

C:\Users\Rajat\Desktop\line.pngUTTAR PRADESHC:\Users\Rajat\Desktop\line.png

In-House Practical Training Project Report on

**Image Recognition Model using Python**

Submitted to:

Amity University, Uttar Pradesh



by

**Dhananjai Sharma**

**(A2305216430)**

Under the guidance of

**MS. ABHA THAKRAL**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY**

**AMITY UNIVERSITY UTTAR PRADESH**

**NOIDA (U.P.)**

**(2018-2019)**

**DECLARATION**

I, Dhananjai Sharma, student of B.Tech Computer Science and Engineering, hereby declare that the project report entitled “Image Recognition Model using python” which is submitted by me to the Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh in partial fulfillment of requirement for award of the degree of Bachelor of Technology in Computer Science and Engineering, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

Place: Noida --------------

Date \_\_\_\_\_\_\_\_\_\_\_\_ **Dhananjai Sharma**

(A2305216430)

**CERTIFICATE**

On the basis of the declaration submitted by Dhananjai Sharma (Enrolment No.: A2305216430), student of B.Tech Computer Science and Engineering, I hereby certify that the project report entitled “Image Recognition Model using Python”, which is submitted to the Department of Computer Science, Amity School of Engineering and Technology, Amity University, Noida, Uttar Pradesh in partial fulfillment of requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is an original contribution with existing knowledge and faithful record of work carried out by him under my guidance and supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

………………………

**Ms. Abha Thakral**

Assistant Professor

Department of Computer Science

Amity School of Engineering and Technology

Amity University, Uttar Pradesh

Date……………......

**ACKNOWLEDGEMENT**

A project cannot be completed without the immense help and motivation given by the faculties and the mentors. I would like to thank Prof (Dr) Abhay Bansal, Professor and Head of department of Computer Science and Engineering, Joint Head, ASET and Director, DICET, Amity University for giving me the opportunity to undertake this project. A vote of special thanks goes to Ms. Abha Thakral, Assistant Professor, CSE department, who has been a constant guiding force throughout the course of this project, and whose immense support and cooperation has made the completion of this project possible. I thank ma’am for her patience and kindness in working with me for this project and for motivation she has given throughput.

Last but not the least, I would like to thank my family and other loved ones, who supported me through my shakes and shudders, both by keeping me harmonious and by helping me put the pieces together. I will be forever in debt of your support.

Dhananjai Sharma

(A2305216430)

**Abstract**

Over the last decade, Digital Image Processing (DIP) has been used in many areas from computerized photography to face/iris recognition. With the advent of recent advances in technology, feature extraction and pattern recognition are being widely used in fields such as forensic and medical analysis. One major area of Digital Image Processing (DIP) is image recognition. Image recognition is used in many areas such as in self driving cars and surveillance cameras to name a few. In this project an image recognition model is created in python by making use of a library called keras (a minimalistic python deep learning library).

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Title** | ­**Page No.** |
| **1.** | Introduction | 7 |
| **2.** | Materials and Methodology   * Software Requirements * Image Classification Problem * What is Deep Learning? * Neural Networks – Artificial Neural Networks * Activation Function * Backpropagation * Gradient Descent and Stochastic Gradient Descent * What are Convolutional Neural Networks * Convolution * Max Pooling * Flattening * Full Connection | 9 |
| **3.** | **Results and Discussion** | 24 |
| **4.** | **Conclusion and Recommendation** | 26 |
| **5.** | **Implication for Future Research** | 27 |
| **6.** | **References** | 28 |

**1. Introduction**

In machine learning typically, we would start with a bunch of features that is extracted from the data and then build models on top of it. In supervised learning, we have labels associated with it. In real world application, that extraction of features can become very challenging. For instance, in the case of autonomous driving, different conditions during the driving patterns people have, the weather patterns, road conditions, illuminations, local roads versus highways, make the feature extraction part challenging. When machine learning is applied to health care, it is a very attractive domain but features may not be easily discoverable. In today's day and age, it's not possible to live without web search engines. And these web search engines are increasingly becoming sophisticated and involve machine translation and document comprehension. Machine translation pops up in other places as well. For instance, in Skype Translator when you are talking, we can translate one language from the other. Speech is an area where different intonations, environment, noise, languages make feature detection very, very challenging. As children we all learn to build primitive constructs to recognize voice, and slowly, higher level abstract constructs help our brain understand language as we grow. Many speech recognition engines mimic such complex abstraction models in delivering the experience we have in the form of different applications and devices. Deep learning solves this problem of learning higher order abstractions. Increasingly, these types of models, because of the higher level abstractions that they are able to build, are becoming pervasive in our day-to-day living. Deep learning enables building complex or higher-level constructs, using simpler constructs. Starting with the raw data, simple constructs, to a range of higher order constructs or abstractions. Deep learning is powered by deep neural networks, also referred to as DNN. Deep learning embodies many of the constructs that is inherently built into our biology, particularly the brain. Deep neural networks may make several layers in the brain. Similarly, deep neural nets in a computer have multiple layers. Each layer learns a higher abstraction on the input from the layer before it. And many practical deep neural nets have large number of parameters. These types of deep networks, which have humongous number of parameters, were previously difficult to build, largely because of lack of large amount of data to train these models, as well as the computing capabilities needed to process that amount of data and build models out of it. With recent advances in computer science, such cross computing power and handling of large data is made feasible. And with increasing number of devices generating data, the amount of data is also becoming abundant, helping us build really complex models that mimic natural behaviours. So that autonomous driving, document comprehension and speech recognition tasks are becoming more and more common. Application domains for deep learning thus are in image and video processing, speech processing, text. And combining these three, Increasing multi-modality and data coming from Internet of Things, is becoming increasingly common.

