**ABSTRACT**: Automatic defects detection in MR images is very important in many diagnostic and therapeutic applications. Because of high quantity data in MR images and blurred boundaries, tumor segmentation and classification is very hard. This work has introduced one automatic brain tumor detection method to increase the accuracy and yield and decrease the diagnosis time. The goal is classifying the brain MR images into two classes normal and abnormal. MR images that have been used here are MR images from normal and abnormal brain tissues. Here, it is tried to give clear description from brain tissues using Convolutional Neural Network. This method uses from neural network to do this classification. The purpose of this project is to classify the brain tissues to normal and abnormal classes automatically, that saves the radiologist time, increases accuracy and yield of diagnosis**.**

**KEYWORS:** MRI Images, CNN-convolutional Neural Network.

Chapter-1

Introduction

Brain tumor is any mass that results from an abnormal and an uncontrolled growth of cells in the brain. Its threat level depends on a combination of factors like the type of tumor, its location, its size and its state of development. Brain

tumors can be

1. cancerous (malignant)
2. non-cancerous (benign)

Benign brain tumors are low grade, non-cancerous brain tumors, which, grow slowly and push aside normal tissue but do not invade the surrounding normal tissue. They are homogeneous, well defined and are known as non- metastatic tumors, because they do not form any secondary tumor. Whereas, malignant brain tumors are cancerous brain tumors, which grow rapidly and invade the surrounding

normal tissue. Malignant brain tumors or cancerous brain tumors can be counted among the most deadly diseases [1].

Brain magnetic resonance imaging (MRI) is one of the best imaging techniques that researchers relied on for detecting the brain tumors and modeling of the tumor progression in both the detection and the treatment phases. MRI images have a big impact in the automatic medical image analysis field for its ability to provide a lot of information about the brain structure and abnormalities within the brain tissues due to the high resolution of the images.

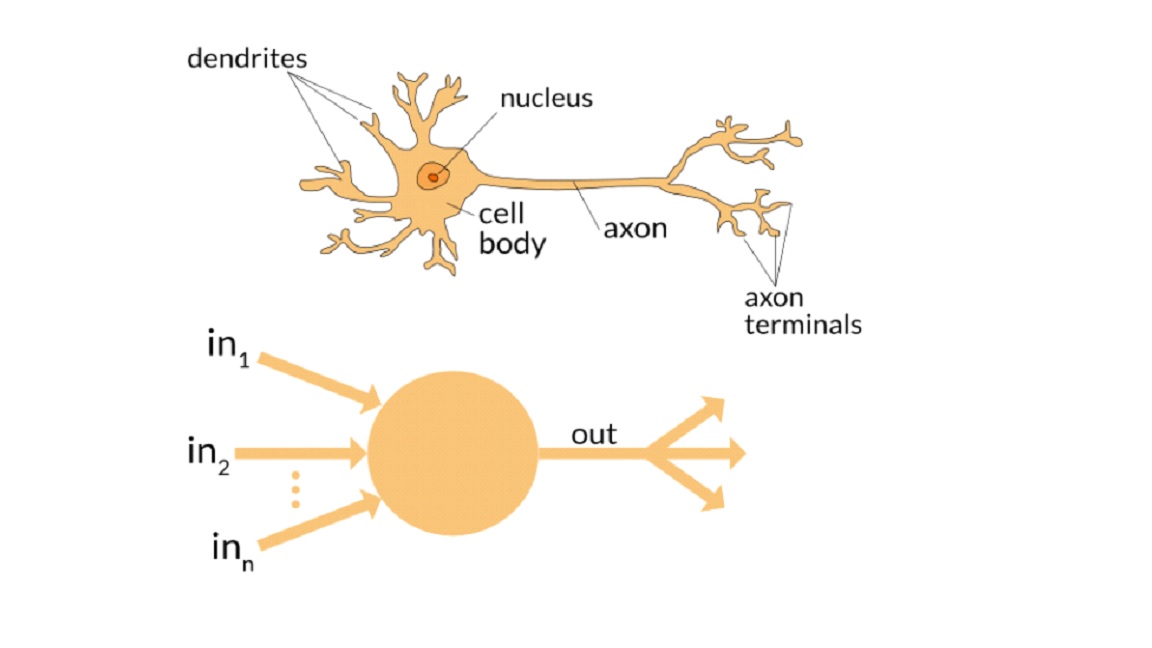
* 1. Machine Learning

Machine learning is a field of computer science that gives computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without

being explicitly programmed Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:

