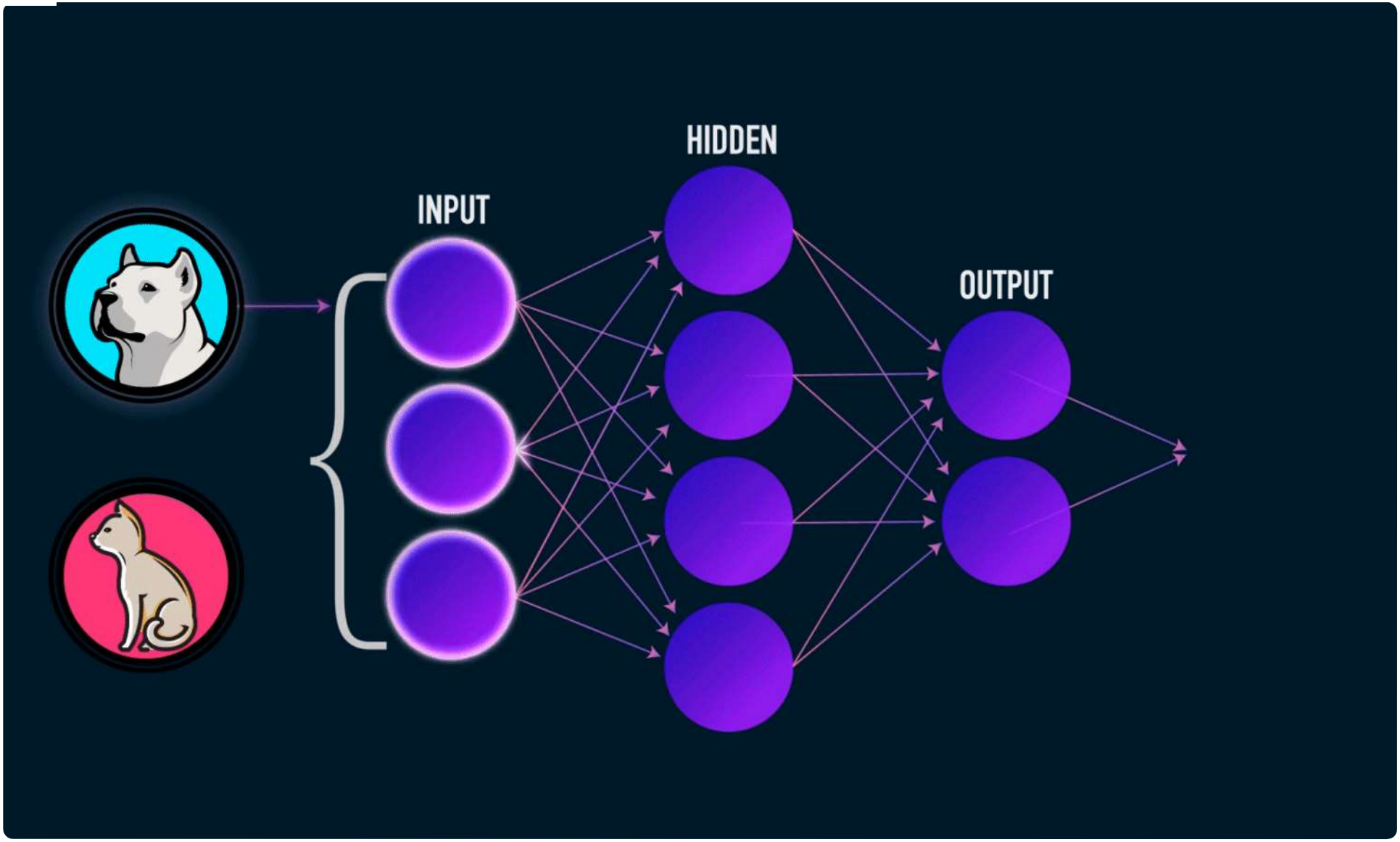


Image Source: https://medium.com/analytics-vidhya/convolutional-neural-network-cnn-and-its-application-all-u-need-to-know-f29c1d51b3e5

another interesting visualization,

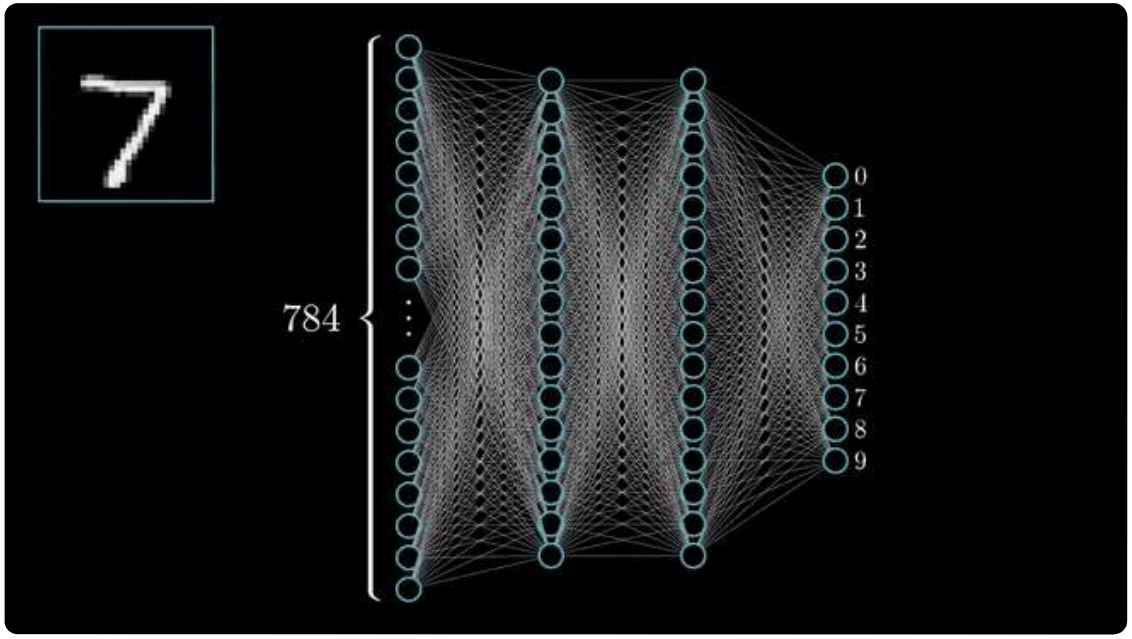


Image Source: https://medium.com/analytics-vidhya/convolutional-neural-network-cnn-and-its-application-all-u-need-to-know-f29c1d51b3e5

Let's see one by one what the steps explain or conveys,

1. Convolution Layer:

The most important portion or segment in the CNN algorithm. The mathematical notation of Convolution is as follows,

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

Image Source: superdatascience.com

here in the above equation, we can infer that convolution is an integration of two simple functions (f and g). We have two terms, one is our input/dataset in matrix format and another one is feature detector or kernel or filters (it will be 3*3 or 5*5 or 7*7 matrix, depends on the architecture that we are using!) when compared to the input image matrix, this matrix is smaller but we can get deeper knowledge from this kernel part only.

Consider an input and kernel matrix as

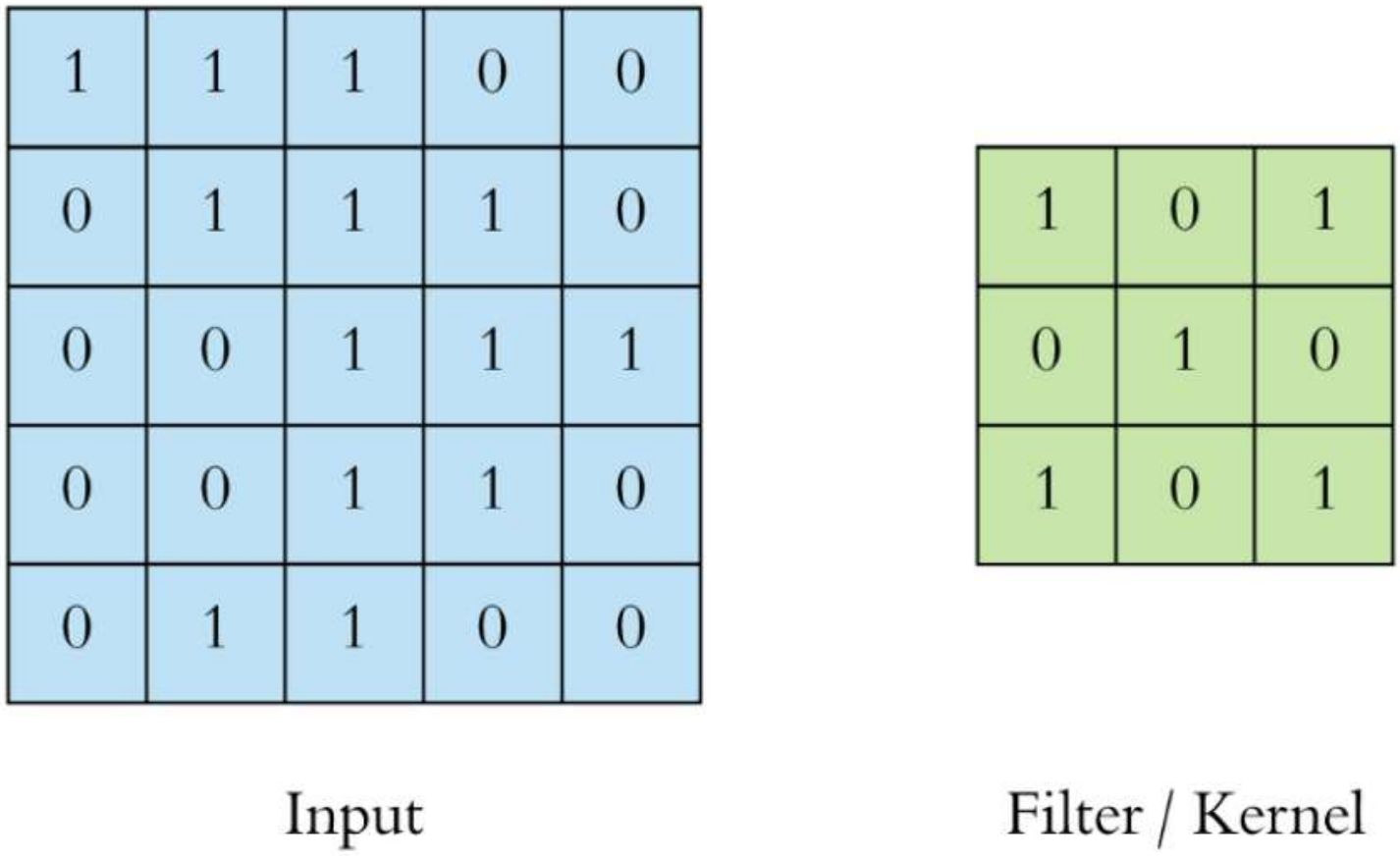


Image Source: towardsdatascience.com

The first step in convolution function is kernel matrix shown above figure, which slides through input image as,

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

In a static way, we can explain the above step as,

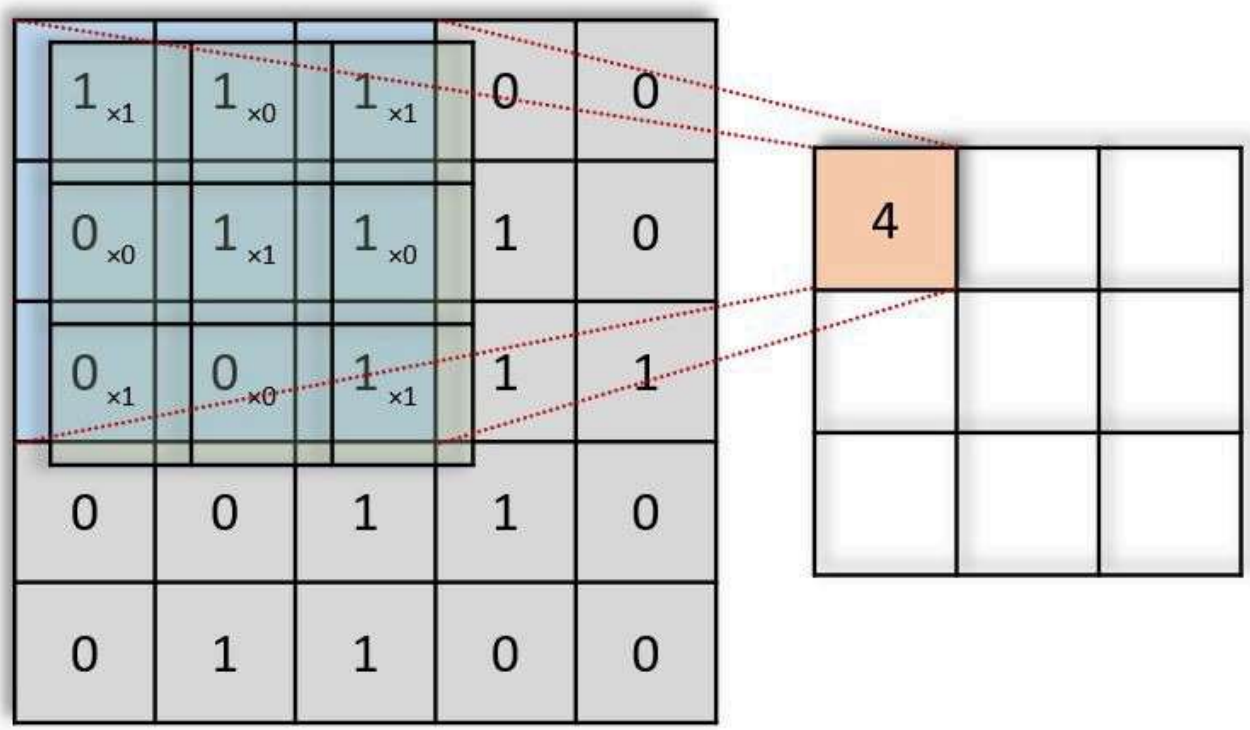
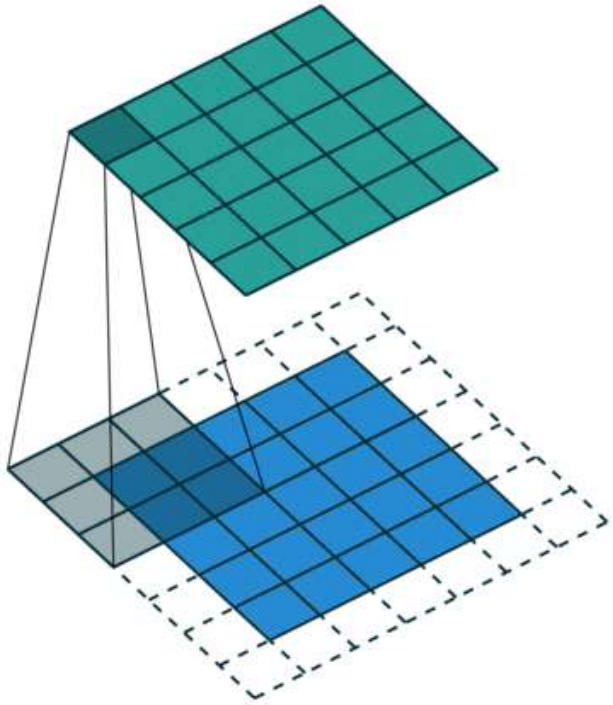


Image Source: towardsdatascience.com

If you note clearly, the value 4 in the output matrix, which depends on the 9 values of 3*3 matrix and another importance is the value doesn't change if any position of the value of the input matrix changes too. This is the RECEPTIVE FIELD of this output value, meaning the importance of the neuron layer to find a particular object or text or anything. Each value in the output value or matrix is sensitive to only a particular region, and hence we need to use our kernel matrix wisely depends on applications.

