**The Maharaja Sayajirao University of Baroda, Vadodara.**
**Polytechnic**

PROJECT REPORT
ON

# IMAGE TO TEXT CONVERSION USING CONVOLUTIONAL NEURAL NETWORK

*Submitted by*

**VRAJ DESAI (607008)**

*In fulfilment for the Course of*

**DIPLOMA**

*in*

**Electronics and Communication (HPP)**

**2018-19**

The Maharaja Sayajirao University of Baroda, Vadodara.

**The Maharaja Sayajirao University of Baroda, Vadodara.**
**Polytechnic**

PROJECT REPORT
ON

# IMAGE TO TEXT CONVERSION USING CONVOLUTIONAL NEURAL NETWORK

*Submitted by*
**VRAJ DESAI (607008)**

*Guided by*
**MR. HIREN JETHAVA**
**MR. GAURAV PATEL**

*In fulfilment for the Course of*
**DIPLOMA**
*in*
**Electronics and Communication (HPP)**
**2018-19**

**The Maharaja Sayajirao University of Baroda, Vadodara.**
**Polytechnic**
**Electronics and Communication (HPP)**

# CERTIFICATE

This is to certify that, **MR. VRAJ DESAI,** has satisfactorily completed his Project Work as a part of course curriculum in Final Year Diploma in Electronics and Communication (HPP).
Titled As: **IMAGE TO TEXT CONVERSION USING CONVOLUTIONAL NEURAL NETWORK**

**Project Guide:**

**Mr. Hiren Jethava**
Lecturer
Electronics and Communication (HPP)
Polytechnic
The Maharaja Sayajirao University of Baroda.

**Mrs. Sheetal S. Shinkhede**
Astt. Director –EC (HPP)
Electronics and Communication (HPP)
Polytechnic
The Maharaja Sayajirao University of Baroda.

**Mr. Gaurav Patel**
Lecturer
Electronics and Communication (HPP)
Polytechnic
The Maharaja Sayajirao University of Baroda.

# ACKNOWLEDGEMENT

I want to thank our principal **Dr. V.H. Kher** sir for providing the powerful platform to achieve the magnificent deep learning.

I would like to thank our Astt. Director **Mrs. Sheetal S. Shinkhede** for believing and helping me out by providing the base for me to stand on.

There was an idea, to create a convolutional neural network that can learn and detect objects for computer and robot vision was successfully conquered with our E.C. faculty **Mr. Gaurav H. Patel** and **Mr. Hiren Jethava** sir who guided me to pursue this project and also helped me through various levels of MATLAB.

AI search engine of Google also helped me to get through the various queries I had about the project through the search engine that it provides. You-tube with its AI equipped search algorithm also helped we through the diverse video content about the convolutional neural network. **Mr. Siraj Raval** a youtuber through the you-tube platform also provided the content in the deep learning algorithm and its type.

Opening the lab in time or even before time for us to work on our project, providing the equipment so I would like to thank **Ms. Kinal Patel** and **Mr. Hemes Salat**.

And at the endgame I would like to thank my parents and MARVEL to free my mind and for entertainment.

# INDEX

## Abstract

*Image recognition and Object classification based on Camera Image identification is the crucial task for the organization which are working based on CCTV (Closed-Circuit Television). The project represents the use case of a section of deep learning called convolution neural network that converts shapes in images to text or character. Primary objects identification such as automobile, airplane, truck, and ship were used and in other case character from optical image 0-9 and A-Z is converted character.*

# Chapter 1

# Introduction

For long time there have been cameras and computer to display the images of videos but they weren't able to tell what the image or video contains. It is us that know what is in an image or video and what is it traying to convey. What is the special ingredient that makes use do that, what makes us identify the different objects around us, how did we learn to do classification. can we create that with the computer?

But first we must learn how the brain works, our brain is created with 100 billon neurons which trigger an electrical impulse which travel through the synapses to the connected neuron. Now whenever we learn anything new the brain creates a new connection through the neurons so whenever we recall any stored memory in the brain it fires the neuron and when the action phonetical goes above a threshold value it fires and through it the next set of neurons fire to get the memory and when a set of neurons fire simultaneously create a dissection.

Now this exact principle is used in a software neuron. In a software neuron for example there is variable that can be consider as a neuron and there are input to that variable which would be a input value now that input value is given to the neuron variable will have a bias now is the input value is more than a particular value the neuron output is high means one or zero is the input value is less than the action value.