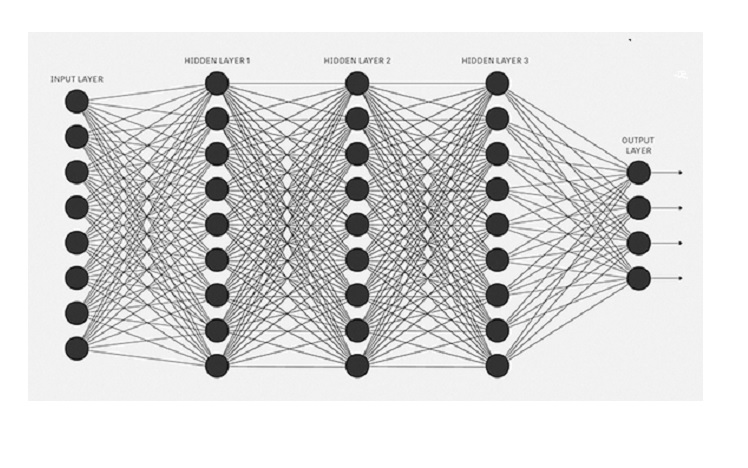
* Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. As special cases, the input signal can be only partially available, or restricted to special feedback
* Semi-supervised learning: the computer is given only an incomplete training signal: a training set with some (often many) of the target outputs missing.
* Active learning: the computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labeling.
* Reinforcement learning: training data (in form of rewards and punishments) is given only as feedback to the program’s actions in a dynamic environment, such as driving a vehicle or playing a game against an opponent.
* Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning)
  1. Artificial Neuron

An artificial neuron is a mathematical function conceived as a model of biological neurons, a neural network.



* 1. Neural Network

Artificial neural networks (ANNs) are software implementations of the neuronal structure of our brains. Artificial neural networks attempt to simplify and mimic this brain behaviour. They can be trained in a supervised or unsupervised manner. In a supervised ANN, the network is trained by providing matched input and output data samples, with the intention of getting the ANN to provide a desired output for a given input.



* 1. CNN(convolutional Neural Network)

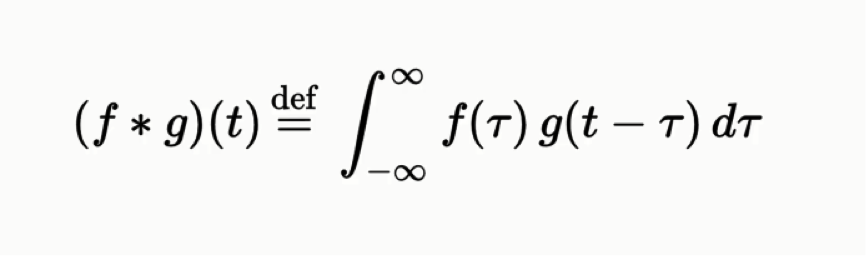
CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs.

CNNs derive their name from the “convolution” operator. The primary purpose of Convolution in case of a CNNs is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

**STEP 1 – CONVOLUTION OPERATION**

**WHAT IS CONVOLUTION?**

In purely mathematical terms, convolution is a function derived from two given functions by integration which expresses how the shape of one is modified by the other. That can sound baffling as it is, but to make matters worse, we can take a look at the convolution formula:

**[](http://www.superdatascience.com/wp-content/uploads/2018/08/Convolutional_Neural_Networks_CNN_Step1_Img1.png)**

Let's get into the actual convolution operation in the context of neural networks. The following example will provide you with a breakdown of everything you need to know about this process.

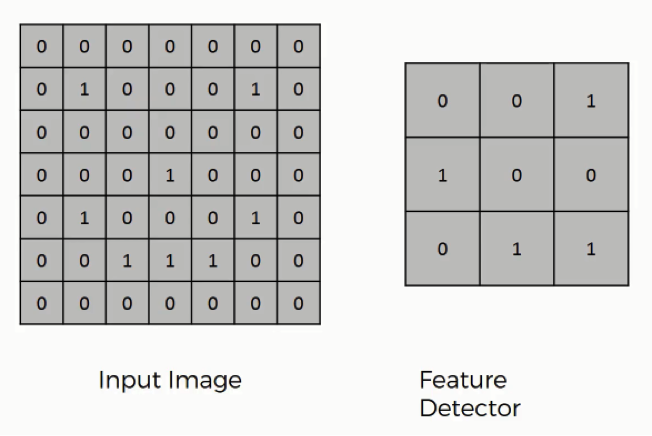
## ****THE CONVOLUTION OPERATION****

Here are the three elements that enter into the convolution operation:

* Input image
* Feature detector
* Feature map

If you look into the pattern of the 1's and 0's, you will be able to make out the smiley face in there.