In a more layman way, convolution process `like,



For a change, let’s consider handwritten digits for a little more understanding through some pictures (as I believe the picture speaks 1000 words!). So here, in the below figure, I need to concentrate only horizontal lines alone, so for this with the help of kernel matrix I can get the output matrix as,

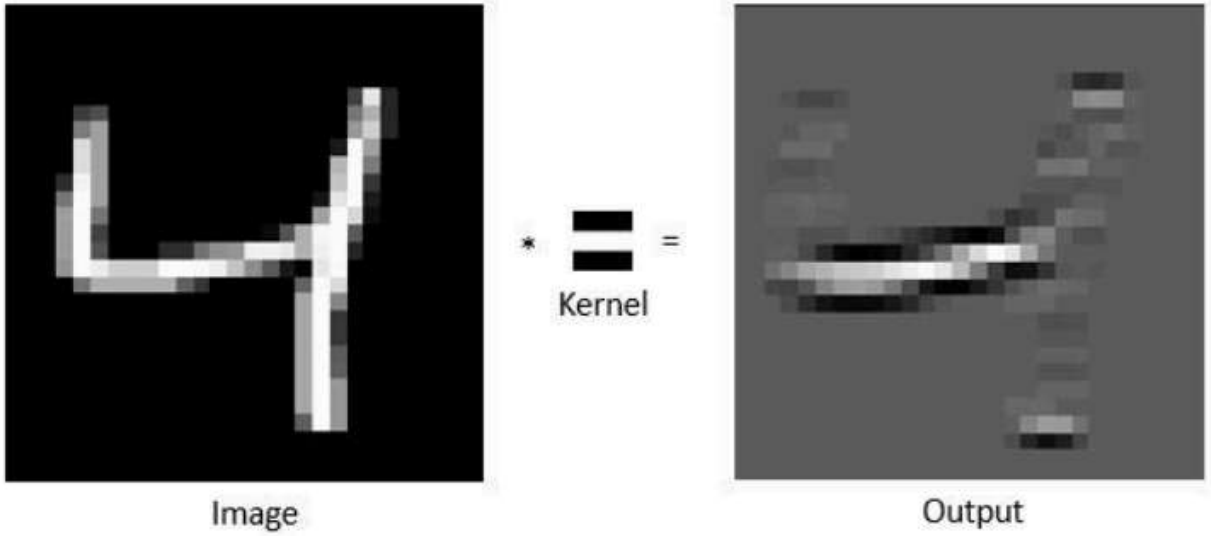


Image Source: towardsdatascience.com

In the same way, if I need to concentrate with respect to the vertical lines, in the same way by using kernel matrix, I can able to get the output values as,

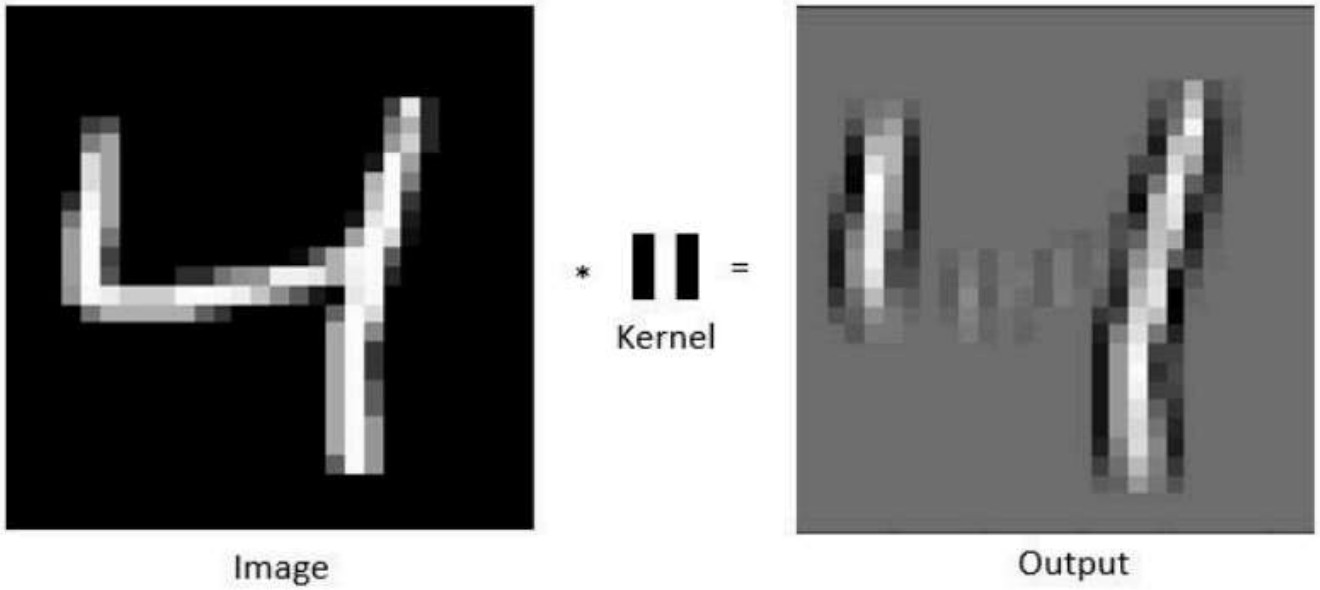


Image Source: towardsdatascience.com

So from all the explanations above, what we infer from this CONVOLUTION block, extraction of important terms, or technically we call it feature extraction. So convolution block is used to achieve feature extraction from the given dataset by avoiding other unimportant features or dimming or avoiding useless information or feature in simple.

Filters play's an important role in transforming the data (numbers/pixels), the resultant output we call activation maps, which means regions where features specific kernel to be detected from the input data. **Dimension of feature map = Dimension of filter**

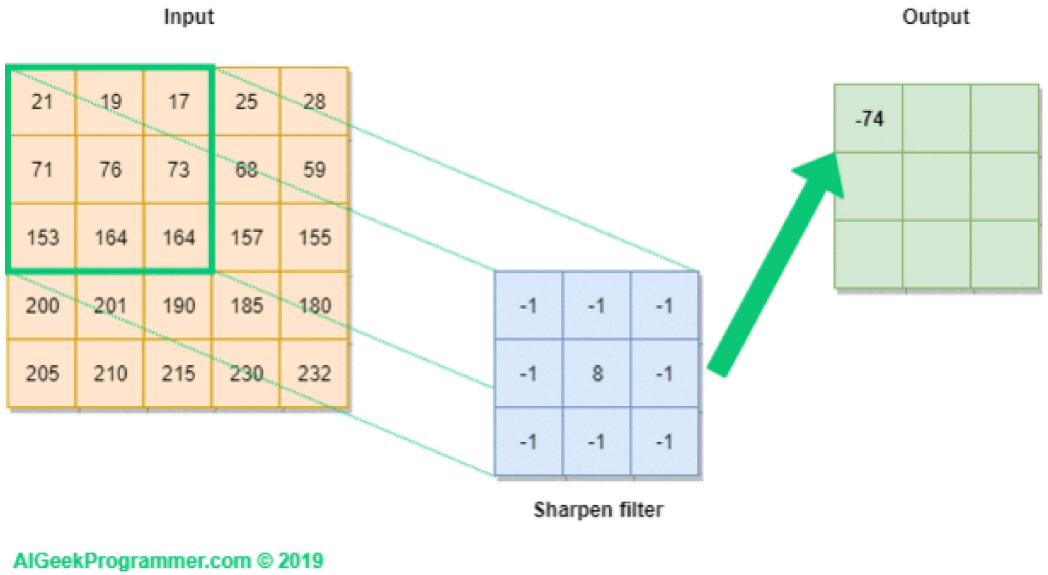


Image Source: https://medium.com/analytics-vidhya/convolutional-neural-network-cnn-and-its-application-all-u-need-to-know-f29c1d51b3e5

In the case of an image, convolution layer works like, because of RGB format wrt image,

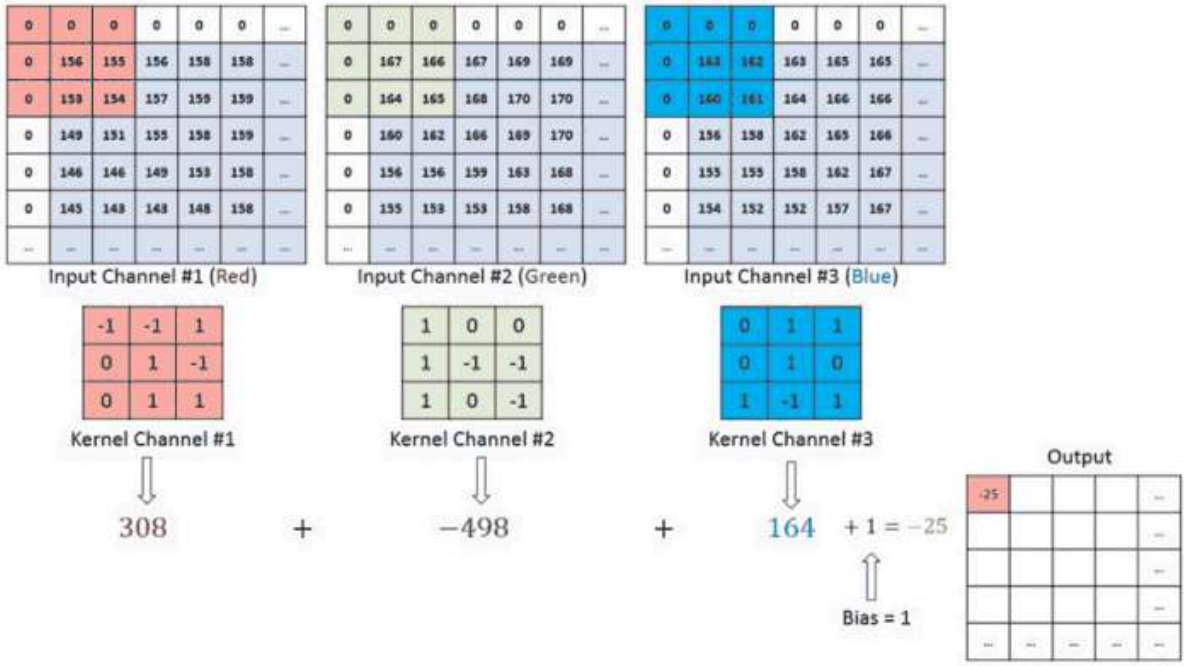


Image source: https://medium.com/analytics-vidhya/convolutional-neural-network-cnn-and-its-application-all-u-need-to-know-f29c1d51b3e5

2. Activation Function:

So once the convolution process gets over for our input dataset (sliding process!), the next important step to proceed in the CNN algorithm is the activation function.

When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding what is to be fired to the next neuron. It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell. It's a non-linear transformation that we do for inputs that we receive before sending them to the next layer or neuron. AF is also known as Transfer Function. For further information about AF visit the following link (https://www.analyticsvidhya.com/blog/2021/06/artificial-neural-networks-better-understanding/)

Let’s consider the same example as we have seen for the handwritten digit dataset, after the convolution process, we have passed AF (ReLU), then the output will be like,

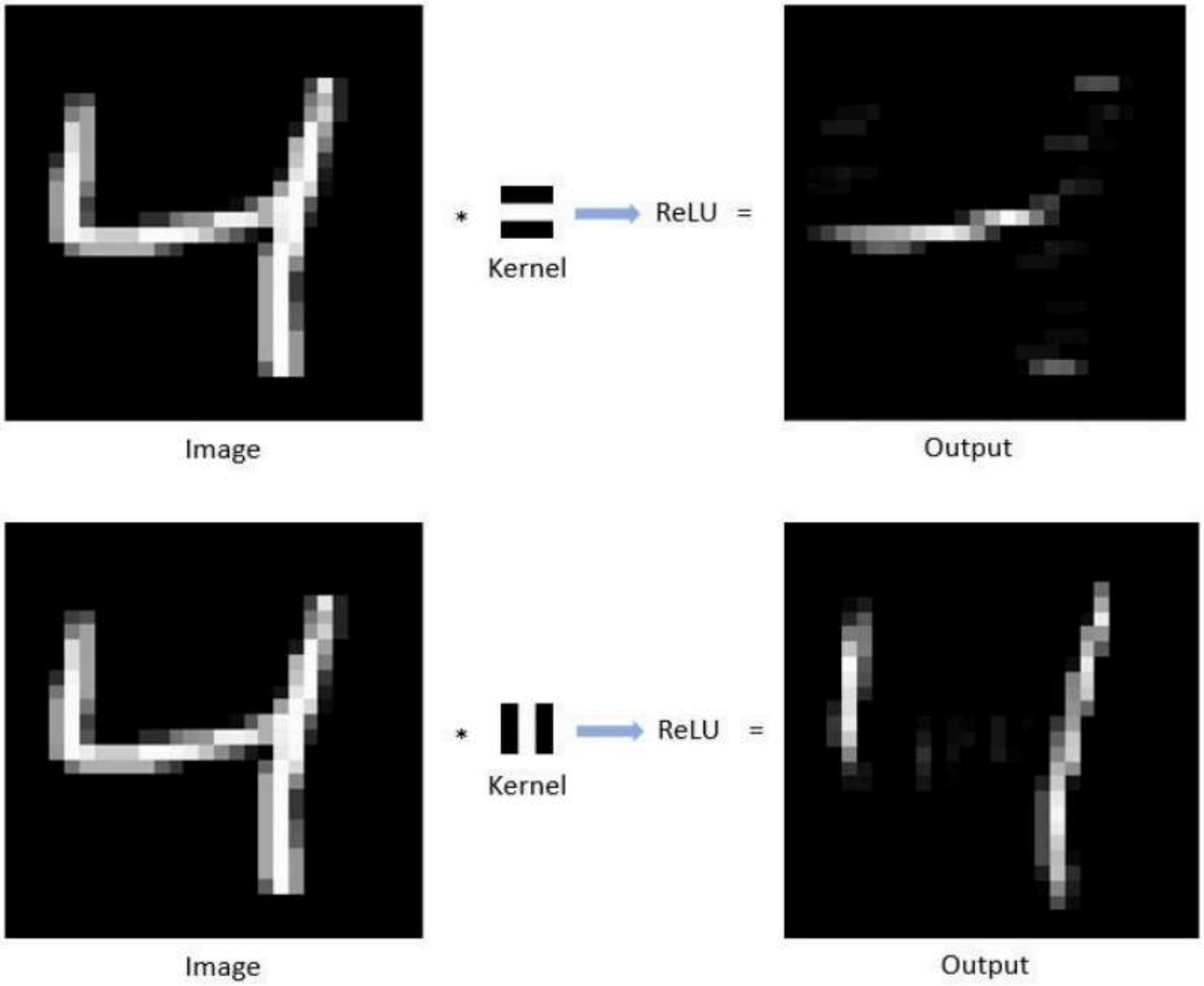


Image Source: towardsdatascience.com

So what you infer from this AF process, the output that we got from the convolution process in general one means it has some or few unwanted information, depending upon the kernel filter or matrix that we used. Here in AF, useful information is passed on to the next process. It's just transferring the information to the next process.