**2. Materials and Methodology**

**2.1 Software Requirements**

**2.1.1 Python (2.7 or above)**

The project is completely python based, that is, the project code is written in python and following python libraries have been used:

* Keras
* PIL (Python Imaging Library)
* numpy

**2.1.2 Spyder**

Spyder is an integrated development environment for the Python language.

Features of Spyder:

1. Open source cross-platform IDE.
2. Advance editing, interactive testing and debugging.
3. Variable explorer, function parameter information and IPython Console provided.

**2.2 Image Classification Problem**

The image classification problem can be well understood by the following example:

Consider the following image:



Fig. 1

It is made out of integer values, 28 of them in rows, and then there are 28 columns. Stretch it out and put it in the form of a vector. And the first one becomes the first element of the vector, the second pixel becomes the second element of the vector, and so on and so forth, and the last one goes at the very end. Now, for a 28 by 28 image, when you stretch it out, if you do the arithmetic, you'll find that there are 784 pixels. Take this array of pixels and treat that as our input. And it's represented by this input variable, which we call it as x, and there is an arrow on top of it, which indicates it's a vector. Now, it's no longer a simple number like it was in representing the average day of the temperature. But instead, there's a set of values, 784 of those, representing each pixel in this array of 28 by 28 numbers representing the digit 3. The 28 by 28, by the way, just happens to be the size of the images from this data set. It doesn't have to be 28 by 28. And what we want to do is, instead of directly classifying this to be a value of 3, we would classify that as digit 3. What we do instead is predict what is the chance this image represents the digit 0. 1, 2, 3, 4, to all the way till 9, so on, so forth. Now, you can see that, in this case, this digit is hand-written digit of 3. So we expect that the value corresponding to that digit should be relatively high. So the model is going to output ten different values, and the highest value should be the one that corresponds to that digit.

**2.3 What is Deep Learning?**

Deep Learning is utilized by Google in its voice and picture recognition algorithms, by Netflix and Amazon to choose what you need to watch or purchase straightaway, and by analysts at Massachusetts Institute of Technology (MIT) to foresee what's to come. The consistently developing industry which has set up itself to offer these instruments is constantly quick to discuss how progressive this all is. Be that as it may, what precisely is it? What's more, is it simply one more trend being utilized to push "out-dated" AI on us, under a hot new mark?

Deep Learning is a sub – domain of machine learning. The idea behind deep learning is to mimic the human brain and recreate it inside a machine so that the machine can behave in a way similar to human. The human brain consists of millions of neurons and each neuron is connected with other neurons. Each neuron consists of dendrites (receiver of signals) and axon (transmitter of signal). The axon - dendrite connection where the signal is passed is called the synapse. The neurons and the interconnection between them is represented inside a machine as neural networks.

**2.3 Neural Networks** –Artificial Neural Networks

Each neuron is represented as ‘node’ inside the machine. A node gets a set of input values and after processing those values it gives the output signal. A neural network primarily consists of three types of layer, the input layer, the hidden layer and the output layer. Neural networks having only one node in the hidden layer are known as a perceptron (in fig. 2). Before passing the signal at each synapse, each signal is given a weight. These weights are crucial to the function of neural networks. By adjusting the weights the neural network decides at every single case which signal is important and which is not. So, weights decide to what strength the signals get passed along.

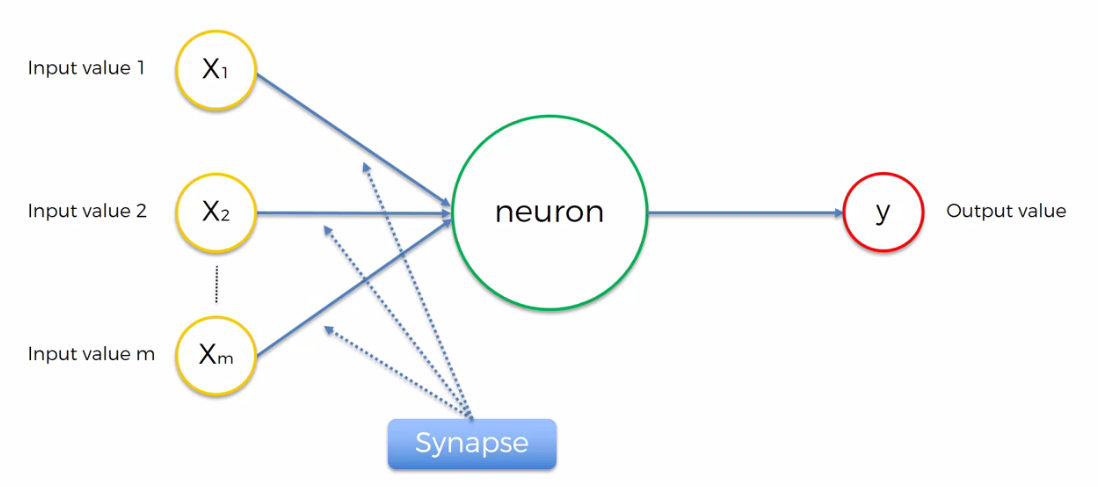


Fig. 2: A Perceptron

**2.4 Activation Function**

When the neuron receives the input signal, it takes weighted sum of all the input values and applies an activation function to it. A visual representation of this step can be seen in figure 3.