MATLAB is used in this project because of its support for array and matrix computation. As in image processing the images consists of pixels and each pixel in an image is represented by a number matrix and MATLAB also have deep learning tool box which makes it easy to create a CONVOLUTIONAL NEURAL NETWORK (Convolution Neural Network) for image classification.

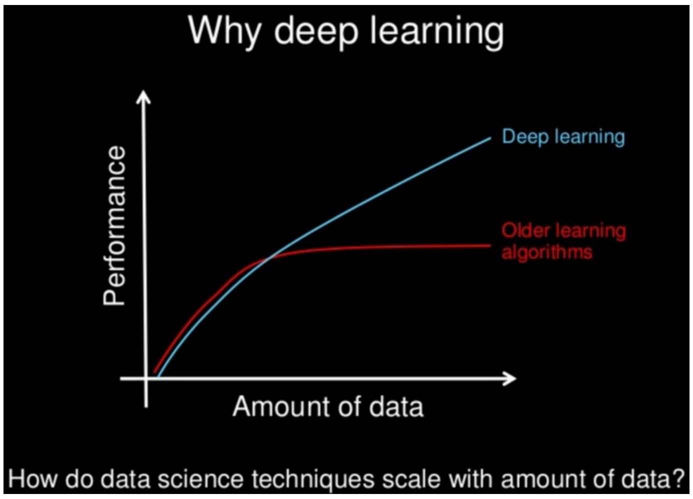
## 1.1 A new look

In traditional way of programing in which input from the user is taken and by applying logic of a programmer we get a desire output.  But AI (artificial Intelligence) is a new way of locking at the problem in which there is an input from the user and we now the output now the task of a computer is to map the input to the output in the most optimal way possible. AI can be divided into ML (Machine learning) and DL (Deep learning).

ML is a one of the ways to create AI in computer. In ML the features that we see the most optimal in an image, audio, or other property of an object, etc. is extracted and the computer is meant to map only the extracted features of the object form the input to output. Meanwhile in DL is a subset of ML. In DL the features form input to output mapping in solely done by the computer, so we don't have to extract any feature from the input for the computer.

DL is more like a human which doesn't know how to do a thing, but he/she knows the input in this case a cycle and knows the output means riding a cycle. So, a human would start riding a cycle and if he/she falls they try it again with a bit change in possession or the way of peddling the cycle and slow he/she learns the way to ride a cycle. This is essentially DL in which computer learns by changing the parameters.

The main advantage of DL over ML is that while a ML machine's performance plateaus out and after which increase in dataset will not improve the performance because of limited features it can map form input to output. Meanwhile in DL increasing in dataset will increase its performance because the computer learns and determine the parameters on its own. So, if we increase the dataset the computer has more data to train on and adjust its perimeters as shown below.



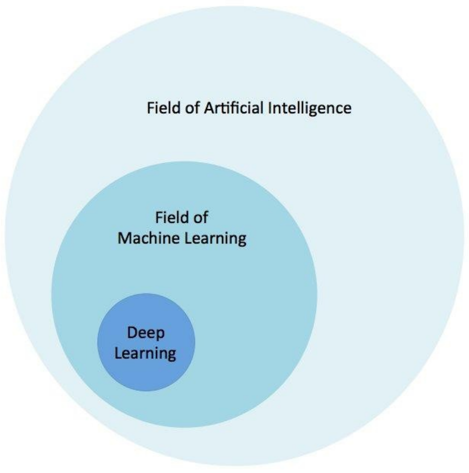


**Fig. 1.1 and 1.2**

# Chapter 2

# Deep learning

Deep learning is modern method of learning algorithm form 2012 till this time machine learning was preferred. In deep learning in there are n number of neurons like in human brain in one layer and there are one to two layers (deep network layer) and the last layer in output layer which predates output. The hidden layer or deep layer in modern network can go up to 10 to 100 layers each containing n number of neurons.

A deep learning network just like brain learns from the input data set by mapping the input data to the output neuron in the most optimum way possible. It does this by back propagation to find global minima of a nth dimension.

DL in this project uses supervised learning.

## 2.1 Supervised learning

Supervised learning is the Data mining task of inferring a function from labelled training data. The training data consist of a set of training examples.