3. Pooling:

We all know the importance of swimming pool right – stress buster, if we do swimming for an hour, whatever stress we have it will be gone, in the same way, Pooling process which means it reduces the dimensionality of our matrix. The main importance of pooling is to reduce overfitting and computation in our dataset. In a layman level, pooling can be explained by,

The main importance of the pooling layer is to detect the edges, corners and in application-like images, it can be used to detect facial features like nose, eyes by using multiple filters. It's like filtering the feature map, because we get the feature map from the convolution process, and here in pooling we are going to apply it to the filter. Mostly the size of the filter is 2*2, which means in the pooling process feature map gets reduced by the factor of 2 (in simple reduced by half of the feature map dimension), intern it reduces each value in the feature map to one quarter.

It has two main types, Max-Pooling and Avg-Pooling.

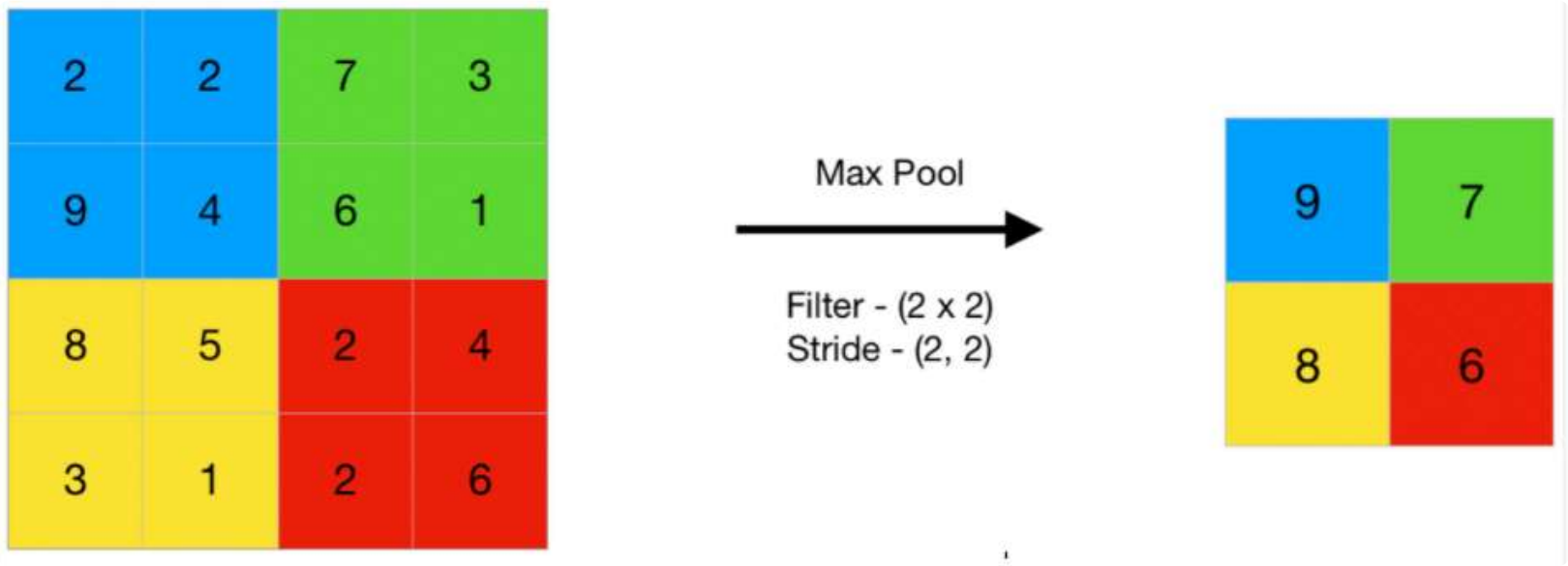


Image Source: towardsdatascience.com

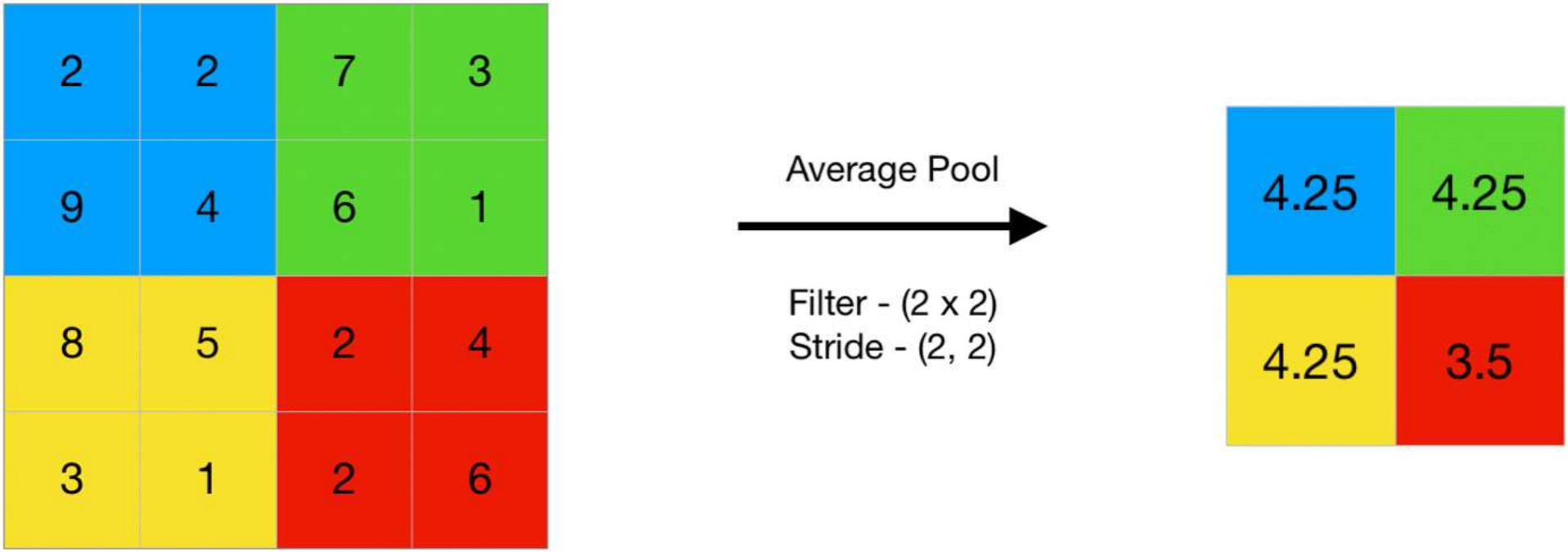


Image Source: towardsdatascience.com

and some other types of pooling are as follows, firstly will see a minimum pooling,

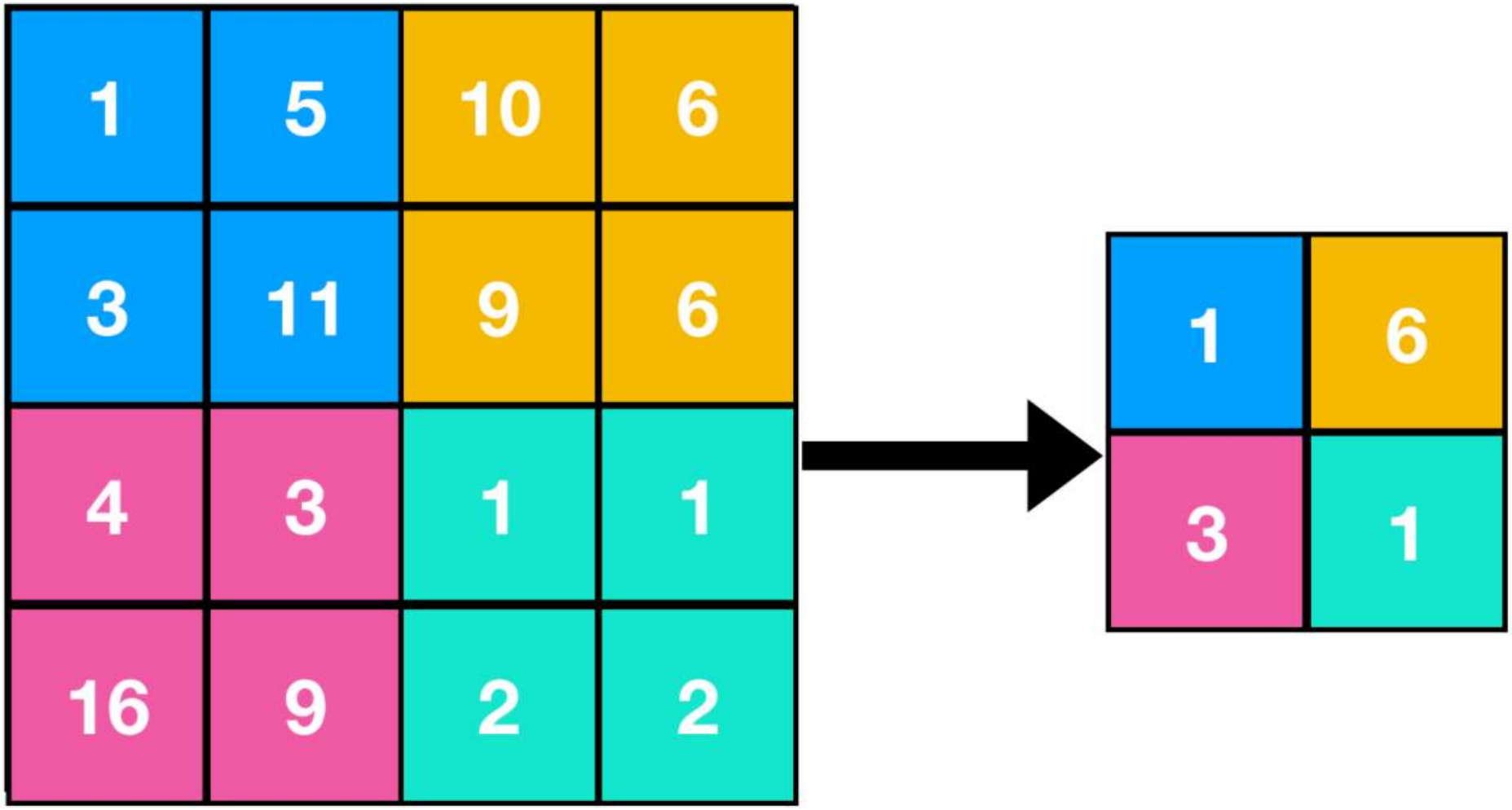


Image Source: towardsdatascience.com

and lastly adaptive pooling, herein the user doesn’t need to manually define hyperparameters, it needs to define only output size, and the parameters are picked up accordingly

In a more understanding, let's consider a cat image in a different position as like this figure,

Image Source: images. google.com

The importance of the convolution layer is to find the feature extraction or location of features through feature maps. But in the above figure if I apply the convolution it may get confused because the first half of the image is in reverse position, so here comes the pooling process, apart from the reduction process it also helps in Translational Invariance (ability to ignore the position shifts).

There are some important terms to be noticed in the Pooling process,

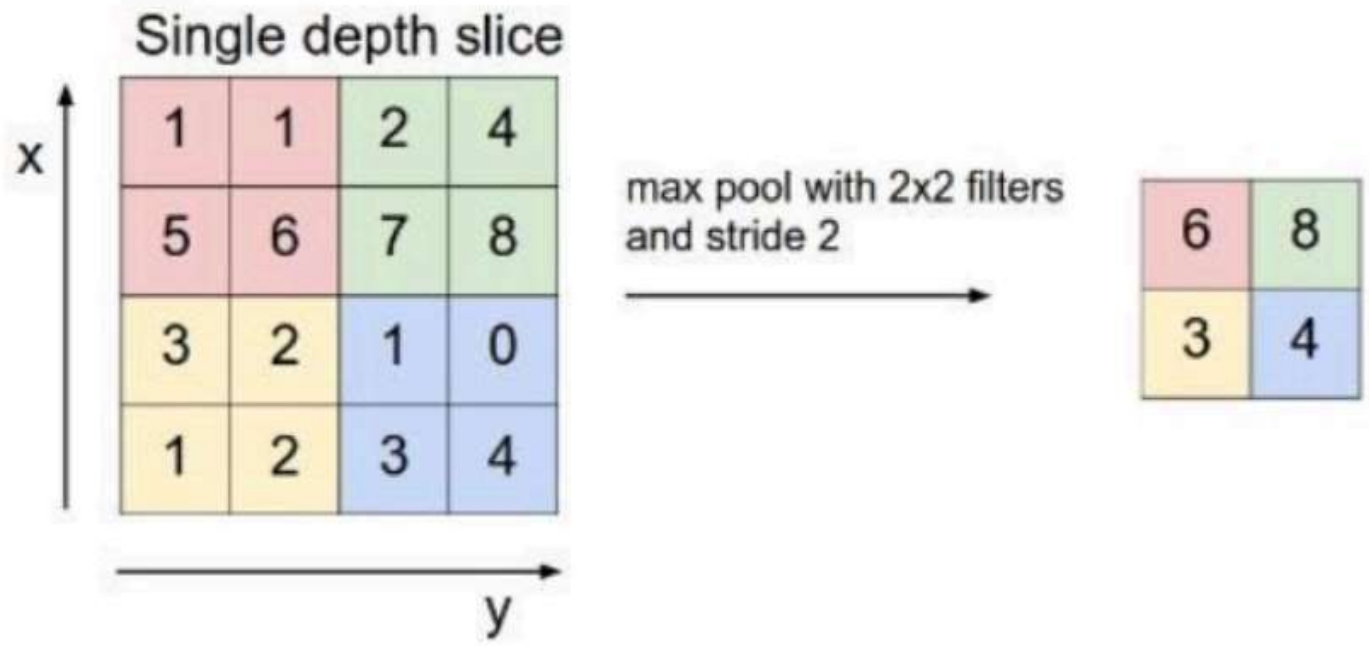
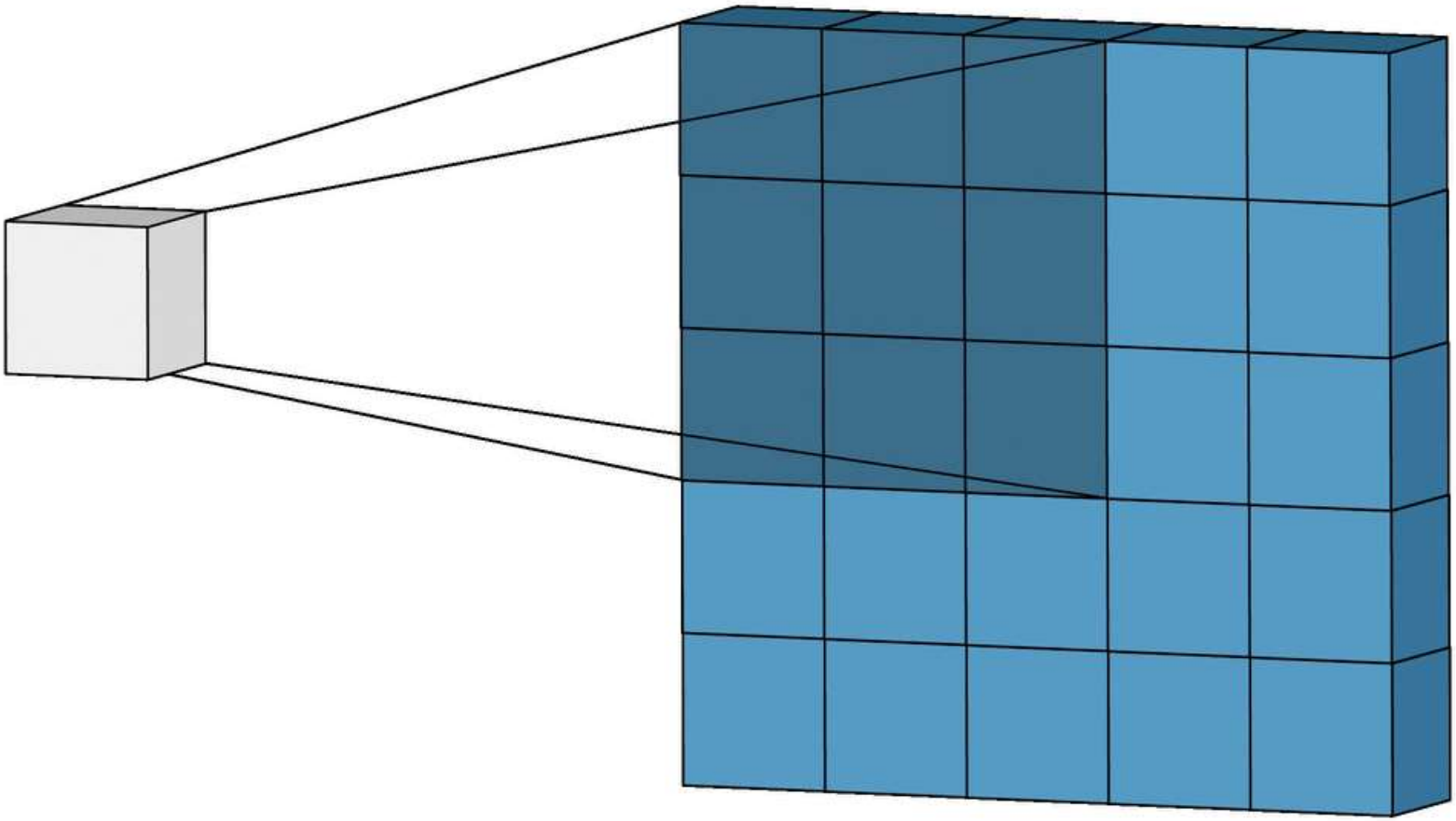