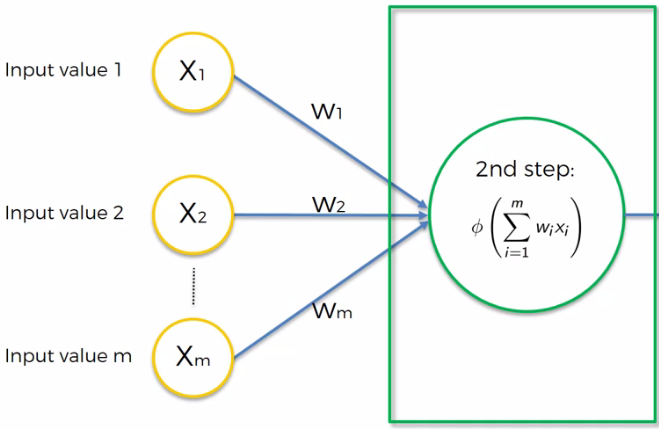


Fig. 3: Activation Function ϕ being applied to the weighted sum of input values

The rectifier function and sigmoid function has been used as activation functions in this project (in the image recognition deep learning model). These functions can be visualized as follows:

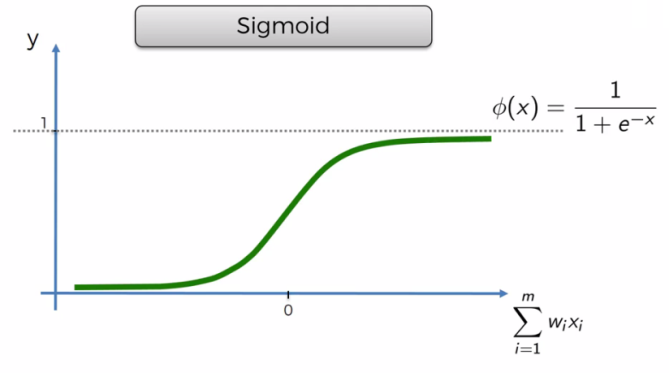
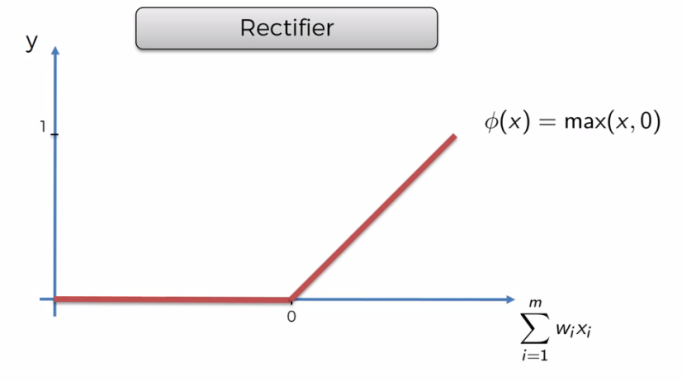


Fig. 4: Rectifier and Sigmoid Functions

**2.5 Backpropagation** –How do neural networks learn?

Each feature of every row/observation is passed in the input layer to different input nodes and the output value is generated based on the default values of the weights. The output value is then compared with the actual value and based on the error (which is determined by a cost function) the weights are updated. Say, we have to predict how much percentage of marks a student is going to score in an exam given the information about his study hours, sleep hours and his internal score.

The neural network for this example with one only hidden layer (and one node) will look something like this:

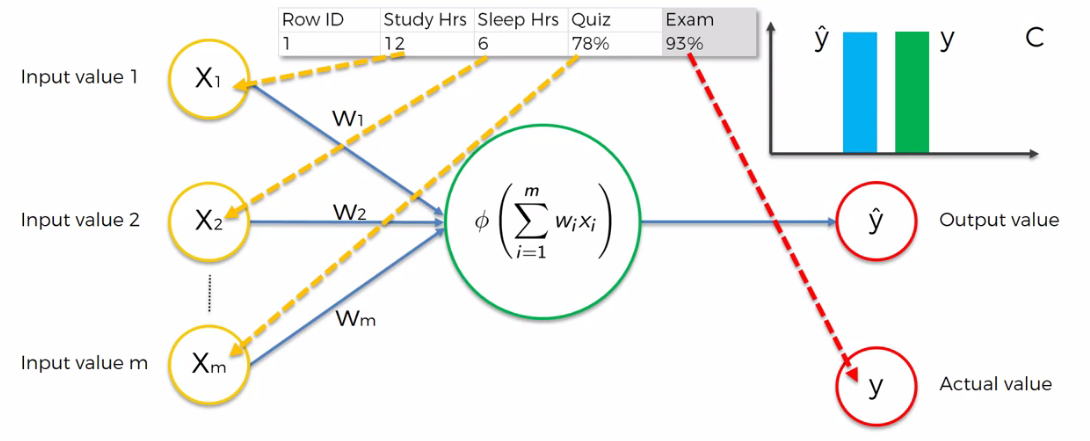


Fig. 5: Each row being served as input

When the study hours, sleep hours and quiz marks are passed into the input layer an activation function is applied to the weighted sum of these inputs and the output value is generated. This output value is compared with the actual value through a cost function. The value of cost function is propagated back to the neuron and the weights are then updated until the point where we get the minimum value of cost function (meaning the output value is nearly equal to the actual value).This process is called backpropagation. In the following figure (fig. 6) the cost function is backpropagated and the value of weights are updated accordingly and the values are again fed to the input layer.