## 2.2 Neuron

In deep learning a neuron like a brain trigger at an input data. It can consider as a variable which is equal to the input from the data or from the neuron from the previous layer. The input to a neuron is first multiplied by the weight and then added with a bias. The reason to multiple the input to the weight is that when using back propagation we can vary the weight accordingly to reduce the output error from the output neuron and the bias is used to shift the wave from to fit the data properly as weight is multiplied with the input data it is dependent and as bias if free from the input data it can be varied to provide a vast variety of change as shown below in Fig. 2.1.

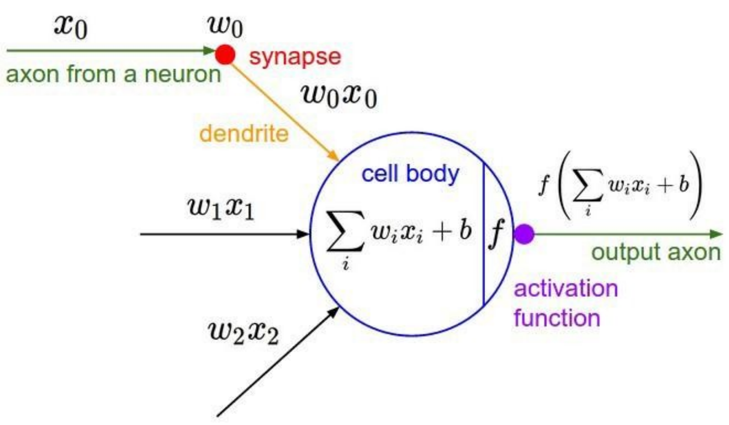


**Fig. 2.1**

## 2.3 Activation function

Activation function works as the name suggest, it fires the neuron. In this project I have used ReLU activation function shown below in fig 2.2,

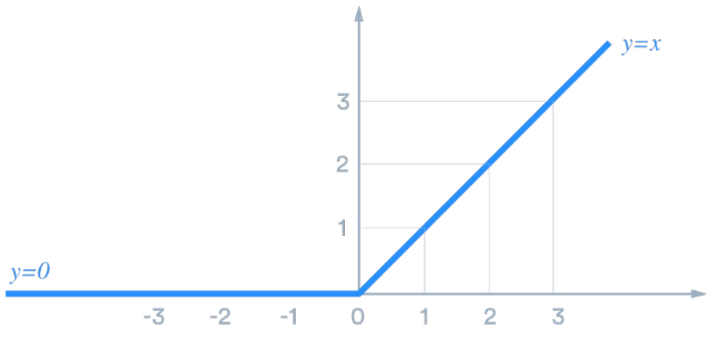


**Fig.2.2**

Before ReLU, tanh and sigemod function was used but as the change in output with respect to input is small so when we dravite it the derivation is very small and as in back propagation. The weights are updated by multiplying the gradient to the weight to reduce error the weights of the final layer will have high gradient, so they will change easily and as we move backwards layer by layer by chain rule the gradient multiple with each other making very small change in the initial layers one of the main problem in this is the initial layers are the most important layers as they are responsible for small feture in the input data which is the building block for complex feature extraction in the data.

So, using ReLU the gradient is large as in figure 2 the slope is constant, so each layer's neuron's weight is changed equally there used in the action function.

Now the activation function is also very important because it brings non linearity in the network because up to this point the weight and bias are of fist degree so it's linear so any linear function is a straight line and it cannot perfectly fit the solution as it can only rotate so a nonlinear function is can twist fold band and fit the data perfectly or optimally in any dimension.

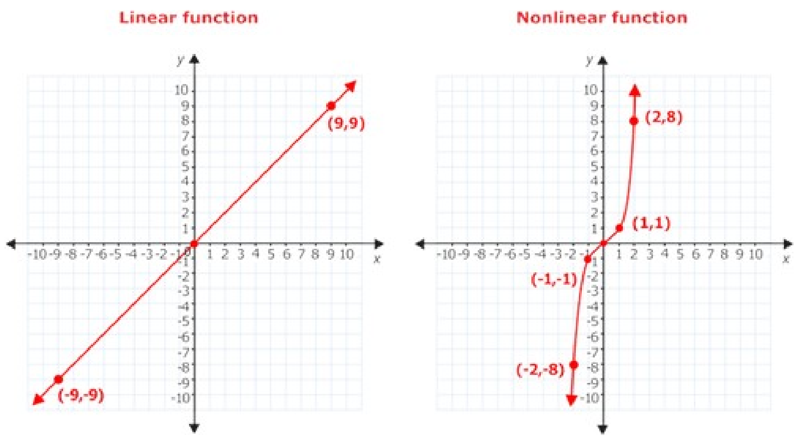


**Fig 2.3**

And non-linear function is also be differentiated in fig 2.3.