Image Source: medium.com

1. Filter Size – Describes filters size like 2*2 or 3*3 or anything

2. Stride – describes the number of steps filter takes while traversing the image

3. Padding – Sometimes filter size creates a border effect in the feature map, the effect can be overcome through padding.



If you notice in the above figure, the filter size is 3*3, stride size is 1.

4. **Flattening:**

So the next layer after the pooling process is flattening, here in this process, we are going to make our matrix into a single column, basically, we are taking row by row numbers and put them into a single column.

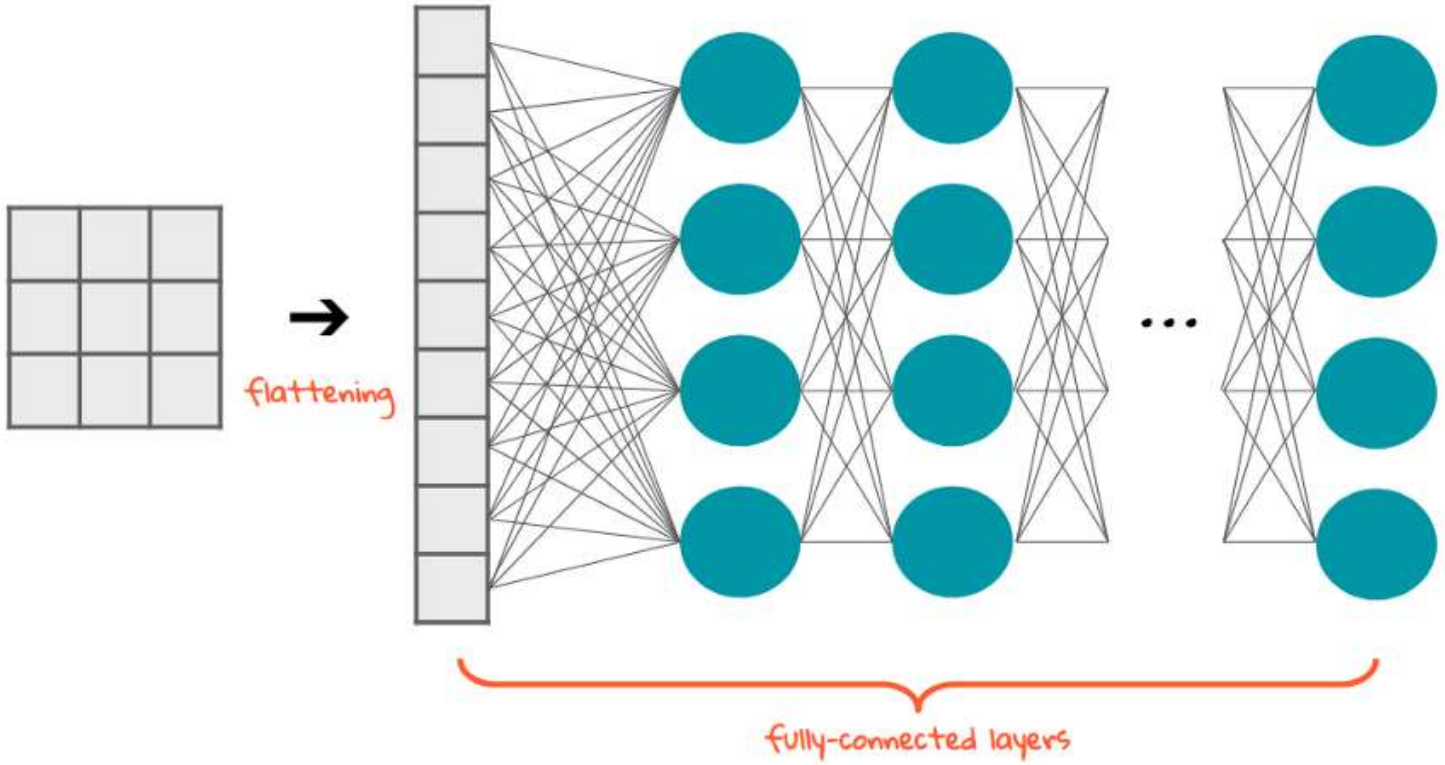


Image Source: https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480

And the importance of making them into a single column is simple, the next step is to feed them to the neural network for further processing.

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully connected layer.
In other words, we put all the pixel data in one line and make connections with the final layer. And once again. What is the final layer for? The classification of 'the cats and dogs' or normal and abnormal, good or bad, or any classification.

5. **Full Connection / Fully Connected layer:**

Last layer, in which each pixel or each number/point is considered as a separate neuron just like a NN. The last fully-connected layer will contain as many neurons as the number of classes to be predicted.

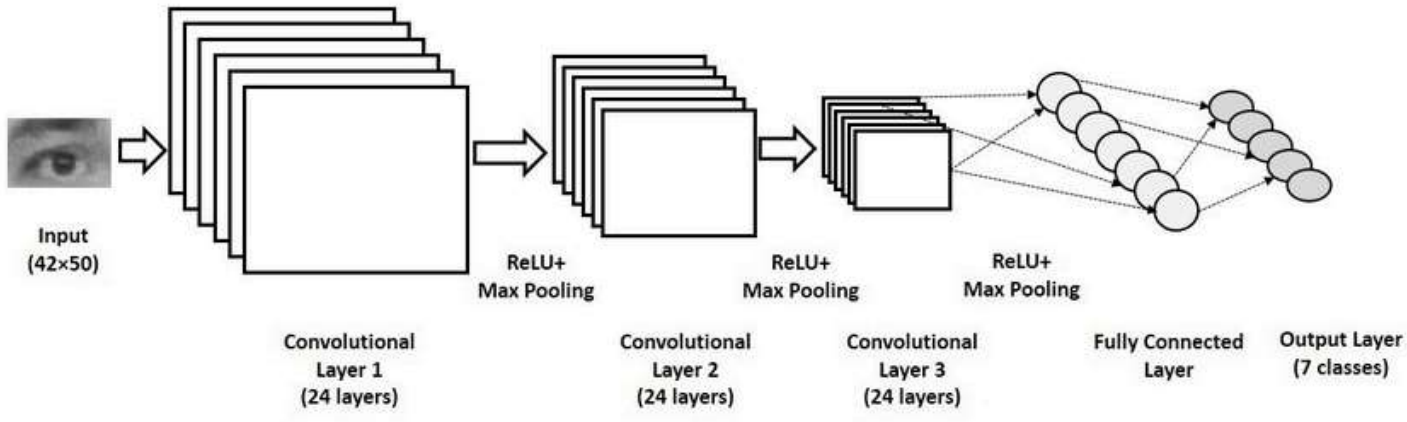


Image Source: https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480

Now that we have converted our input image into a suitable form for our Multi-Level fully connected architecture, we shall flatten the image into one column vector. The flattened output is fed to a feed-forward neural network and backpropagation is applied to every

iteration of training. Over a series of epochs, the model can distinguish between dominating and certain low-level features in images and classify them.

This layer assigns random weights to the inputs and predicts a suitable label

Apart from these blocks, there are also some special blocks used for CNN,

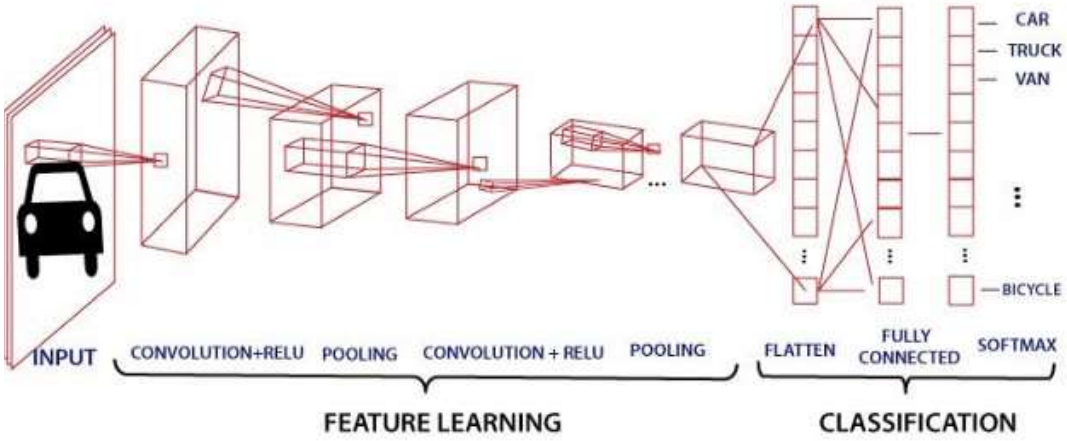


Image Source: https://medium.com/analytics-vidhya/convolutional-neural-network-cnn-and-its-application-all-u-need-to-know-f29c1d51b3e5

Fully Connected Output layer:

The final layer of the CNN model contains the results of the labels determined for the classification and assigns a class to the dataset (input)

Softmax

The reason why softmax is useful is that it converts the output of the last layer in your neural network into what is essentially a probability distribution. It is mainly used to normalize neural networks output to fit between zero and one. It is used to represent the

certainty “probability” in the network output.

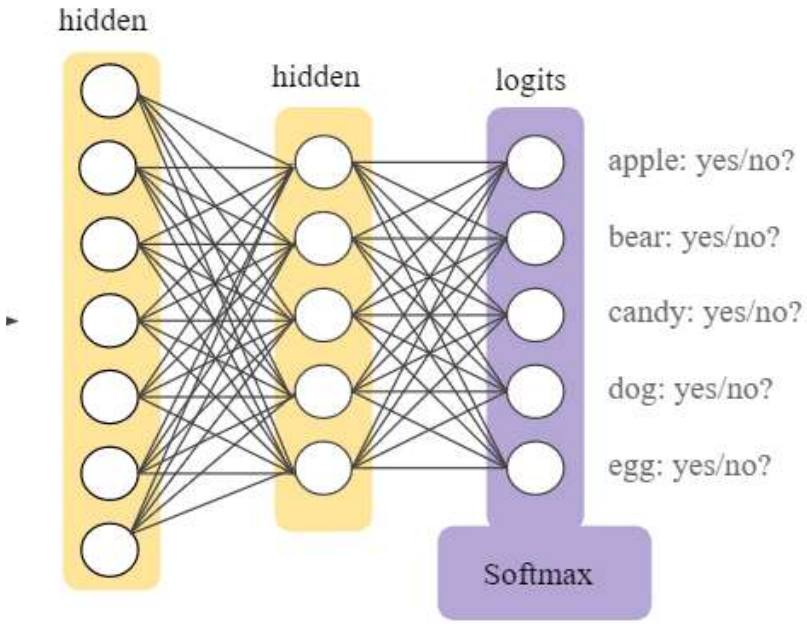


Image Source: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax

Dropout

The main idea behind using dropout is simple, to avoid overfitting. The most popular regularization method is used in Neural networks. Even if we are using a different model and achieve around 90% of accuracy, by using dropout we can able to achieve around 92% of accuracy.

During training time, at each iteration, a neuron is temporarily “dropped” or disabled with probability *p*. This means all the inputs and outputs to this neuron will be disabled at the current iteration. The dropped-out neurons are resampled with probability p at every

training step, so a dropped-out neuron at one step can be active at the next one. The hyperparameter *p* is called the dropout rate and it's typically a number around 0.5, corresponding to 50% of the neurons being dropped out.