Sometimes a 5×5 or a 7×7 matrix is used as a feature detector, but the more conventional one, and that is the one that we will be working with, is a 3×3 matrix. The feature detector is often referred to as a “kernel” or a “filter,”

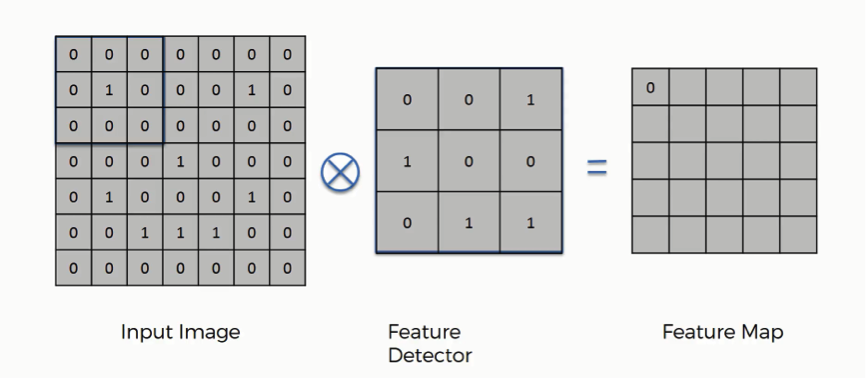


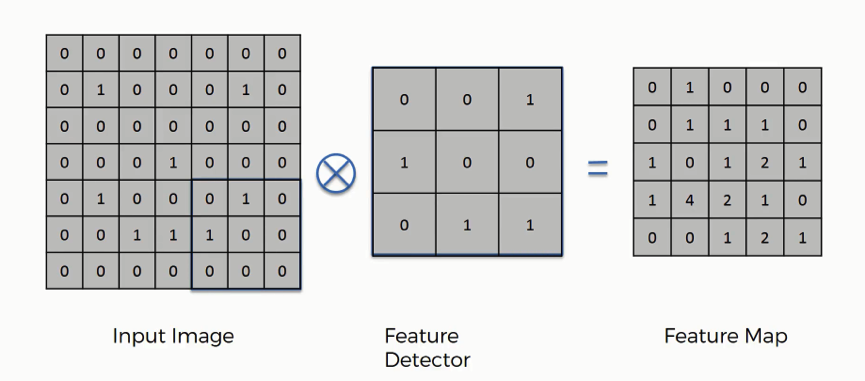
**HOW EXACTLY DOES THE CONVOLUTION OPERATION WORK?**

You can think of the feature detector as a window consisting of 9 (3×3) cells. Here is what you do with it:

* You place it over the input image beginning from the top-left corner within the borders you see demarcated above, and then you count the number of cells in which the feature detector matches the input image.
* The number of matching cells is then inserted in the top-left cell of the feature map.
* You then move the feature detector one cell to the right and do the same thing. This movement is called a and since we are moving the feature detector one cell at time, that would be called a stride of one pixel.
* What you will find in this example is that the feature detector's middle-left cell with the number 1 inside it matches the cell that it is standing over inside the input image. That's the only matching cell, and so you write “1” in the next cell in the feature map, and so on and so forth.
* After you have gone through the whole first row, you can then move it over to the next row and go through the same process.

It's important not to confuse the feature map with the other two elements. The cells of the feature map can contain any digit, not only 1's and 0's. After going over every pixel in the input image in the example above, we would end up with these results





By the way, just like feature detector can also be referred to as a kernel or a filter, a feature map is also known as an activation map and both terms are also interchangeable.

## ****WHAT IS THE POINT FROM THE CONVOLUTION OPERATION?****

There are several uses that we gain from deriving a feature map. These are the most important of them:

Reducing the size of the input image, and you should know that the larger your strides (the movements across pixels), the smaller your feature map.

In this example, we used one-pixel strides which gave us a fairly large feature map.

When dealing with proper images, you will find it necessary to widen your strides. Here we were dealing with a 7×7 input image after all, but real images tend to be substantially larger and more complex.

That way you will make them easier to read.

## ****DO WE LOSE INFORMATION WHEN USING A FEATURE DETECTOR?****

The answer is **YES**.

The feature map that we end up with has fewer cells and therefore less information than the original input image. However, the very purpose of the feature detector is to sift through the information in the input image and filter the parts that are integral to it and exclude the rest.