## 2.4 Backward pass

The backward pass is the main reason deep learning is used in which using back propagation is used to map the input data to the output neuron.

## 2.5 Loss function

Loss function is used to calculate the error generated by the network. In this project in case 1 the network classifies 4 objects in which the network will predict an output by firing an output neuron now if the output is wrong then the output is subtracted for the neuron to be fired for that input image and this is done for each output neuron and then average is found to calculate the error or cost.

The cost function changes for each input training data from a dataset.

## 2.6 Back propagation

A data can be an array of number or even an image can be a represented as a matrix of number which can be converted in to array and an array of n elements can be consider a point in a nth dimension and a group of data will create a wave form in a nth dimension.

The human mind cannot imagen beyond 3rd dimension so for example in a 2d parabola when we calculate the slope of the waveform when positive as we move it to leave the slope decreases and is the slope is negative we move right, did to this motion the slope decreases and we reach an optimum solution which with less loss.

This concept when expended to 3rd dimension the slope is called gradient dissent and by just adding a minus sine we move down in the most optimum solution.

So, the cost function's gradient dissent with respect to weights is multiplied with the weights of the neuron to the function reaches the most optimum solution in fig 2.4.

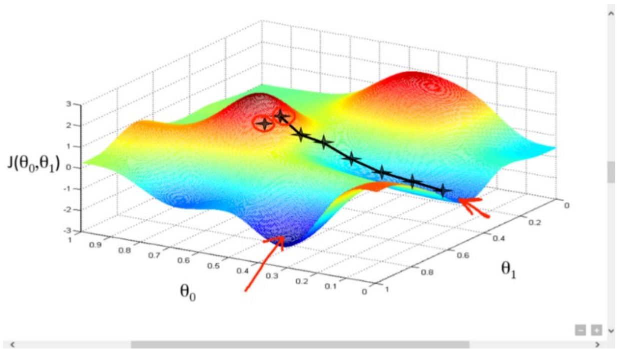


**Fig. 2.4**

# Chapter 3

# Technical survey

## 3.1 An Introduction to Convolutional Neural Networks: -

This was the main moving force in the choosing the image processing project using convolutional neural network which provided the foundation of the idea of artificial neural network (ANN). That these biologically inspired computational models can exceed the performance of previous forms of AI in common machine learning task.

This paper introduced me to the basic building block of convolutional neural network like: -

1. artificial neural (ANN).
2. Supervised learning.
3. Convolutional neural network (CONVOLUTIONAL NEURAL NETWORK).
4. Overfitting
5. Pooling
6. Kernel
7. Fully connected layer

## 3.2 Introduction to CONVOLUTIONAL NEURAL NETWORK by Stanford: -

This ppt introduced me the deep insight to the convolutional neural network various layers and how they work and how a classification layer of fully connected layer work with convolutional neural network to covert image to text.

## 3.3 Convolutional Neural Network by Siraj: -

This is YouTube video that gave me an insight to the convolution process happing in convolutional neural network at how exactly the kernel changes the input image to a feature map.

## 3.4 MATLAB: -

Various documents provided the meaning of various syntax and how-to setup various hyperparameter of how to train a convolutional neural network and the how to give input to the trained convolutional neural network network input to classify.

# Chapter 4

# Convolutional neural network

## 4.1 Image formation

Before going into the image feature extraction first it becomes much easier to know how an image is formed.

An image is made up of many pixels shown fig 4.1.

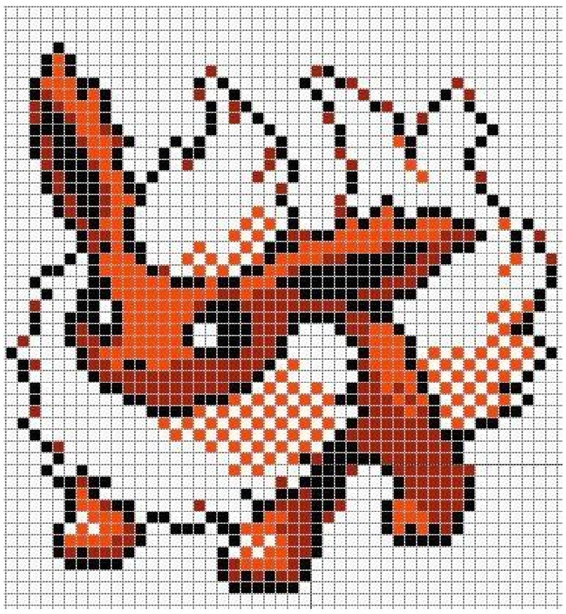


**Fig. 4.1**

Now each pixel is represented by a number from 0 to 255 for an 8 bits image and for one channel like red channel, blue channel, and green channel. So, for a 10 * 10 * 3 image so for 10*10 image there are 100 pixels in one channel and there are three such channel so 300 numbers represent an image of 10*10*3 where the channel is equal to depth.