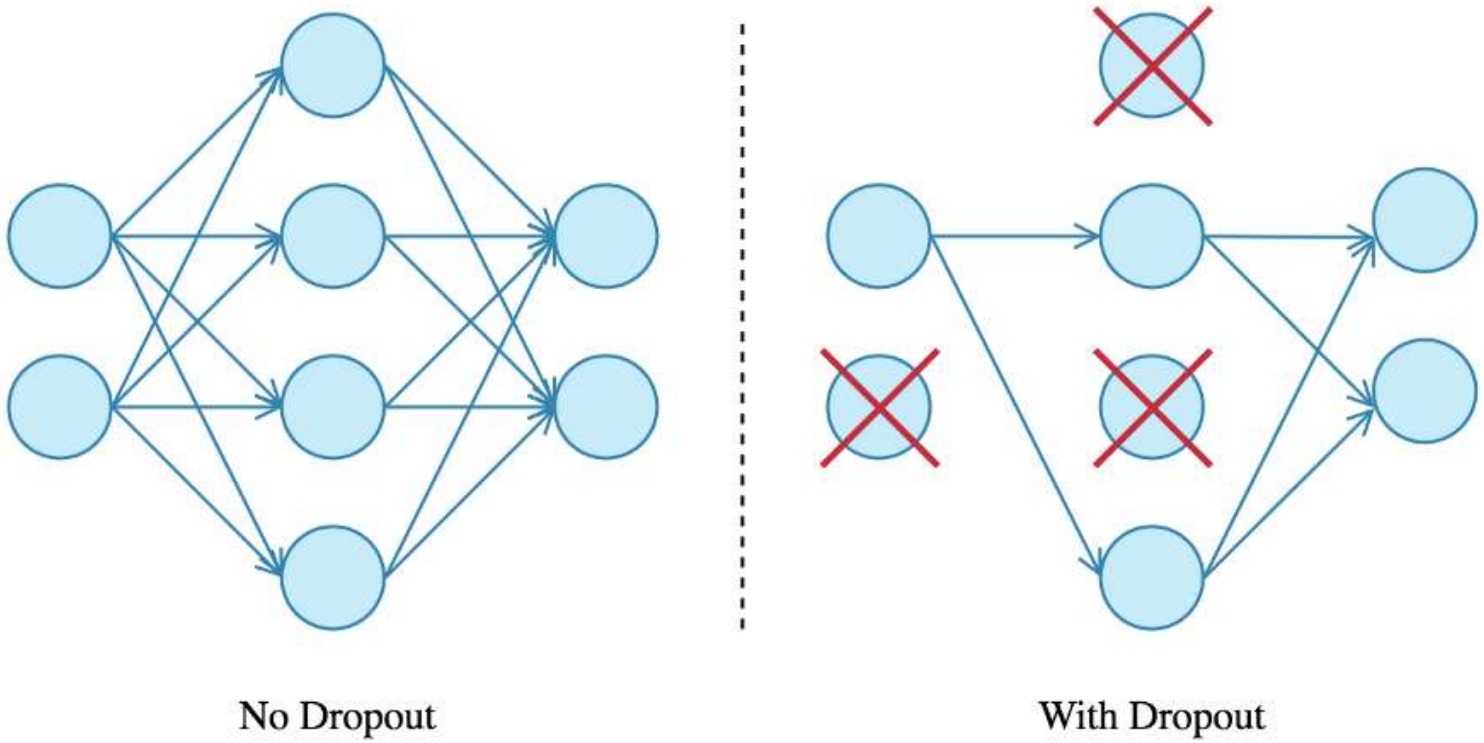


Image source: image.google.com

Almost all state-of-the-art deep networks now incorporate dropout. There is another very popular regularization technique called batch normalization

Summary of CNN:

1. Provide the data/dataset (any format of data) into the convolution layer

2. Take convolution with featured kernel/filters

3. Next apply a layer, Pooling for reducing the dimensions

4 If you need a deeper understanding add these layers repeatedly.

5. Next step is to apply to flatten for a single column

6. Feed into a fully connected layer

7. Now lastly train the model with appropriate metrics and optimizer

CNN – Different Architecture Models

Before getting deeper into the models, let's understand some basic terminologies,

1. **Wider –** means more feature maps (filters) in the convolutional layers

2. **Deeper –** means more convolutional layers

3. **High Resolution –** means that it processes input images with larger width and depth (spatial resolutions). That way the produced feature maps will have a higher spatial dimensions.

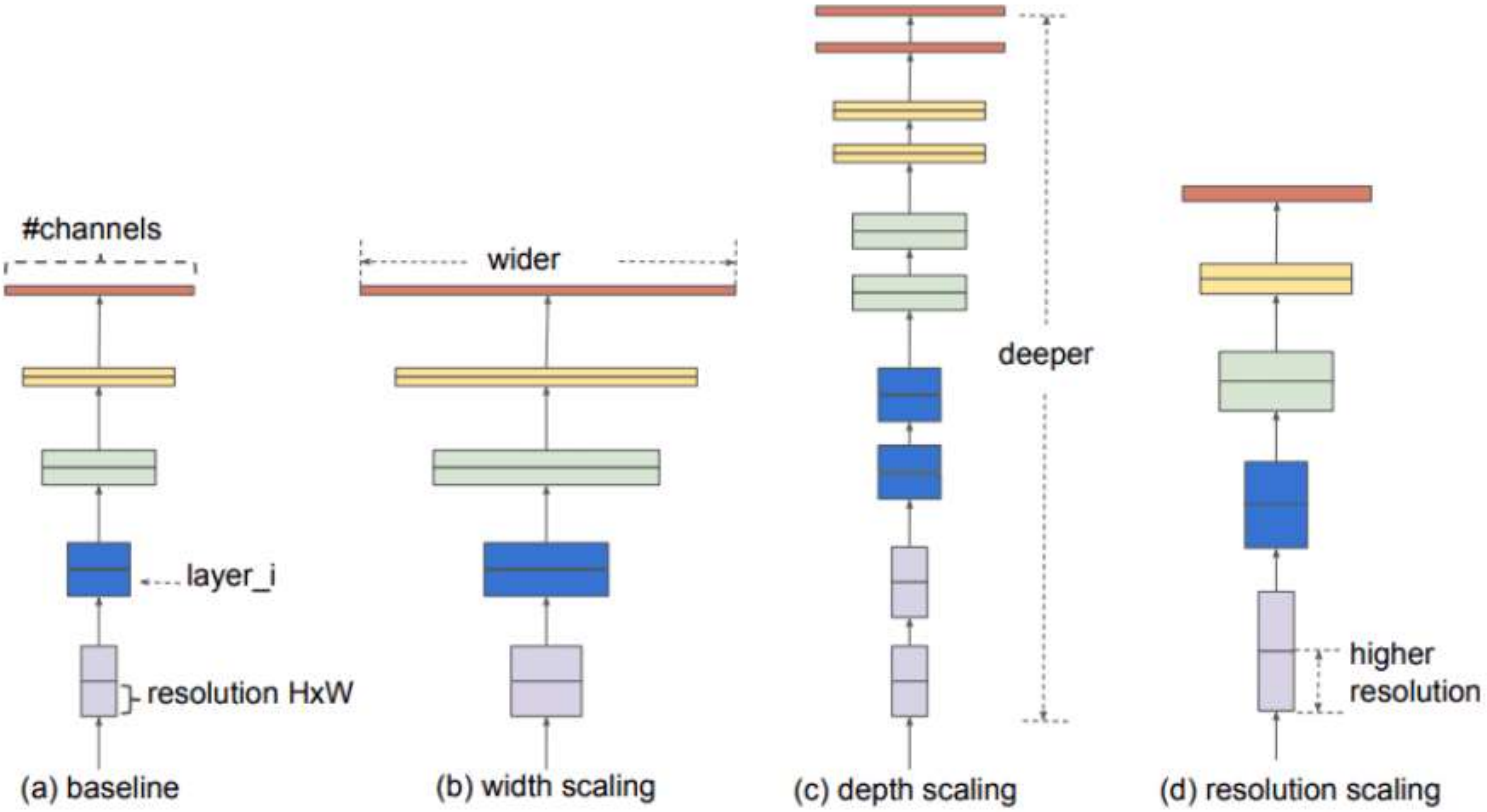
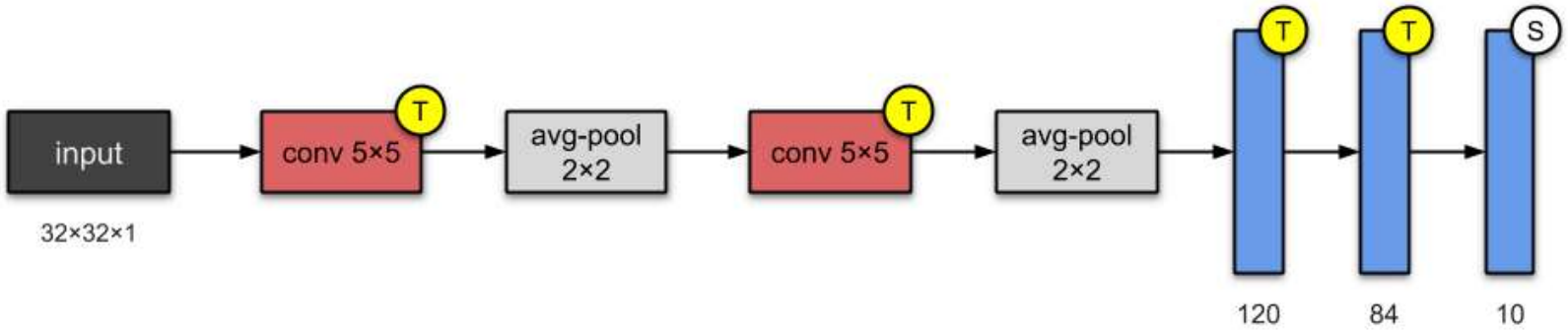


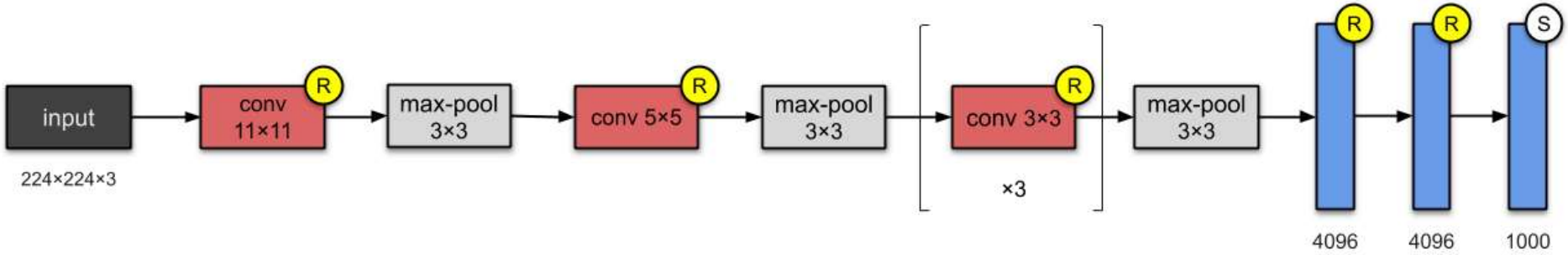
Image Source: https://arxiv.org/abs/1905.11946

Based on the architecture of layers that we have seen so far with some technical terms, CNN is categorized into different models, some of them are as follows,

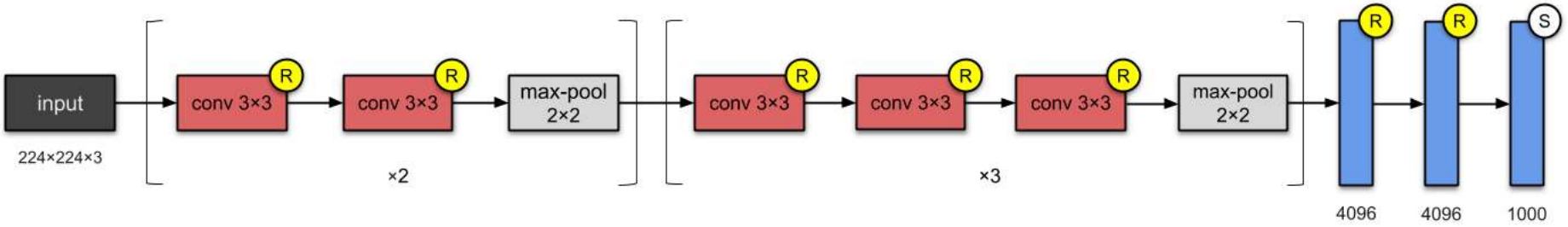
1. **LeNet-5** (2 – Convolution layer & 3 – Fully Connected layers) – 5 layers



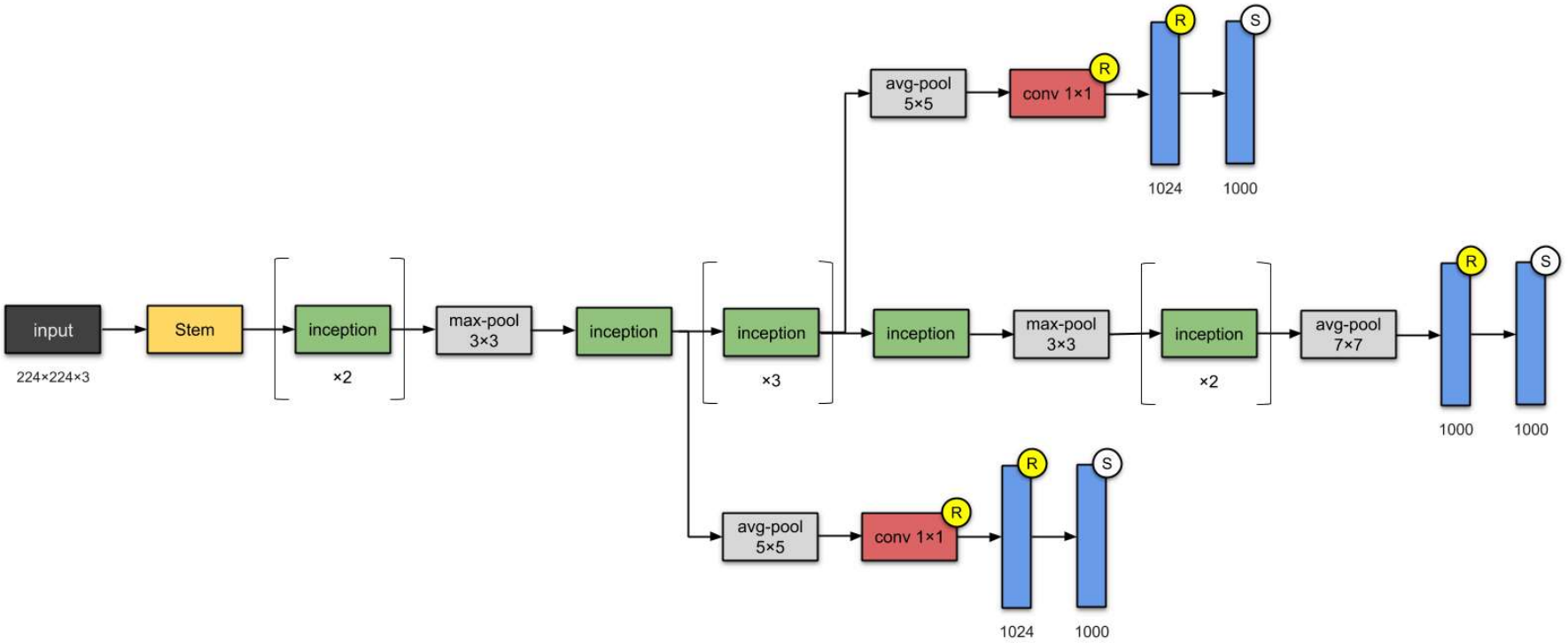
2. **AlexNet** (5 – Convolution layer & 3 – Fully Connected layers) – 8 layers