Basically, it is meant to separate the wheat from the chaff.

## ****WHY DO WE AIM TO REDUCE THE INPUT IMAGE TO ITS ESSENTIAL FEATURES?****

Think of it this way.

What you do is detect certain features, say, their eyes and their nose, for instance, and you immediately know who you are looking at.

These are the most revealing features, and that is all your brain needs to see in order to make its conclusion. Even these features are seen broadly and not down to their minutiae.

If your brain actually had to process every bit of data that enters through your senses at any given moment, you would first be unable to take any actions, and soon you would have a mental breakdown. Broad categorization happens to be more practical.

Convolutional neural networks operate in exactly the same way.

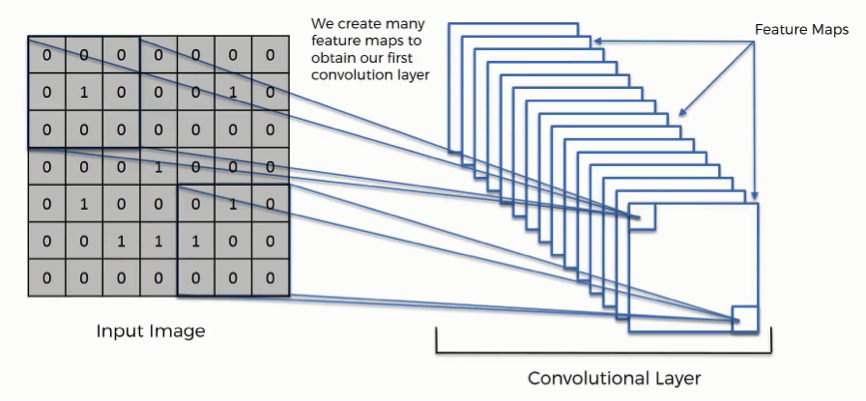
## ****HOW DO CONVOLUTIONAL NEURAL NETWORKS ACTUALLY PERFORM THIS OPERATION?****

The example we gave above is a very simplified one, though.

In reality, convolutional neural networks develop multiple feature detectors and use them to develop several feature maps which are referred to as convolutional layers (see the figure below).

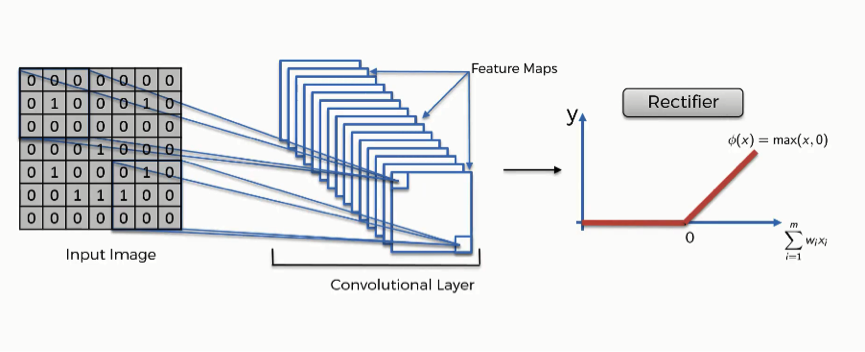
Through training, the network determines what features it finds important in order for it to be able to scan images and categorize them more accurately.

Based on that, it develops its feature detectors. In many cases, the features considered by the network will be unnoticeable to the human eye, which is exactly why convolutional neural networks are so amazingly useful. With enough training, they can go light years ahead of us in terms of image processing.



# ****STEP 1(B): THE RECTIFIED LINEAR UNIT (RELU)****

The Rectified Linear Unit, or ReLU, is not a separate component of the convolutional neural networks' process.



The purpose of applying the rectifier function is to increase the non-linearity in our images.

The reason we want to do that is that images are naturally non-linear.

When you look at any image, you'll find it contains a lot of non-linear features (e.g. the transition between pixels, the borders, the colors, etc.).

The rectifier serves to break up the linearity even further in order to make up for the linearity that we might impose an image when we put it through the convolution operation.

# ****STEP 2 – MAX POOLING****

## ****WHAT IS MAX POOLING?****

The purpose of max pooling is enabling the convolutional neural network to detect the required pattern when presented with the image in any manner.

Max pooling is concerned with teaching your convolutional neural network to recognize that despite **all of the differences like viewing angle, lightening condition, etc.**

**In order to do that, the network needs to acquire a property that is known as “spatial variance.”**

This property makes the network capable of detecting the object in the image without being confused by the differences in the image's textures, the distances from where they are shot, their angles, or otherwise.