The cost function is denoted as ‘C’:

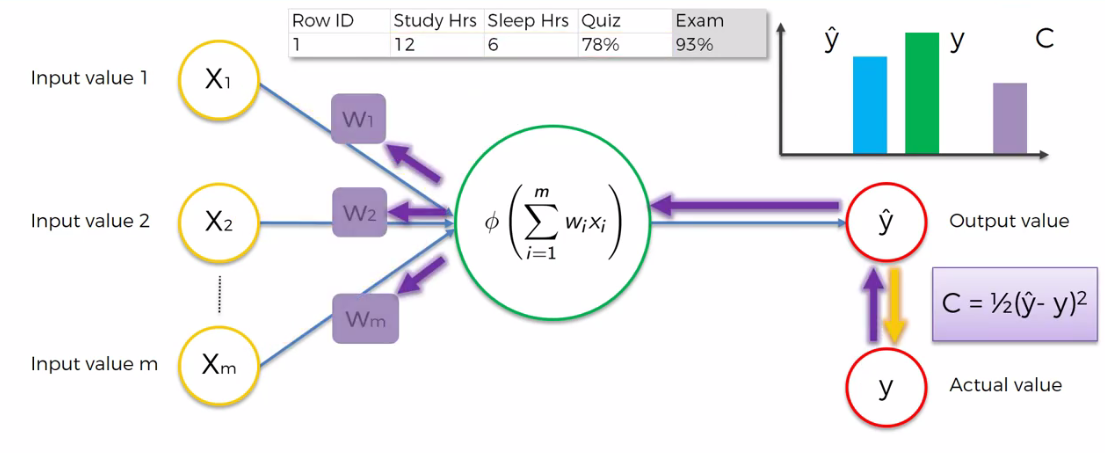


Fig. 6

The same thing can be visualized for more than one row. In this case the sum of cost function for all rows is backpropagated and only then the weights are updated (Fig. 7). The goal here is to minimize the cost function so that the output value and actual value become equal.

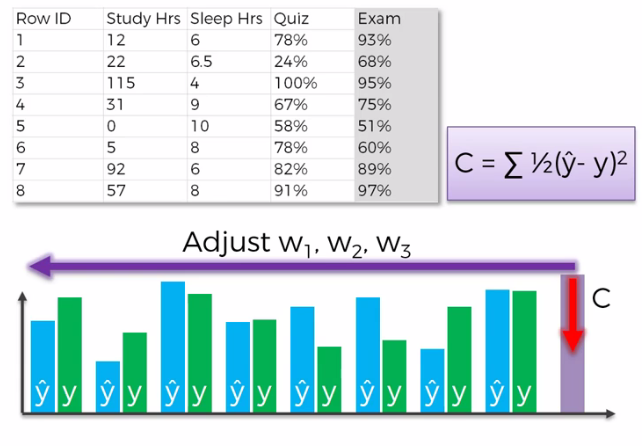


Fig. 7

**2.6 Gradient Descent and Stochastic Gradient Descent**

In updating the weights we cannot use a brute force approach as for even a simple neural network there will be thousands of thousands of possible combination out of which one will be the best set. To solve this optimization problem gradient descent (sometimes also called as batch gradient descent) is introduced. Gradient Descent is an efficient method to solve the optimization problem where we are trying to minimize the cost function by finding the slope at a point and taking a step down(descent) to finding the optimal solution(least value of cost function). A representation of gradient descent can be seen in figure 8.