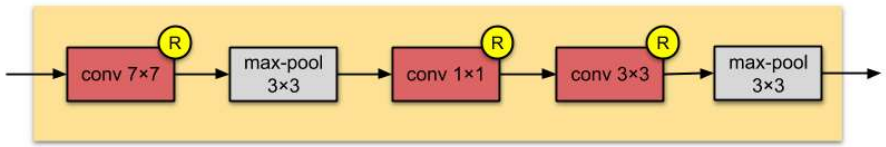


3. **VGG 16** (13 – Convolution layer & 3 – Fully Connected layers) – 16 layers

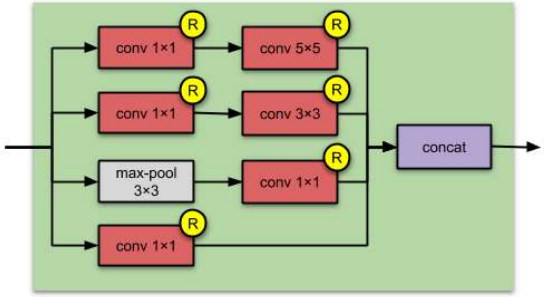


4. **Inception V1** – 22 layers



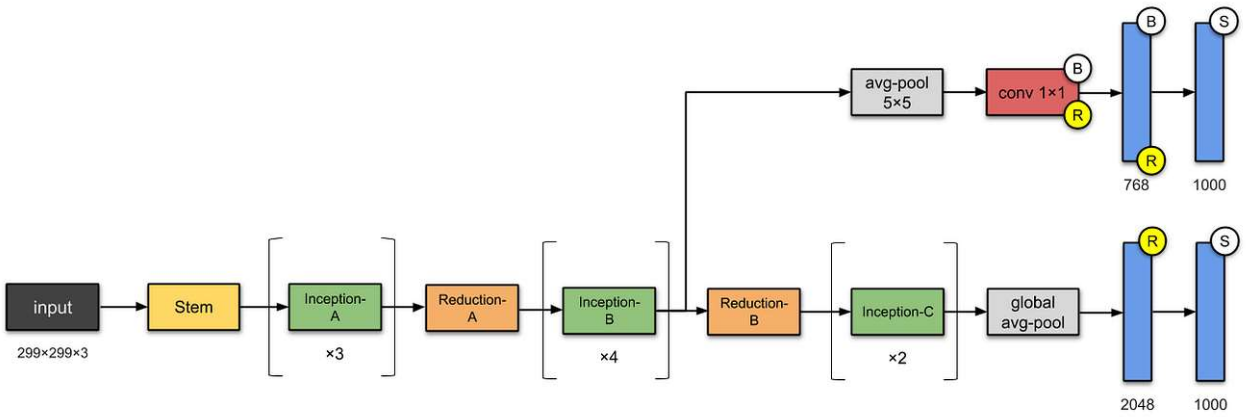


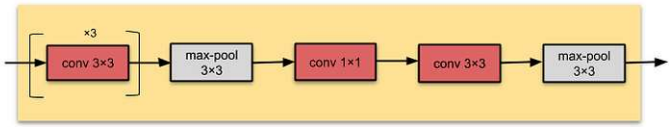
Stem



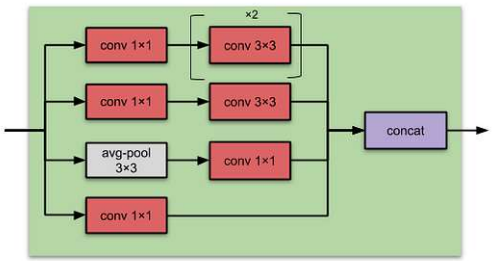
Inception module

5. **Inception V3**

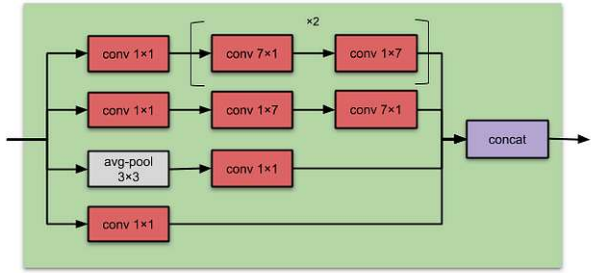




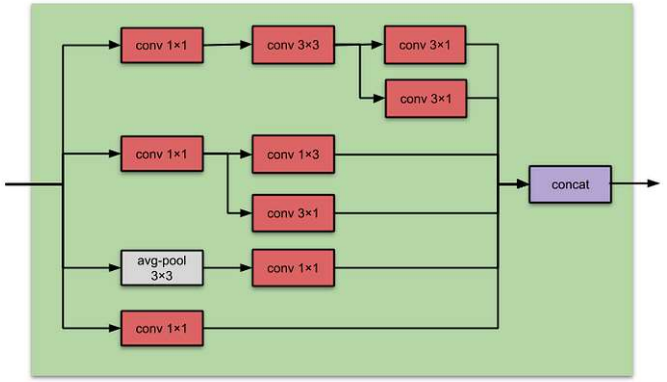
Stem



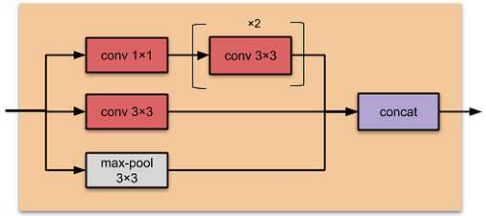
Inception-A



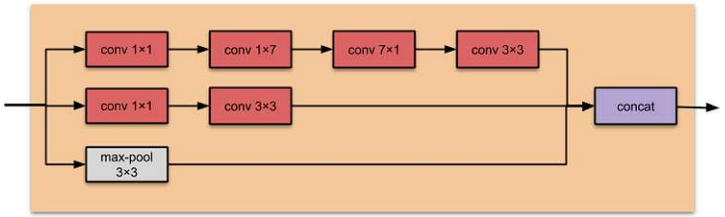
Inception-B



Inception-C

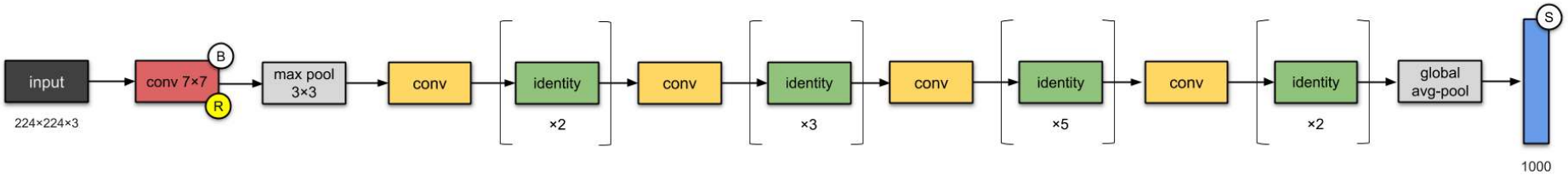


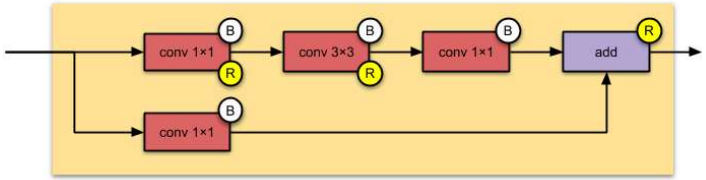
Reduction-A



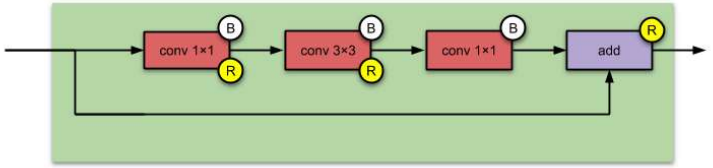
Reduction-B

6. **ResNet 50**



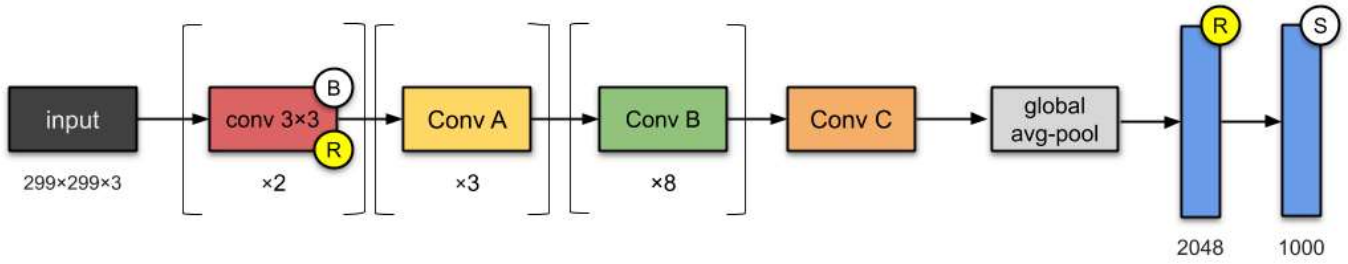


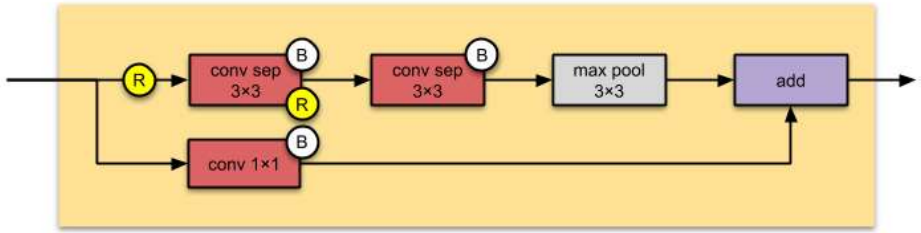
Conv block



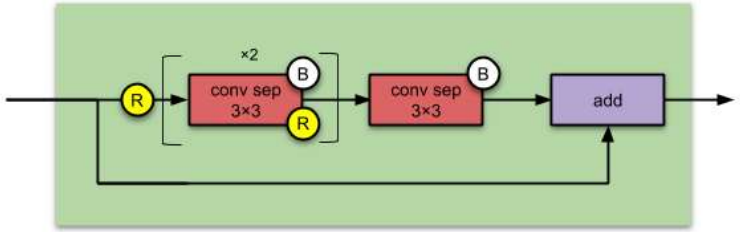
Identity block

7. **Xception**

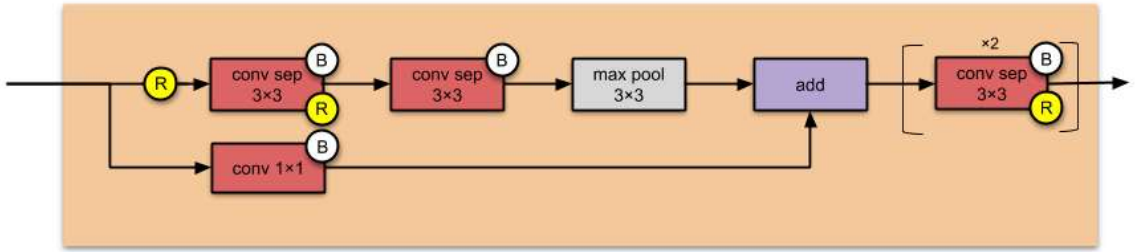




Conv A

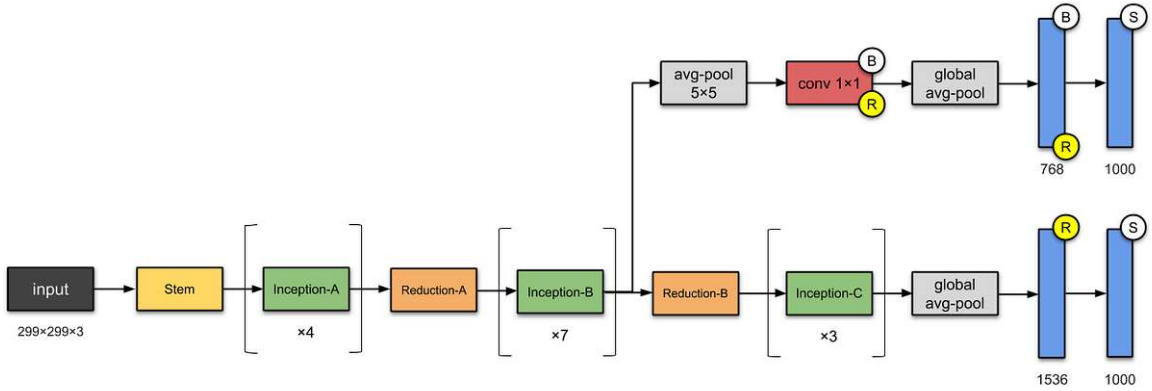


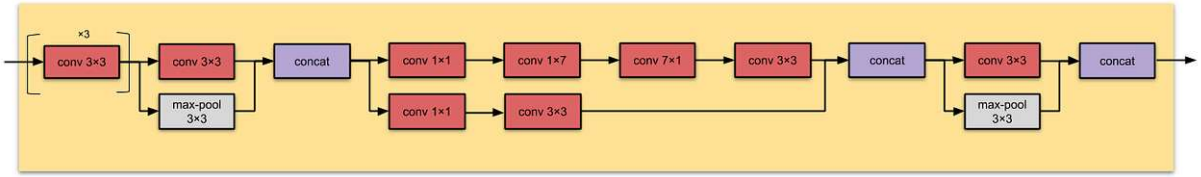
Conv B



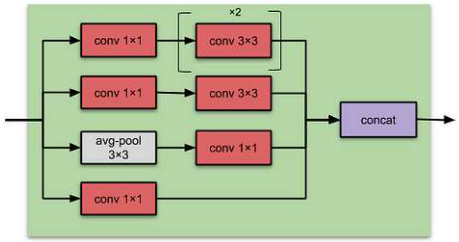