## 4.2 Image classification

Image classification is the method of finding or classification of an image according to the content in the image.

## 4.3 Classical method

In classical image classification method the image is first converted into the array then the is feed to the input of the deep learning network that is equal to the number of elements in the input of the network each neuron in the input neuron is connected to the each neurons of the other neurons of the other following layer so like this the way there are many layers that form a deep network and the final output layer is the perdition of the network it is show in fig 4.2.

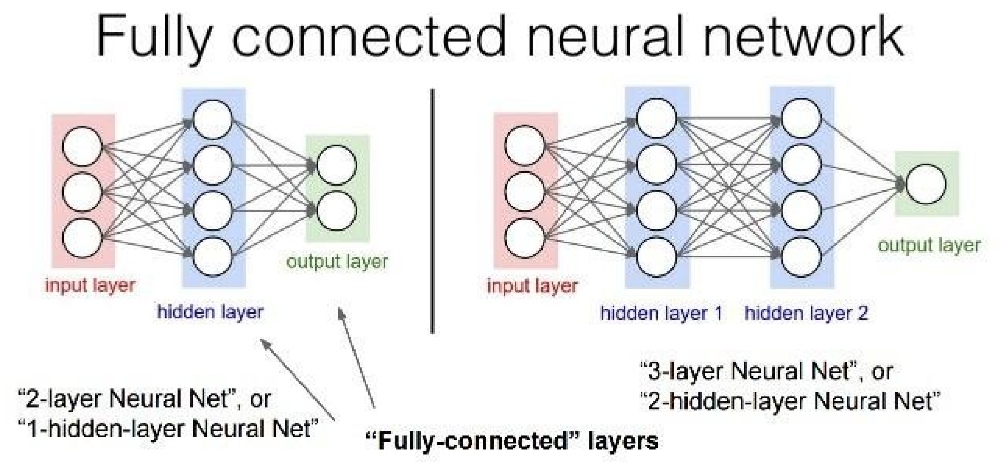


**Fig. 4.2**

Now each connection has a weight and each neuron has a bias so in 1 hidden layer which has the 4 neuron and the input has 3 neurons so each neuron in input layer is connected to each neuron of the hidden layer so 4 neurons in hidden layer as three connection so total of 12 weights and 4 bias.

The number is small but if the pixels are 300 so the hidden layer with just 4 neurons will have weight of 1200 so this number increases with image dimension, so the full connected layers classification network is not stated because of a lot of perimeter weights and biases so we need a lot of training data set to get a proper output.

## 4.4 Convolution Neural Network (CNN)

A convolution neural network is uses filter to reduce the perimeter as in fully connected each neuron is connected to the next layer neuron so in convolutional neural network only a part the input image is connected to the next layer form a pixel.

Convolution means comparing a filter or kernel to the inter image and if the kernel matches the image section it is being compared the is a large number and if the kernel compared to the image section is a very low value then we can say that the feature we are looking for in the image is not present in the image it is shown in image 4.3, 4.4, 4.5, and 4.6.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Pixel representation of filter