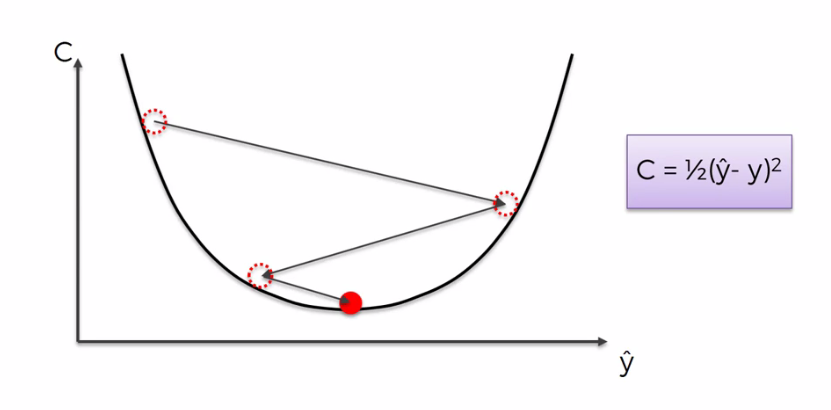


Fig. 8: Gradient Descent in convex function

So what we do with this bowl-like function? We first start with a randomly initialized parameter value. So this would mean that the weights and the corresponding biases would be initialized with some random value. And we want to get to the bottom of the bowl. If you think about the way one would do it is, start marching along the slope of this bowl where it is the steepest. That would lead to the fastest way you can reach this minimum point. Mathematically one will do here is compute the gradient at any given point, so this would correspond to this point on this surface that you see here. We will compute the gradient, the slope at that point and you would step towards the minimum. A step is taken, and the direction of the step would be on the opposite direction where you have the steepest ascent because we want to descend along the curve, and hence the name gradient descent. This value is referred to as the gradient of the loss with respect to that particular parameter. There is a variant of the gradient descent which has vastly changed the way you can scale up the learnings of a very deep model. So there are two variants of the basic Gradient Descent Technique. The first one is called Stochastic Gradient Descent. So in Stochastic Gradient Descent, you update the parameters much more frequently than we would do in regular gradient descent. We do not update it with the whole set either, but we can define, say, at any given time we are going to compute the loss from the first image to the 32nd image. So for a mini batch set of 32, this mini batch size indicates that there are 32 samples of images that you are going to take. And the average loss for each of these images is going to be your loss for that particular iteration. And based on this loss, you will update your theta 1 to the corresponding data two. Consequently what happens is you can start with a point, it will still be wiggly, it won't be as smooth as it was for the Gradient Descent, but it will be somewhat less so. And the fact that you can load a certain set of data points in to the memory rather quickly, the computational load that comes across is very easily parallelizable. I won't go into the details because that's beyond the scope of the course that we have here, but you can read a lot about the learners in the link that I have provided you here. One of the important things, though, to keep in mind is that SGD is not the only learner that is available to you as a programmer and as a person who is training deep learning models. Here are some of the popular learners that you have. Momentum-SGD, Nesterov, Adagrad, Adadelta, and Adam, and you can see here in this animation that they have different kinds of convergence property. Some of them wiggle more then the others, others are much smoother. And at the end of the day all of them find the optimal operating point which in this case is marked by the star this will be our final value of the loss and this will be our model parameter chosen in your final selection of the model.

Gradient descent is only useful when our cost function is convex i.e. having only one global minimum (like in the fig. 8).

So when our cost function is not a convex function it is very much likely that we end up with a sub-par neural network having local minima (as in fig. 9).

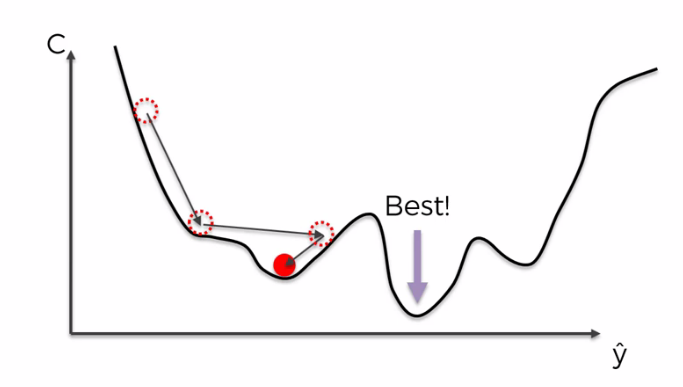
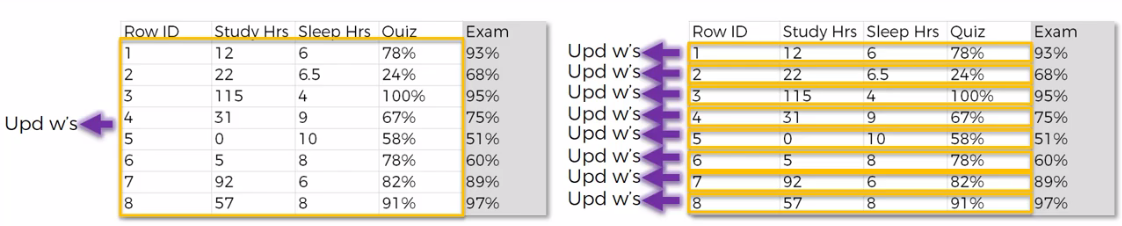


Fig. 9: Gradient Descent in a non-convex function

The solution to this problem is stochastic gradient descent. In stochastic gradient descent method we run the neural network for all the rows but update the weights after the processing of each row and backpropagating the cost function for each output value unlike in gradient descent method where we update all the weights only after all the rows are processed and the sum of cost function is backpropagated. This difference can be visualized as follows:



Batch Gradient Descent Stochastic Gradient Descent

Thus, stochastic gradient descent helps to avoid the problem where we find local minimum rather than the global minimum. The difference between stochastic gradient descent algorithm and gradient descent algorithm is given in table 1.

Table 1. Difference between gradient descent and stochastic gradient descent

|  |  |
| --- | --- |
| **Stochastic Gradient Descent** | **Batch Gradient Descent** |
| It updates weights after processing each row | It updates weights after processing all the rows |
| It is a much lighter and faster algorithm as compared to batch gradient descent algorithm. | It is slower the stochastic gradient descent algorithm as it has to store the results of each row before updating the weights |
| It is a stochastic (having a random probability distribution) algorithm | It is a deterministic algorithm i.e. if we have the same starting weights, same results are obtained at all iterations |

**2.7 What are Convolutional Neural Networks?**

Like artificial neural networks, convolutional neural networks are also made up of neurons that have learnable weight and biases. The only difference is that it takes input in the form of an image and generates an output label for the image.

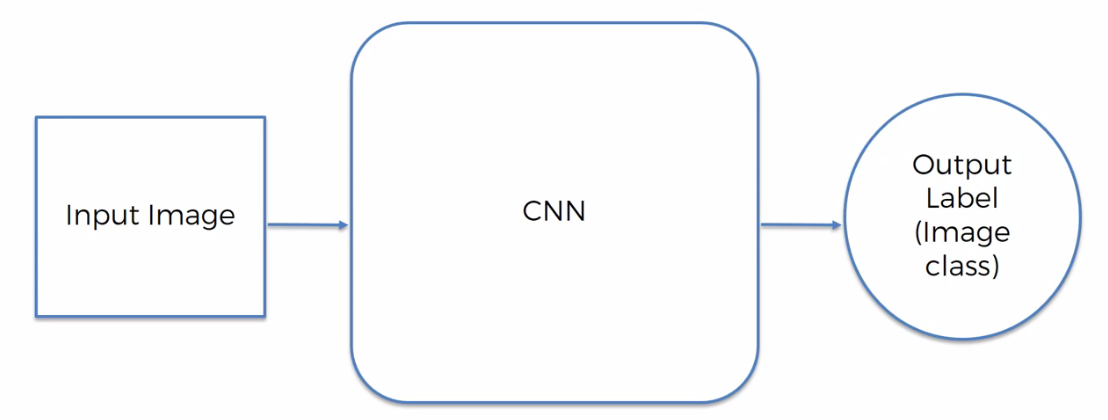


Fig. 10

To take an overview, convolutional neural networks convert input image in the form of an array (2-D array for B/W images and 3-D array for coloured images).