Conv C

8. **Inception V4**

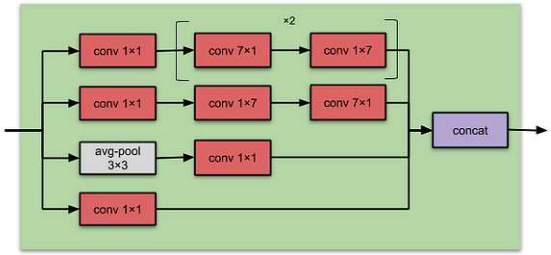




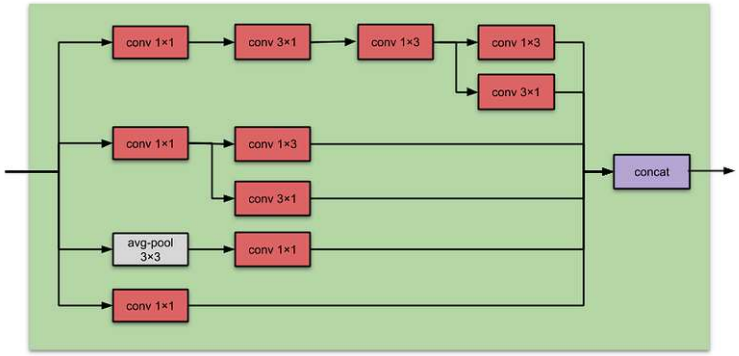
Stem



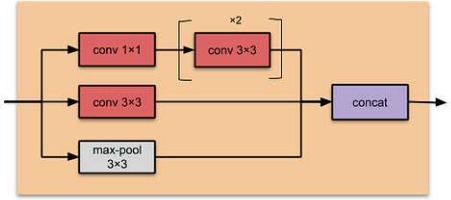
Inception-A



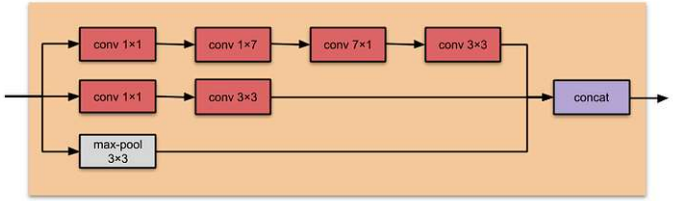
Inception-B



Inception-C

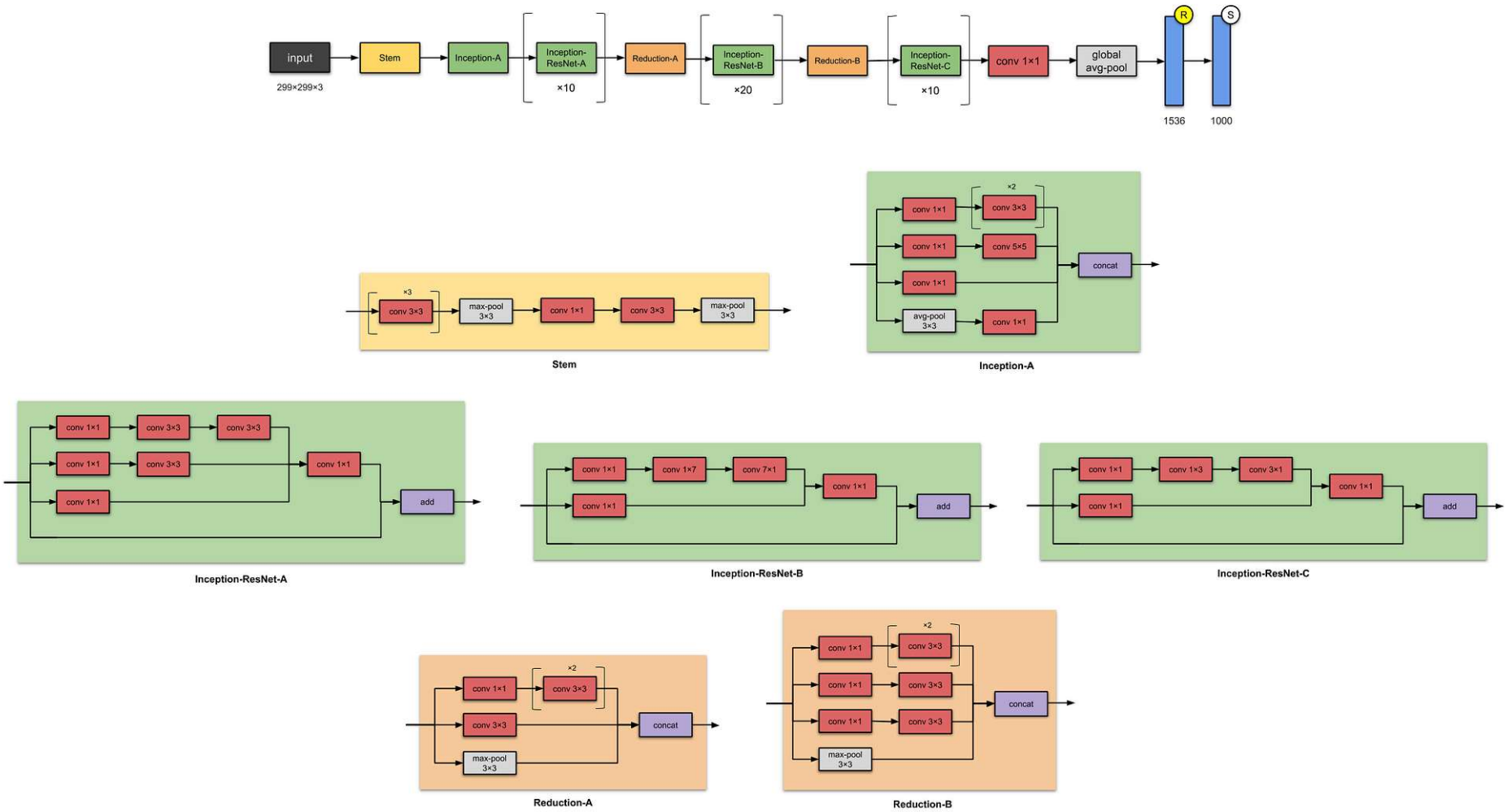


Reduction-A

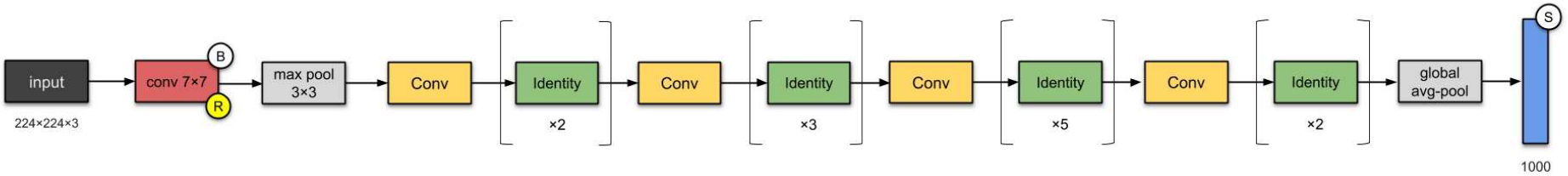


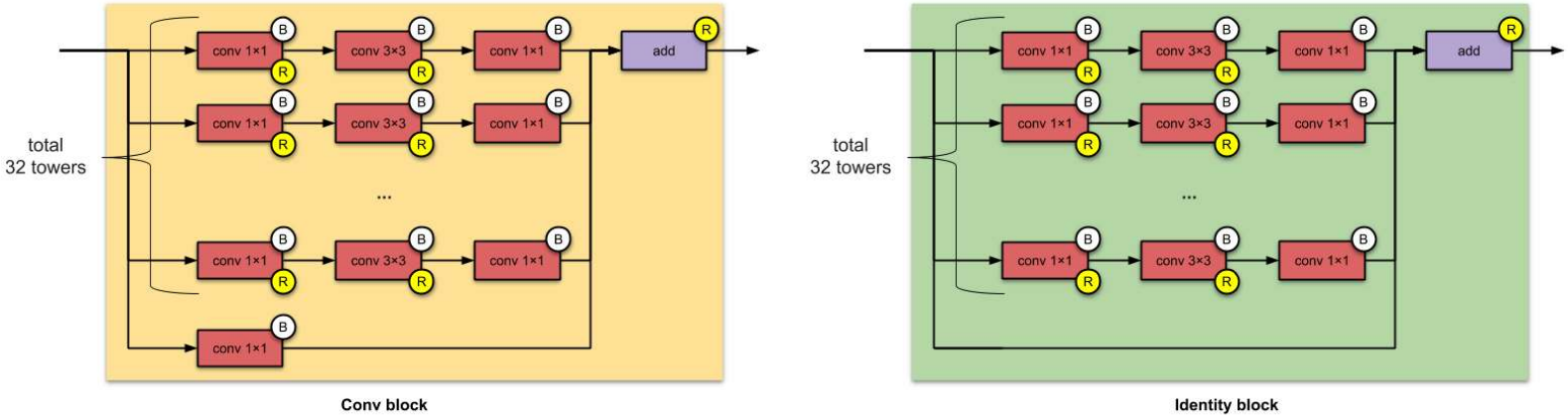
Reduction-B

9. **Inception ResNet V2**



10. **ResNeXt 50**





All the pictures are taken from https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

11. **Squeeze Net**

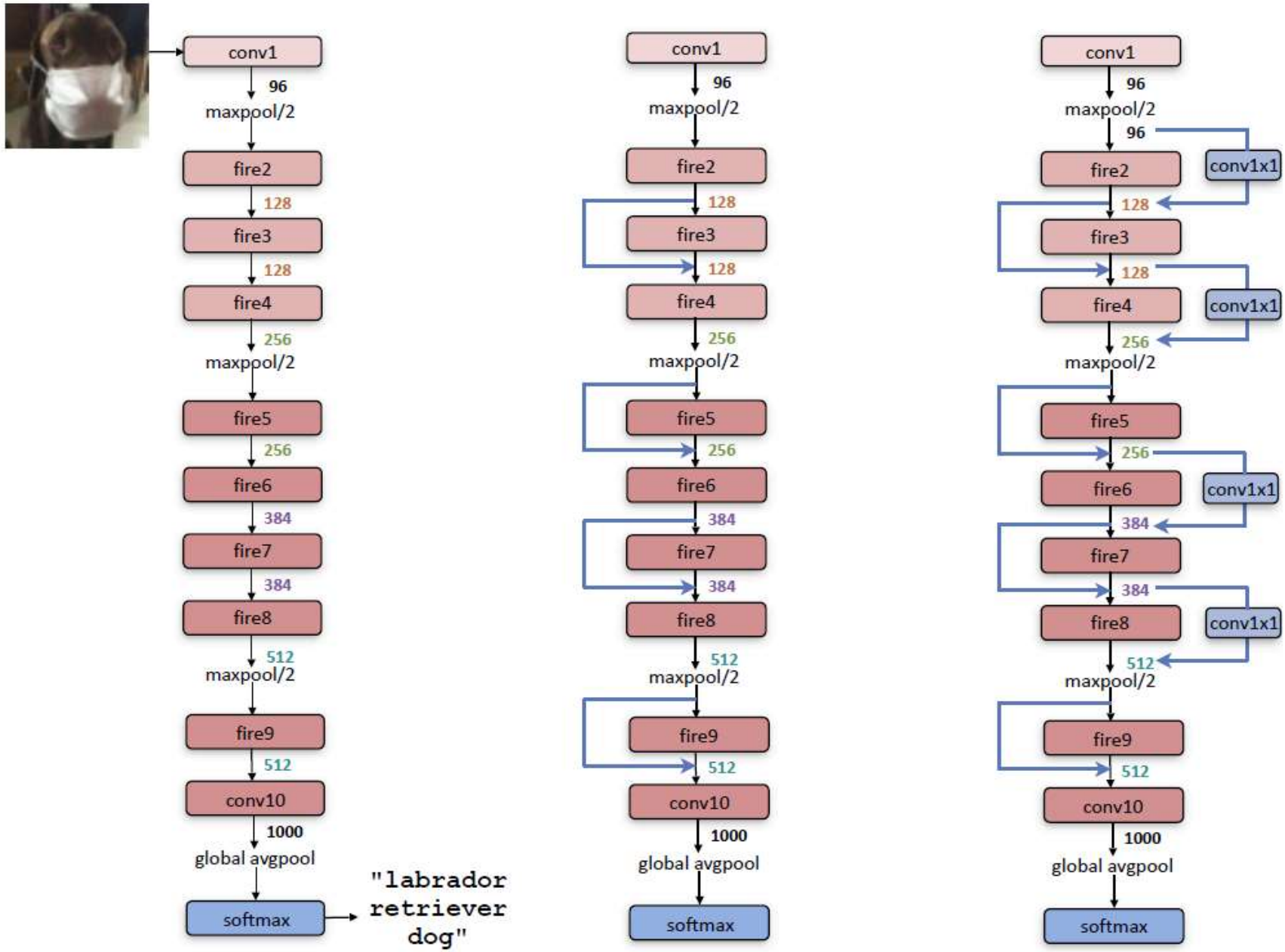


Image Source: https://towardsdatascience.com/review-squeezenet-image-classification-e7414825581a

12. **Dense Net**

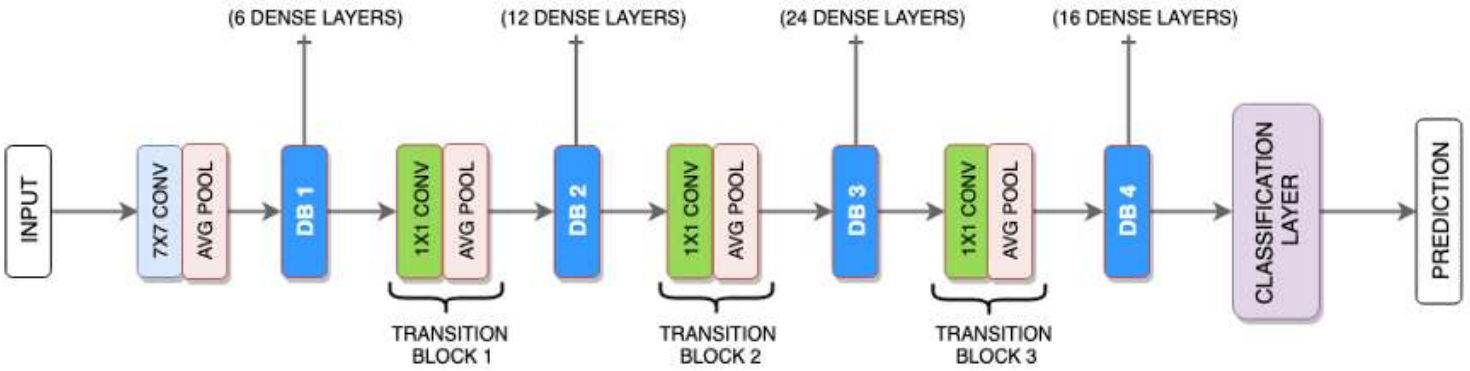


Image Source: https://towardsdatascience.com/paper-review-densenet-densely-connected-convolutional-networks-acf9065dfefb

13 **Shuffle Net**

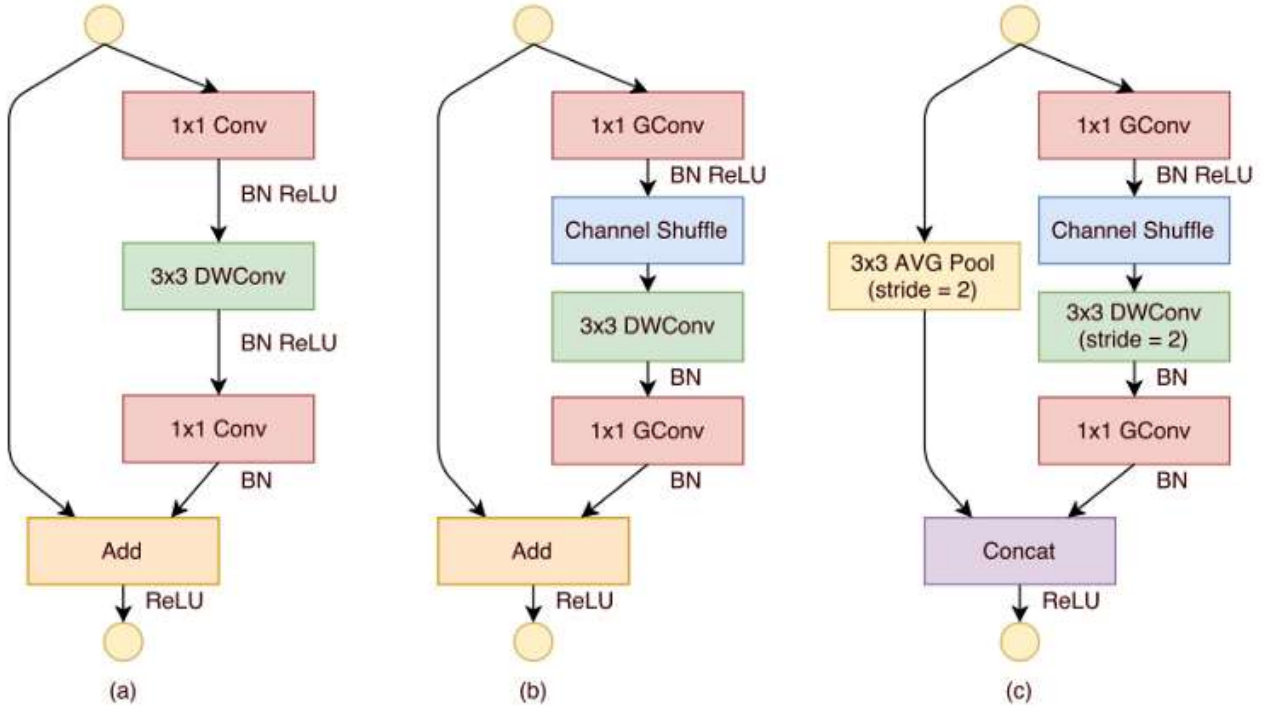


Image Source: https://medium.com/syncedreview/shufflenet-an-extremely-efficient-convolutional-neural-network-for-mobile-devices-72c6f5b01651