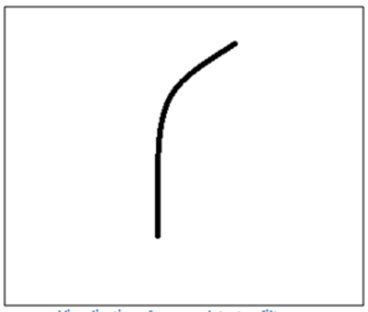

Visualization of a curve detector filter

**Fig. 4.3**

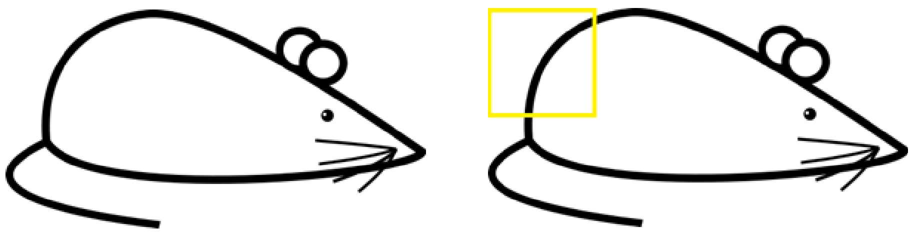
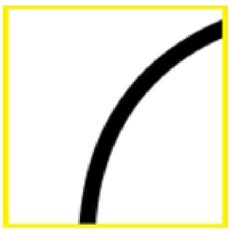


**Fig. 4.4**


Visualization of the receptive field

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 30 |
| 0 | 0 | 0 | 0 | 50 | 50 | 50 |
| 0 | 0 | 0 | 20 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |
| 0 | 0 | 0 | 50 | 50 | 0 | 0 |

Pixel representation of the receptive field

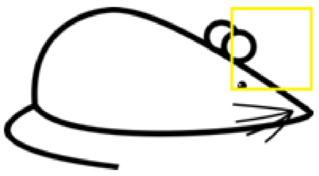
*

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

**Fig. 4.5**


Visualization of the filter on the image

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 40 | 0 | 0 | 0 | 0 | 0 |
| 40 | 0 | 40 | 0 | 0 | 0 | 0 |
| 40 | 20 | 0 | 0 | 0 | 0 | 0 |
| 0 | 50 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 50 | 0 | 0 | 0 | 0 |
| 25 | 25 | 0 | 50 | 0 | 0 | 0 |

Pixel representation of receptive field

*

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 30 | 0 |
| 0 | 0 | 0 | 0 | 30 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 30 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Pixel representation of filter

Multiplication and Summation = 0

**Fig. 4.6**

So, increasing the number of kernels will give us more feature to compare and obtain batter understanding of the image and through convolution we can reduce the image for the fully connected layer to learn and changing the kernel element we can learn the most optimal feature that helps us to get the information.

As an image is made up of matrix of number and there are three such matrix of 3 depth image. So, in convolutional neural network contains first convolution layer which has a filter of n*m where is n and m are integer so if there is a 100 * 100 * 3 image and a convolutional layer is 3*3 the depth must be equal to the depth of the input of the input image for of 3 and there may be n number of such filter with equal depth. One thing to note is the adding more filter gives us more spectral information of the image one thing to note is that not just add more filter is good because again a lot of perimeter again requires more dataset. Each filter creates a whole image so 20 filter will create a 20-depth image at the input on the previous layer.

So, convolution means just adding each elements of the filter which is weight is multiplied with each element of the image and added to gather to form one-pixel information for the output image it is shown in Fig. 4.7 and 4.8.