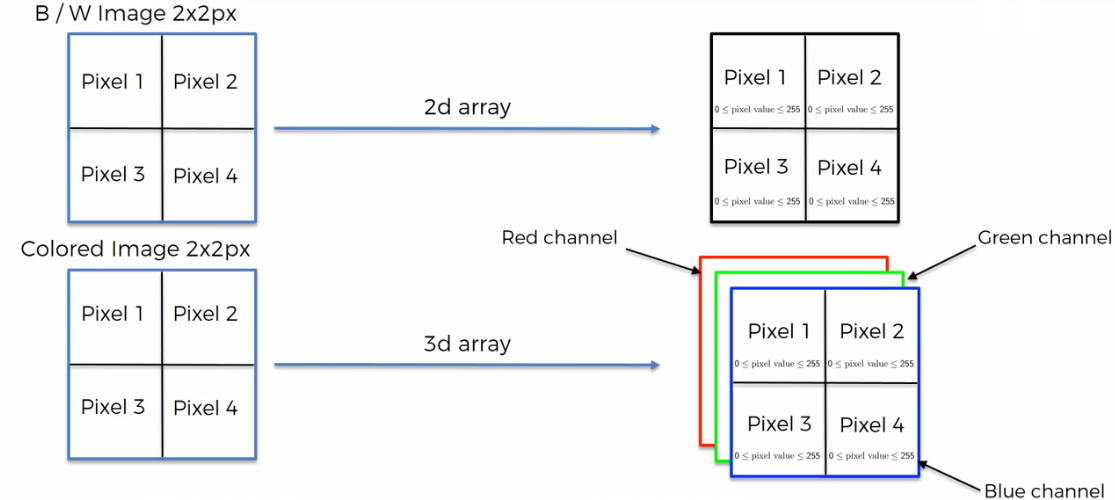


Fig. 11: Image representation inside computer system

The complete process, from taking an image as an input and generating the output label can be divided into four major steps. These steps are as follows:

i) Convolution – ReLu layer

ii) Max Pooling

iii) Flattening

iv) Full Connection

**2.8 Convolution**

In the convolution step, the size of the input is reduced by applying feature detector to the input image to look for only the specific features of the image that are relevant for a particular output class. During its training, the CNN decides which features are important for certain types or certain categories and hence create feature detectors. When feature detector is applied to the input image the size of the image is reduced and only the key features are preserved. Therefore, a collection of feature map is obtained after the input image is filtered by the application of feature detectors. After obtaining the feature maps from the feature detectors non-linearity is added by applying the rectifier function to the feature map in the rectified linear unit step.

The convolution step can be visualized as follows:

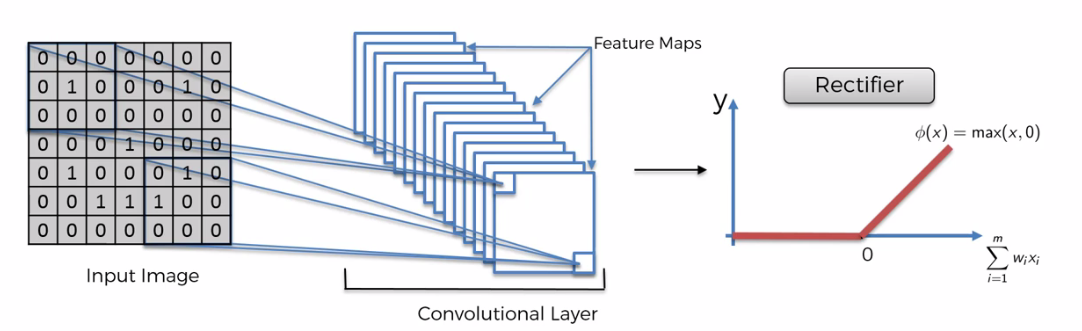


Fig. 12: Convolution step



Fig. 13. Input Image

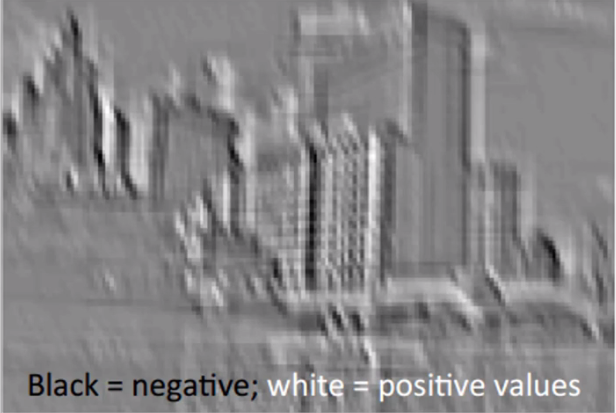


Fig. 14: Feature map (After applying feature detector):