In order to reach the pooling step, we need to have finished the convolution step, which means that we would have a feature map ready.

## ****TYPES OF POOLING****

There are several types of pooling. These include among others the following:

* Mean pooling
* Max pooling
* Sum pooling

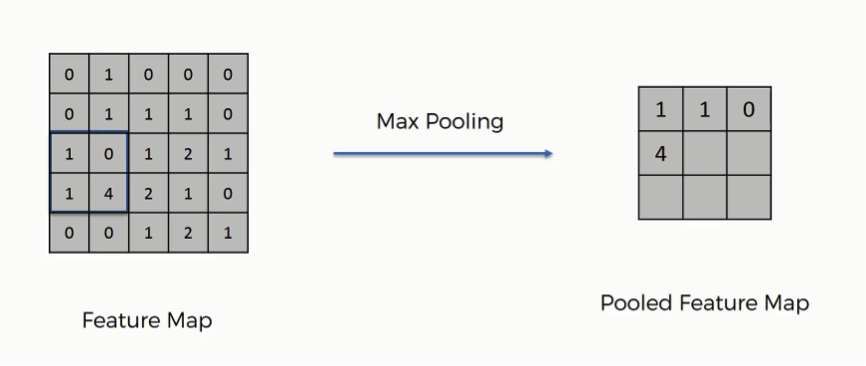
Our main focus here will be max pooling.

## ****POOLED FEATURE MAP****

The process of filling in a pooled feature map differs from the one we used to come up with the regular feature map.

This time you'll place a 2×2 box at the top-left corner, and move along the row.

For every 4 cells your box stands on, you'll find the maximum numerical value and insert it into the pooled feature map. In the figure below, for instance, the box currently contains a group of cells where the maximum value is 4.

**[](http://www.superdatascience.com/wp-content/uploads/2018/08/Convolutional_Neural_Networks_CNN_Step2_Img3.png)**

In this example, we are using 2-pixel strides.

That's why we end up with a 3×3 pooled featured map. Generally, strides of two are most commonly used.

Note that in the third movement along the same row, you will find yourself stuck with one lonely column.

You would still proceed despite the fact that half of your box will be empty. You still find your maximum value and put it in the pooled feature map. In the least step, you will face a situation where the box will contain a single cell. You will take that value to be the maximum value.

Just like in the convolution step, the creation of the pooled feature map also makes us dispose of unnecessary information or features. In this case, we have lost roughly 75% of the original information found in the feature map since for each 4 pixels in the feature map we ended up with only the maximum value and got rid of the other 3. These are the details that are unnecessary and without which the network can do its job more efficiently.

The reason we extract the maximum value, which is actually the point from the whole pooling step, is to account for distortions. Let's say we have three cheetah images, and in each image the cheetah's tear lines are taking a different angle.

The feature after it has been pooled will be detected by the network despite these differences in its appearance between the three images. Consider the tear line feature to be represented by the 4 in the feature map above.

Imagine that instead of the four appearing in cell 4×2, it appeared in 3×1. When pooling the feature, we would still end up with 4 as the maximum value from that group, and thus we would get the same result in the pooled version.

This process is what provides the convolutional neural network with the “spatial variance” capability. In addition to that, pooling serves to minimize the size of the images as well as the number of parameters which, in turn, prevents an issue of “overfitting” from coming up.

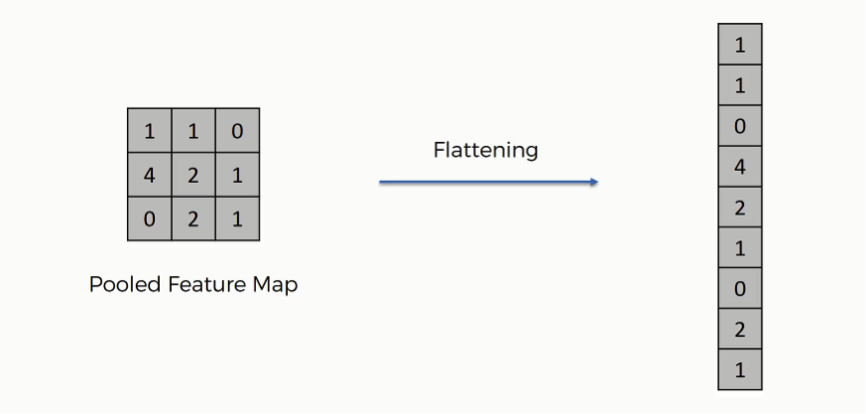
Overfitting in a nutshell is when you create an excessively complex model in order to account for the idiosyncracies we just mentioned.