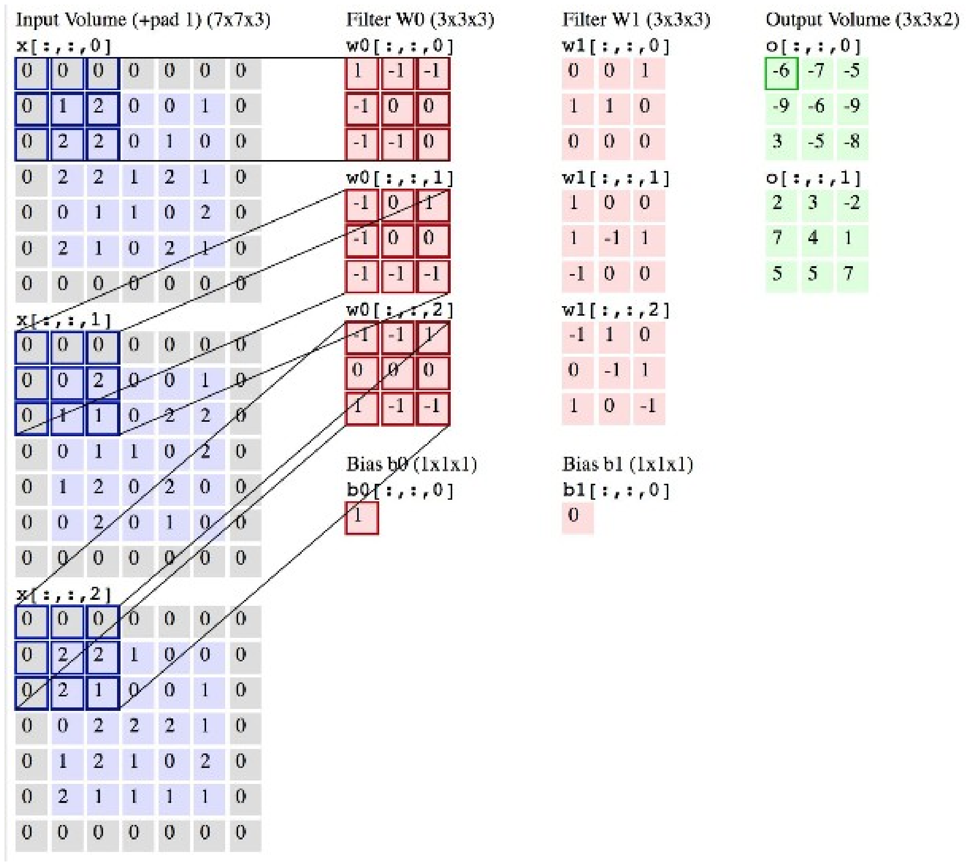
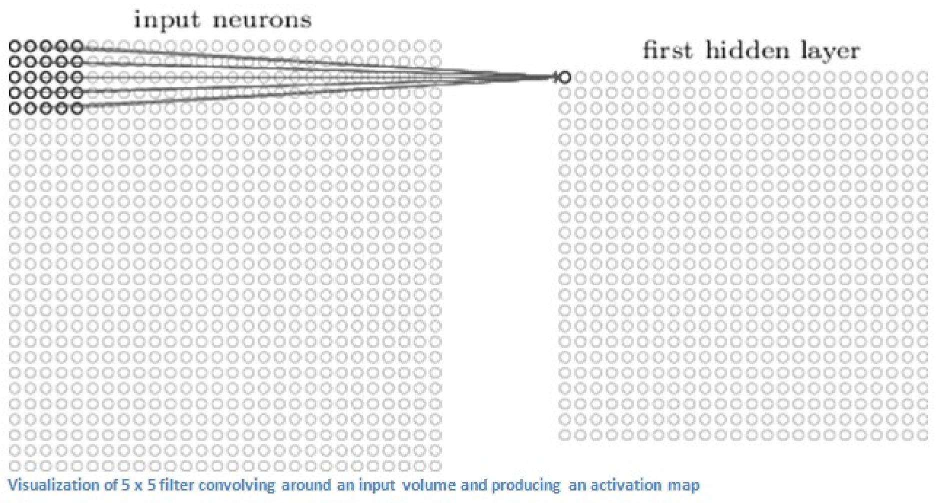


**Fig. 4.7**



input neurons
first hidden layer
Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

**Fig. 4.8**

Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on. The below figure shows convolution would work with a stride of 2 it is shown in Fig. 4.9.

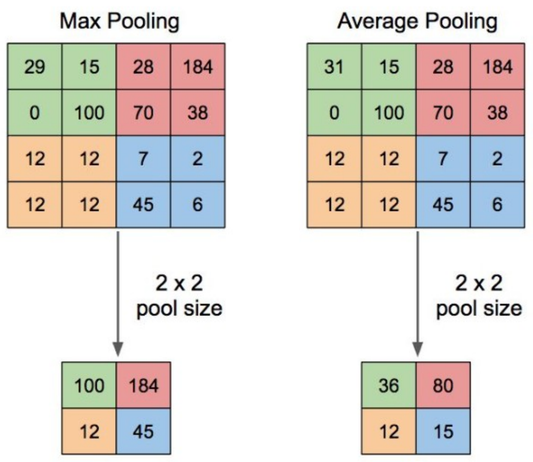


**Fig. 4.9**

Now after the feature extraction we do not need the complete image so to again reduce the image resolution and to obtain max feature we use polling to get the maximum number in a 2*2 filter in a 5*5 with a slider of 2 will just cut the image by half. And this does not reduce the information because maximum number will only contribute to the output for example see fig.

Now for activation ReLU is used which removes any number below zero and allows all positive number.

The layer we call as FC (fully connected) layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network.

So, by using convolutional neural network we can drastically reduce the parameters required for the classification of image. Not only that the convolutional neural network layer convolution layer innately extracts the simplest edges from the images like lines like horizontal and vertical lines then as the image goes further in the layer convolutes is detected and reaching to the final layer very fine details are detected is shown in fin. 4.10.