Fig.15: After applying rectifier function

**2.9 Max Pooling –** What is pooling and why do we need it?

One of the key aspect of training these networks is to guard against overfitting because you're dealing with a lot of parameters. Pooling is one of the ways of dealing with it. Pooling adds spatial variance to the image recognition model, meaning that it does not care where the feature is present in the image. Max pooling take the maximum of the feature value from the feature map into the pooled feature map, hence preserving the feature and reducing the size. Pooling is typically inserted between successive convolution layers. And the goal is to reduce the number of parameters and control overfitting. There are many pooling options such as average pooling, max pooling etc. The model created in this project uses max pooling taking a two by two region with a stride of two. What we do here is we take a region (2x2) in the input image and find the maximum value within it and that becomes the output of that particular patch. Also at the same time reducing the number of parameters associated by reducing the size of the output of the max pooling layer. The max pooling step can be well visualized from the following figure:

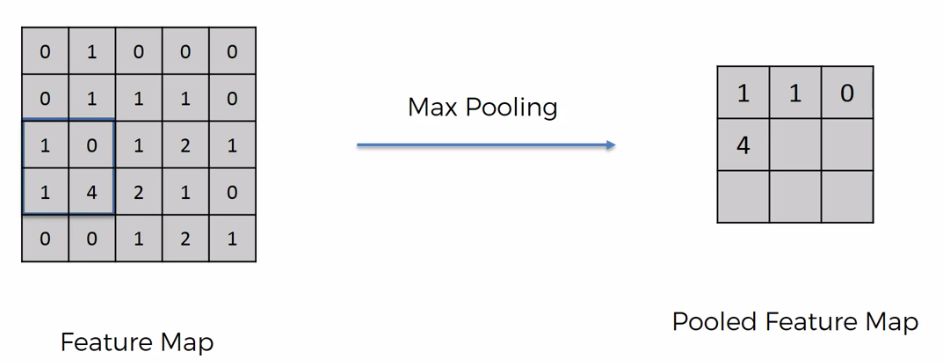


Fig. 16. Moving with a stride of two and taking the maximum at each step.

**3.4 Flattening**

The pooled feature map (a matrix of integer values) obtained in the pooling step is taken and it is flattened into a column and this column will serve as input for the artificial neural network for further processing. This step can visualized as in Fig. 15.

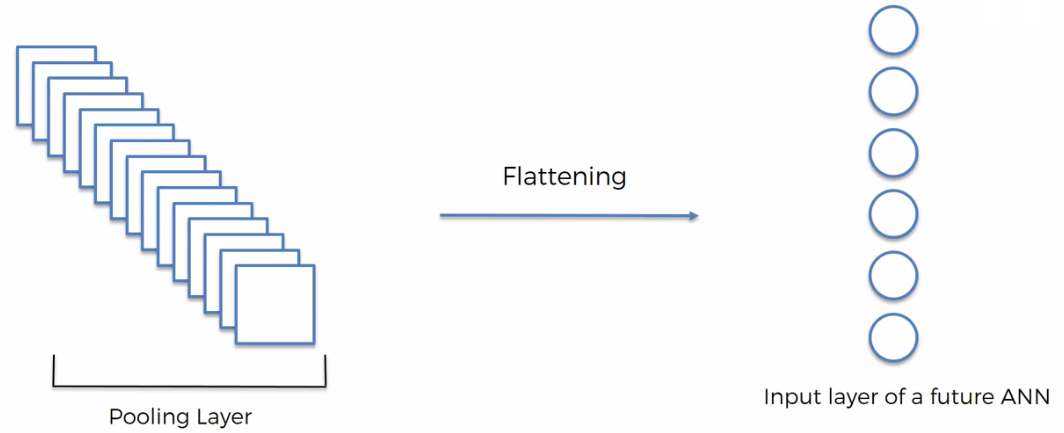


Fig. 17: Flattening

**2.10 Full Connection**

After flattening we connect our CNN with an artificial neural network in which all the hidden layers are fully connected and this process comes under the full connection step.

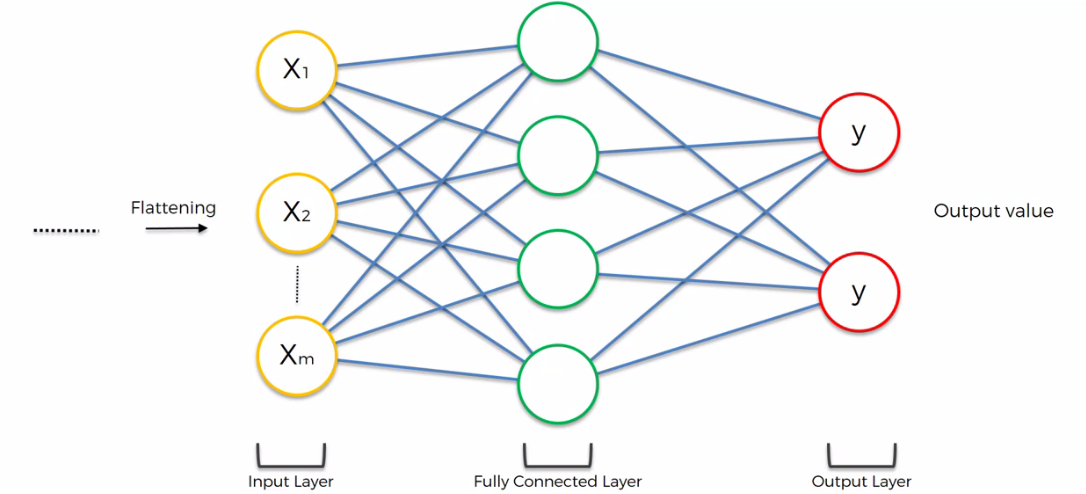
****

Fig. 18: Full connection

The output of this step becomes our final prediction.