Again, this is an abstract explanation of the pooling concept without digging into the mathematical and technical aspects of it.

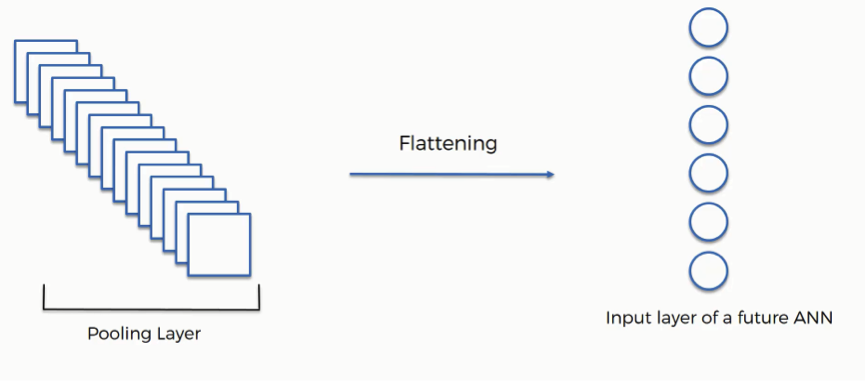
We can draw an analogy here from the human brain. Our brains, too, conduct a pooling step, since the input image is received through your eyes, but then it is distilled multiple times until, as much as possible, only the most relevant information is preserved for you to be able to recognize what you are looking at.

# ****STEP 3: FLATTENING****

After finishing the previous two steps, we're supposed to have a pooled feature map by now. As the name of this step implies, we are literally going to flatten our pooled feature map into a column like in the image below.

**[](http://www.superdatascience.com/wp-content/uploads/2018/08/Convolutional_Neural_Networks_CNN_Step3_Img1.png)**

The reason we do this is that we're going to need to insert this data into an artificial neural network later on.

**[](http://www.superdatascience.com/wp-content/uploads/2018/08/Convolutional_Neural_Networks_CNN_Step3_Img2.png)**

As you see in the image above, we have multiple pooled feature maps from the previous step.

What happens after the flattening step is that you end up with a long vector of input data that you then pass through the artificial neural network to have it processed further.

# ****STEP 4: FULL CONNECTION****

Here's where artificial neural networks and convolutional neural networks collide as we add the former to our latter.

It's here that the process of creating a convolutional neural network begins to take a more complex and sophisticated turn.

As you see from the image below, we have three layers in the full connection step:

* Input layer
* Fully-connected layer
* Output layer

**[](http://www.superdatascience.com/wp-content/uploads/2018/08/Convolutional_Neural_Networks_CNN_Step4_Img1.png)**

Notice that when we discussed artificial neural networks, we called the layer in the middle a “hidden layer” whereas in the convolutional context we are using the term “fully-connected layer.”

## ****THE FULL CONNECTION PROCESS****

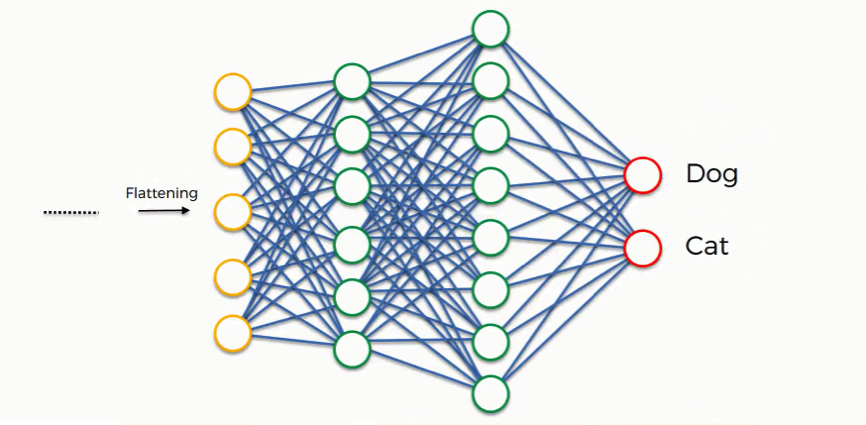
The input layer contains the vector of data that was created in the flattening step. The features that we distilled throughout the previous steps are encoded in this vector.

### WHAT IS THE AIM OF THIS STEP?

The role of the artificial neural network is to take this data and combine the features into a wider variety of attributes that make the convolutional network more capable of classifying images, which is the whole purpose from creating a convolutional neural network.