14 **ENet**

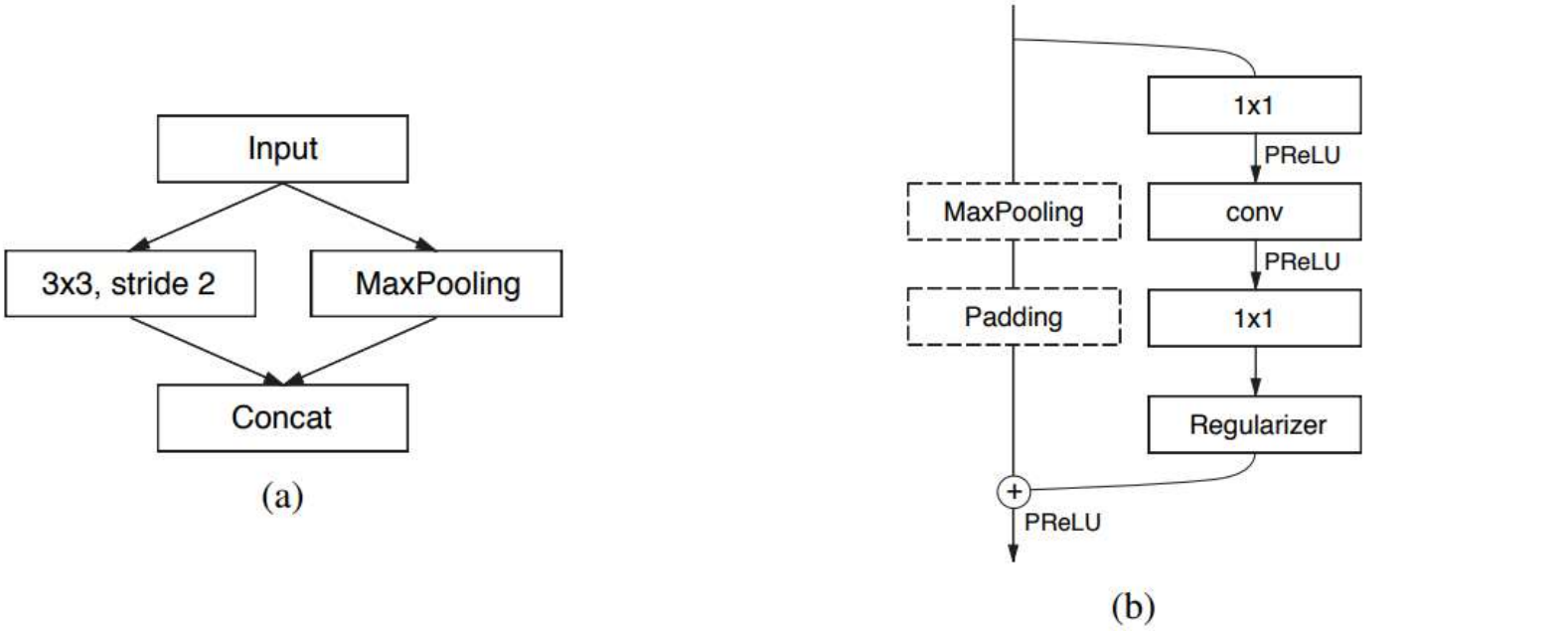


Figure 2: (a) ENet initial block. MaxPooling is performed with non-overlapping 2×2 windows, and the convolution has 13 filters, which sums up to 16 feature maps after concatenation. This is heavily inspired by [28]. (b) ENet bottleneck module. `conv` is either a regular, dilated, or full convolution (also known as deconvolution) with 3×3 filters, or a 5×5 convolution decomposed into two asymmetric ones.

Image Source: https://towardsdatascience.com/enet-a-deep-neural-architecture-for-real-time-semantic-segmentation-2baa59cf97e9

Some of the most famous model comparisons are shown below,

Model name	Number of parameters [Millions]	ImageNet Top 1 Accuracy	Year
AlexNet	60 M	63.3 %	2012
Inception V1	5 M	69.8 %	2014
VGG 16	138 M	74.4 %	2014
VGG 19	144 M	74.5 %	2014
Inception V2	11.2 M	74.8 %	2015
ResNet-50	26 M	77.15 %	2015
ResNet-152	60 M	78.57 %	2015
Inception V3	27 M	78.8 %	2015
DenseNet-121	8 M	74.98 %	2016
DenseNet-264	22M	77.85 %	2016
BiT-L (ResNet)	**928 M**	87.54 %	2019
NoisyStudent EfficientNet-L2	**480 M**	**88.4** %	2020
Meta Pseudo Labels	**480 M**	**90.2** %	2021

Image Source: https://theaisummer.com/cnn-architectures/

Advantage of CNN:

1. Automatically detects important features without any human supervision

2. Computationally efficient (Convolution and Pooling process performs parameter sharing)

3. Weight Sharing

4. High Statistical efficiency

5. Translational Invariance

Disadvantages of CNN:

1. Classification of Images with different Positions (different angles, different backgrounds, different lighting conditions)

2. Adversarial examples (CNN takes an image along with some noise it recognizes the image as a completely different image whereas the human visual system will identify it as the same image with the noise)

3. Coordinate Frame (Coordinate frame is basically a mental model which keeps track of the orientation and different features of an object)

4. Slower due to functions like Maxpool

5. As several layers are functional then the training process takes lots of time if our CPU doesn't have a good GPU

These drawbacks lead to the formation of the Capsule Neural Network

Application of CNN:

1. Decoding Facial Recognition

2. Analysing Documents

3. Classification

4. Segmentation (Organs or any anatomical structures is a fundamental Image Processing technique for Medical Image Analysis)

5. Detection (Common task for Radiologist is to detect abnormalities with Medical Images)

Python Coding with CNN

I hope you understand some bit about CNN from the above explanation and some visualization,

Dataset: CIFAR 10 Image Classification

Copy Code

```
#importing basic libraries

import tensorflow as tf

import matplotlib.pyplot as plt

#importing default dataset

from tensorflow.keras.datasets import cifar10
#loading dataset

# setting class names from dataset

class_names=['airplane','automobile','bird','cat','deer','dog','frog','horse','ship','truck']

#loading the dataset

(x_train,y_train),(x_test,y_test)=cifar10.load_data()

#normalizing train data
```

```
x_train=x_train/255.0

x_train.shape

#normalizing test data

x_test=x_test/255.0

x_test.shape
#random check dataset

plt.imshow(x_test[222])

#Building CNN model

cnn_model=tf.keras.models.Sequential()

# First Layer

cnn_model.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding="same", activation="relu", input_shape=[32,32,3]))

# Second Layer

cnn_model.add(tf.keras.layers.Conv2D(filters=32,kernel_size=3,padding="same", activation="relu"))

# Max Pooling Layer

cnn_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2,padding='valid'))

# Third Layer

cnn_model.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding="same", activation="relu"))

# Fourth Layer

cnn_model.add(tf.keras.layers.Conv2D(filters=64,kernel_size=3,padding="same", activation="relu"))

# Max Pooling Layer

cnn_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2,padding='valid'))
```

```
    # fifth Layer
    cnn_model.add(tf.keras.layers.Conv2D(filters=128,kernel_size=3,padding="same", activation="relu"))
    # Sixth Layer
    cnn_model.add(tf.keras.layers.Conv2D(filters=128,kernel_size=3,padding="same", activation="relu"))
    # Max Pooling Layer
    cnn_model.add(tf.keras.layers.MaxPool2D(pool_size=2,strides=2,padding='valid'))
    # Flattening Layer
    cnn_model.add(tf.keras.layers.Flatten())
    # Droput Layer
    cnn_model.add(tf.keras.layers.Dropout(0.5,noise_shape=None,seed=None))
    # Adding the first fully connected layer
    cnn_model.add(tf.keras.layers.Dense(units=128,activation='relu'))
    # Output Layer
    cnn_model.add(tf.keras.layers.Dense(units=10,activation='softmax'))

cnn_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dropout (Dropout)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 550,570
Trainable params: 550,570
Non-trainable params: 0
```

Image Source: Author

Copy Code

```
#Model -compiling
cnn_model.compile(loss="sparse_categorical_crossentropy", optimizer="Adam", metrics=["sparse_categorical_accuracy"])
```

```
#Model - training
cnn_model.fit(x_train,y_train,epochs=15)
```

```
Epoch 1/15
1563/1563 [==============================] - 115s 73ms/step - loss: 1.8392 - sparse_categorical_accuracy: 0.3107
Epoch 2/15
1563/1563 [==============================] - 115s 73ms/step - loss: 1.1402 - sparse_categorical_accuracy: 0.5918
Epoch 3/15
1563/1563 [==============================] - 115s 74ms/step - loss: 0.9228 - sparse_categorical_accuracy: 0.6776
Epoch 4/15
1563/1563 [==============================] - 114s 73ms/step - loss: 0.7929 - sparse_categorical_accuracy: 0.7173
Epoch 5/15
1563/1563 [==============================] - 115s 74ms/step - loss: 0.7003 - sparse_categorical_accuracy: 0.7536
Epoch 6/15
1563/1563 [==============================] - 116s 74ms/step - loss: 0.6416 - sparse_categorical_accuracy: 0.7737
Epoch 7/15
1563/1563 [==============================] - 118s 75ms/step - loss: 0.5809 - sparse_categorical_accuracy: 0.7933
Epoch 8/15
1563/1563 [==============================] - 119s 76ms/step - loss: 0.5507 - sparse_categorical_accuracy: 0.8052
Epoch 9/15
1563/1563 [==============================] - 121s 78ms/step - loss: 0.5138 - sparse_categorical_accuracy: 0.8182
Epoch 10/15
1563/1563 [==============================] - 117s 75ms/step - loss: 0.4861 - sparse_categorical_accuracy: 0.8290
Epoch 11/15
1563/1563 [==============================] - 116s 74ms/step - loss: 0.4568 - sparse_categorical_accuracy: 0.8391
Epoch 12/15
1563/1563 [==============================] - 116s 74ms/step - loss: 0.4429 - sparse_categorical_accuracy: 0.8455
Epoch 13/15
1563/1563 [==============================] - 119s 76ms/step - loss: 0.4242 - sparse_categorical_accuracy: 0.8517
Epoch 14/15
1563/1563 [==============================] - 128s 82ms/step - loss: 0.3985 - sparse_categorical_accuracy: 0.8583
Epoch 15/15
1563/1563 [==============================] - 127s 81ms/step - loss: 0.3828 - sparse_categorical_accuracy: 0.8629

<tensorflow.python.keras.callbacks.History at 0x12e3e882640>
```

#Accuracy and loss

```
test_loss, test_accuracy = cnn_model.evaluate(x_test, y_test)

print("Test accuracy: {}".format(test_accuracy))
```

```
313/313 [==============================] - 5s 16ms/step - loss: 0.6533 - sparse_categorical_accuracy: 0.7946
Test accuracy: 0.7946000099182129
```

For full code https://github.com/anandprems/cnn

EndNote:

Did you find this article helpful? Please share your opinions/thoughts in the comments section below. Learn from mistakes is my favourite quote, if you found anything wrong too, just highlight it, I am ready to learn from the learners like you people.

About me in short, I am Premanand.S, Assistant Professor Jr and a researcher in Machine Learning. Love to teach and love to learn new things in Data Science. Mail me for any doubt or mistake, er.anandprem@gmail.com, and my Linkedin https://www.linkedin.com/in/premsanand/

FAQs

The media shown in this article are not owned by Analytics Vidhya and are used at the Author's discretion.

Blogathon CNN Deep Learning

Premanand S

Premanand S is a dedicated academic with over a decade of research experience, specializing in Bio-signal Processing, Machine Learning, and Deep Learning. He completed his B.Tech in 2009 from Amrita Vishwa Vidyapeetham, Bangalore, and his M.E. in 2011 from Rajalakshmi Engineering College, Chennai, where his thesis focused on Deep Learning for ECG Signal Processing.

He is pursuing his Ph.D. at VIT-Chennai, with a tentative research title of "Deep Learning Approaches for Enhanced ECG Signal Processing and Arrhythmia Classification." His research aims to leverage cutting-edge deep learning techniques to improve the accuracy and efficiency of ECG signal analysis, contributing significantly to cardiac health monitoring.

A recipient of the prestigious TCS-RSP (Research Scholarship) in 2014, Cycle 9, Premanand has become a recognized figure in the academic community. He has delivered several invited talks on Data Science, Machine Learning, and Deep Learning at prominent institutions across India.

In his role as an Assistant Professor at VIT-Chennai, he continues to inspire the next generation of researchers while advancing the boundaries of knowledge in his field.

Beginner | Computer Vision | Deep Learning | Python

Free Courses

Generative AI - A Way of Life

Explore Generative AI for beginners: create text and images, use top AI tools, learn practical skills, and ethics.

Getting Started with Large Language Models

Master Large Language Models (LLMs) with this course, offering clear guidance in NLP and model training made simple.

Building LLM Applications using Prompt Engineering

This free course guides you on building LLM apps, mastering prompt engineering, and developing chatbots with enterprise data.

Improving Real World RAG Systems: Key Challenges & Practical Solutions

Explore practical solutions, advanced retrieval strategies, and agentic RAG systems to improve context, relevance, and accuracy in AI-driven applications.

Microsoft Excel: Formulas & Functions

Master MS Excel for data analysis with key formulas, functions, and LookUp tools in this comprehensive course.

Responses From Readers

What are your thoughts?...

Submit reply

Frequently Asked Questions

Q1. How many layers are in CNN?

CNNs typically include three main layers:
Convolutional layers: Detect patterns in input data.
Pooling layers: Reduce feature map size and model complexity.
Fully connected layers: Classify extracted features.

Q2. Which algorithm is CNN?

- Build Your Brand & Audience

- Cash In on Your Knowledge
- Join a Thriving Community
- Level Up Your Data Science Game

Flagship Courses

GenAI Pinnacle Program | AI/ML BlackBelt Courses

Free Courses

Generative AI | Large Language Models | Building LLM Applications using Prompt Engineering | Building Your first RAG System using LlamaIndex | Stability.AI | MidJourney | Building Production Ready RAG systems using LlamaIndex | Building LLMs for Code | Deep Learning | Python | Microsoft

Excel | Machine Learning | Decision Trees | Pandas for Data Analysis | Ensemble Learning | NLP | NLP using Deep Learning | Neural Networks | Loan

RECOMMENDED ARTICLES

Introduction to Convolutional Neural Networks (...

Introduction to Convolutional Neural Networks i...

Beginners Guide to Convolutional Neural Network...

CONVOLUTIONAL NEURAL NETWORK(CNN)

Basics of CNN in Deep Learning

Image Processing using CNN: A beginners guide

20 Questions to Test your Skills on CNN (Convol...

Building a Convolutional Neural Network Using T...

Convolutional Neural Networks : Understand the ...

What is the Convolutional Neural Network Archit...

Mistral-7b | Gemini 1.5 Pro | Gemini Flash 1.5 | Bedrock | Vertex AI | DALL.E | Midjourney | Stable Diffusion

Data Science Tools and Techniques

Python | R | SQL | Jupyter Notebooks | TensorFlow | Scikit-learn | PyTorch | Tableau | Apache Spark | Matplotlib | Seaborn | Pandas | Hadoop | Docker | Git | Keras | Apache Kafka | AWS | NLP | Random Forest | Computer Vision | Data Visualization | Data Exploration | Big Data | Common Machine Learning Algorithms | Machine Learning