**3. Results and Discussion**

The following images were used to test the validity of the model. The classifier correctly predicted the differect categories of animals in all the four cases:

****

cat\_or\_dog\_1 cat\_or\_dog\_2

****

cat\_or\_dog\_3 cat\_or\_dog\_4

Fig. 19: Input images used for prediction

In figure 20, the output of the prediction for the image ‘cat\_or\_dog\_3’ can be seen in the IPython console when the code for the single prediction is executed.

The result is stored in string format in a variable called prediction that can be seen in the variable explorer in fig. 20.

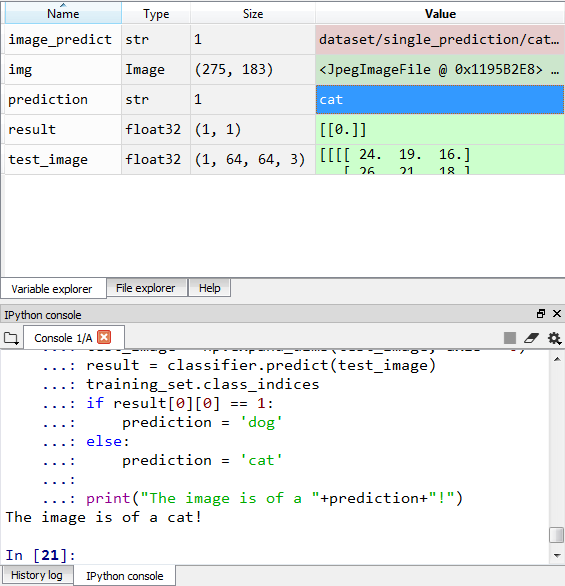
****

Fig. 20: Result for the image ‘cat\_or\_dog\_3’

**4. Conclusion(s) and Recommendations**

The keras model was trained with the training set consisting of 4000 images of dogs and cats and each and tested with 1000 images of dogs and cats each. The model consisted of two convolutional layers followed by two pooling layers respectively connected by two hidden layers. In one layer rectified linear function was used and in the other layer sigmoid function. The number of epochs for which the model was trained was 25 (an epoch is one complete iteration of all the elements in the dataset). The mean validation accuracy achieved by the model was 81.33 percent. The accuracy at each epoch can be seen in the following figure:

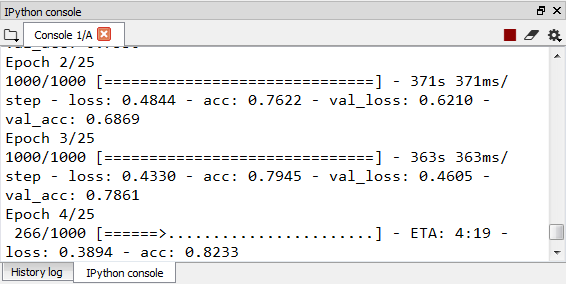


Fig. 21: Accuracies at different epochs

**5. Implication for Future Research**

The following image classifier predicts results with an accuracy of approximately 80%. Further accuracy can be achieved by making the convolutional neural network more deep, that is, increasing the number of hidden layers (fully connected layers) or by tuning different parameters in the search of finding a combination that gives the maximum accuracy. Also, using a powerful GPU and a huge dataset of different animals, an even bigger classifier can be created which is not only limited to two categories but can classify among all the animals.

**6. References**

* Yann LeCun et al., 1998, [*Efficient BackProp*](http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf)
* By Xavier Glorot et al., 2011, [*Deep sparse rectifier neural networks*](http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf)
* CrossValidated, 2015, [*A list of cost functions used in neural networks, alongside applications*](http://stats.stackexchange.com/questions/154879/a-list-of-cost-functions-used-in-neural-networks-alongside-applications)
* Andrew Trask, 2015, [*A Neural Network in 13 lines of Python (Part 2 – Gradient Descent)*](https://iamtrask.github.io/2015/07/27/python-network-part2/)
* Michael Nielsen, 2015, [*Neural Networks and Deep Learning*](http://neuralnetworksanddeeplearning.com/chap2.html)
* Yann LeCun et al., 1998, [*Gradient-Based Learning Applied to Document Recognition*](http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf)
* Jianxin Wu, 2017, [*Introduction to Convolutional Neural Networks*](http://cs.nju.edu.cn/wujx/paper/CNN.pdf)
* C.-C. Jay Kuo, 2016, [*Understanding Convolutional Neural Networks with A Mathematical Model*](https://arxiv.org/pdf/1609.04112.pdf)
* Kaiming He et al., 2015, [*Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*](https://arxiv.org/pdf/1502.01852.pdf)
* Dominik Scherer et al., 2010, [*Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition*](http://ais.uni-bonn.de/papers/icann2010_maxpool.pdf)
* Adit Deshpande, 2016, [*The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)*](https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html)
* Rob DiPietro, 2016, [*A Friendly Introduction to Cross-Entropy Loss*](https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/)
* Peter Roelants, 2016, [*How to implement a neural network Intermezzo 2*](http://peterroelants.github.io/posts/neural_network_implementation_intermezzo02/)