We can now look at a more complex example than the one at the beginning of the tutorial.

We'll explore how the information is processed from the moment it is inserted into the artificial neural network and until it develops its classes.

**[](http://www.superdatascience.com/wp-content/uploads/2018/08/Convolutional_Neural_Networks_CNN_Step4_Img2.png)**

At the very beginning, as you know by now, we have an input image which we convolve, pool, flatten, and then pass through the artificial neural network.

By the end of this channel, the neural network issues its predictions.

Say, for instance, the network predicts the figure in the image to be a malignant by a probability of 80%, yet the image actually turns out to be of a benign.

An error has to be calculated in this case.

In the context of artificial neural networks, we call this calculation a “cost function” or a mean squared error, but as we deal with convolutional neural networks, it is more commonly referred to as a “loss function.”

We use the cross-entropy function in order to achieve that.

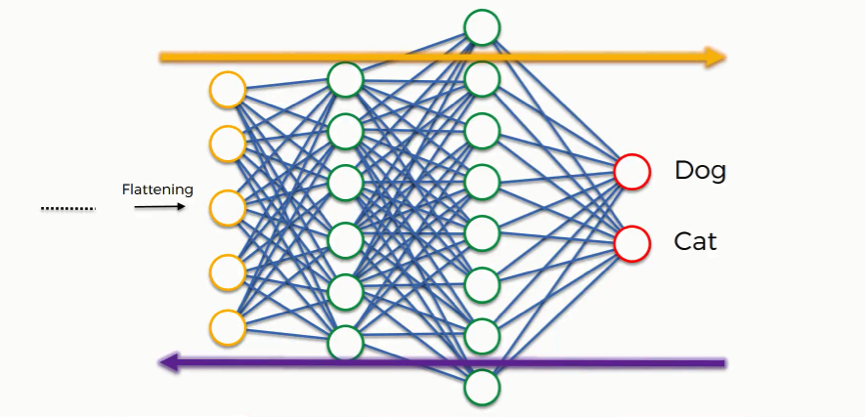
Loss function informs us of how accurate our network is, which we then use in optimizing our network in order to increase its effectiveness.

That requires certain things to be altered in our network.

These include the weights (the blue lines connecting the neurons, which are basically the synapses), and the feature detector since the network often turns out to be looking for the wrong features and has to be reviewed multiple times for the sake of optimization.

Just as we said when discussing artificial neural networks, the information is then conveyed in the opposite direction as you see in the figure below. As we work to optimize the network, the information keeps flowing back and forth over and over until the network reaches the desired state.

## ****CLASS RECOGNITION****

**[](http://www.superdatascience.com/wp-content/uploads/2018/08/Convolutional_Neural_Networks_CNN_Step4_Img3.png)**

Benign

Malignant

Up until now, we've been discussing examples where the output consists of a single neuron.

Since this one contains two, there are some differences that show up.

Let's first look at the “Benign” class.

In order to understand how it will play out, we need to check out the weights placed on each synapse linking to this class so that we can tell which attributes/features are most relevant to it.

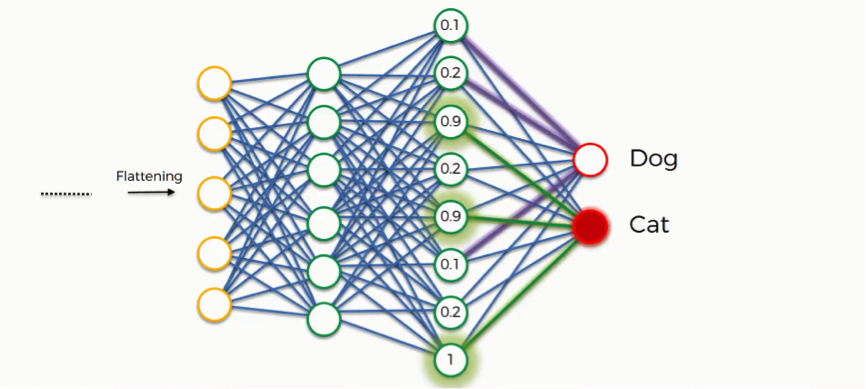
This full connection process practically works as follows:

* The neuron in the fully-connected layer detects a certain feature; say, a tumor cell.
* It preserves its value.
* It communicates this value to both the “Benign” and the “Malignant” classes.
* Both classes check out the feature and decide whether it's relevant to them.

In our example, the weight placed on the tumor cell-Benign synapse is high (1.0), which means that the network is confident that this is a Benign Tumor.

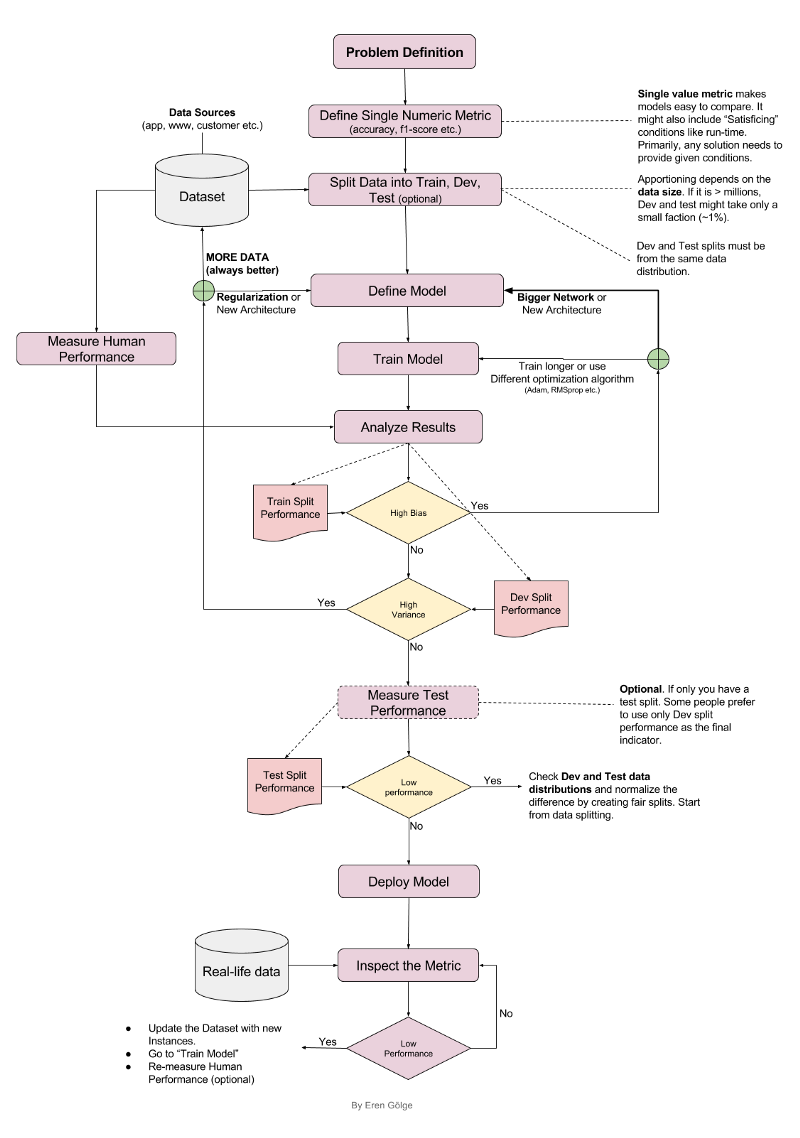
Since the information is constantly flowing in both directions, the “Malignant” class takes note of this and understands that since this is a Benign Tumor, then it simply can't be a Malignant Tumor. Even if at first it would have considered the signal saying “small rounded swollen” because this might be a malignant as well as a Benign Tumor, now it dismisses this feature.

This happens gradually as it receives the same reading multiple times. The Benign class on its part will start focusing more on the attributes carrying the highest weight (the three thick purple lines), and it will ignore the rest.

**[](http://www.superdatascience.com/wp-content/uploads/2018/08/Convolutional_Neural_Networks_CNN_Step4_Img4.png)**The same process simultaneously occurs with the Malignant, enabling it to pick out its own priority features. What we end up with is what you see in the image below. As this process goes on repeat for thousands of times, you find yourself with an optimized neural network.

Malignant

Benign



Problem Definition

Define single Numeric Metric

Split data into Train and Test

Dataset

Define the Model i.e. CNN

Training Dataset

Train the CNN

Analyse the results

NO

NO

Good Training Accuracy

YES

Measure test accuracy

Testing Dataset

Good Performance

YES

Deploy the model

Real Time Data

Doing predictions for real time data

**REFERENCES**

1. P. John, “Brain Tumor Classification Using Wavelet and Texture Based Neural Network”,International Journal of Scientific & Engineering

Research, Vol. 3, Issue 10, pp. 1-7, 2012.