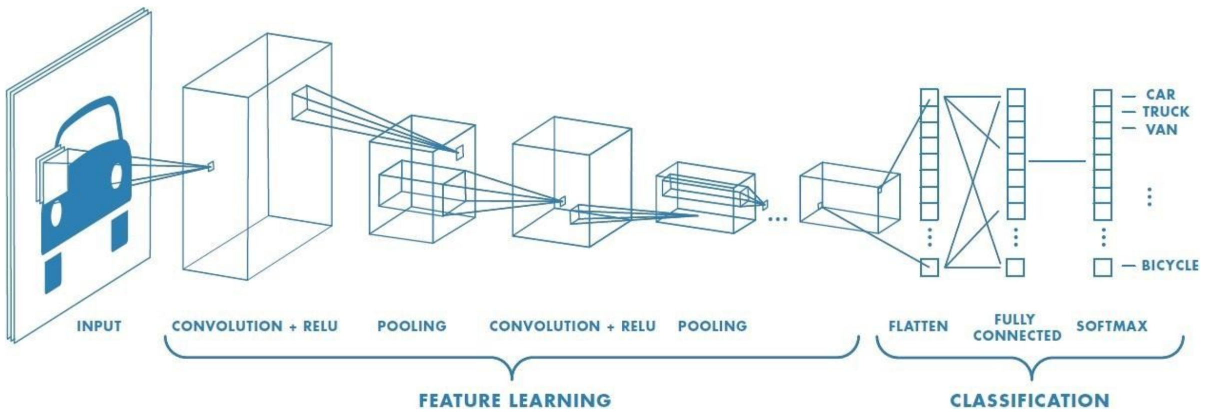


**Fig. 4.10**

So, a convolutional neural network is made up of an input layer the three are convolution, ReLU, and pooling section such section is repeated few times and at the final pooling layer the matrixes are flatten in to an array for the fully connected layer see fig.

The above network is good for learning and deploy it to work. But in full connected layer knows as classifier uses an activation pattern for same object in input image but this makes the network less acceptable to new data. It can be thought as humans get old they set in their life in only same pattern and adopting to something new becomes heard so to cure this behavior of the classifier network there is a dropout layer before the fatten layer and the succeeding fully connecter layers but not to the last layer which identifies the object or the output layer. The dropout layer fixes this by randomly making some of neuron's weight to zero so it forces the data to flow from new neuron, so the network can predict wide input data in can be seen below Fig 4.11.

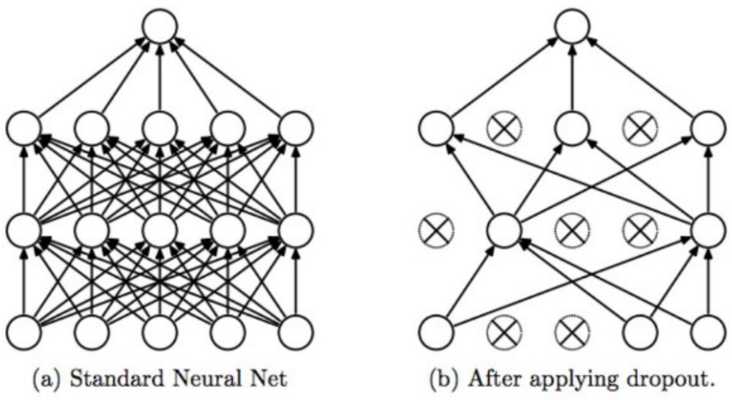


**Fig. 4.11**

# Chapter 5

# Training Convolutional neural network

There are many perimeters that can be changed to train a neural network.

## 5.1 Stochastic gradient descent (SGD)

SGD minibatch SGD, in this we don't have to evaluate the gradient for the whole training set but only for one sample or a minibatch of samples, this is usually much faster than batch gradient descent. Minibatches have been used to smooth the gradient and parallelize the forward and backpropagation.

## 5.2 Learning rate

Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. The lower the value, the slower we travel along the downward slope. While this might be a good idea (using a low learning rate) in terms of making sure that we do not miss any local minima, it could also mean that we'll be taking a long time to converge —especially if we get stuck on a plateau region.

Typically learning rates are configured naively at random by the user. At best, the user would leverage on past experiences (or other types of learning material) to gain the intuition on what is the best value to use in setting learning rates.

As such, it's often hard to get it right. The below diagram demonstrates the different scenarios one can fall into when configuring the learning rate shown in fig. 5.1.

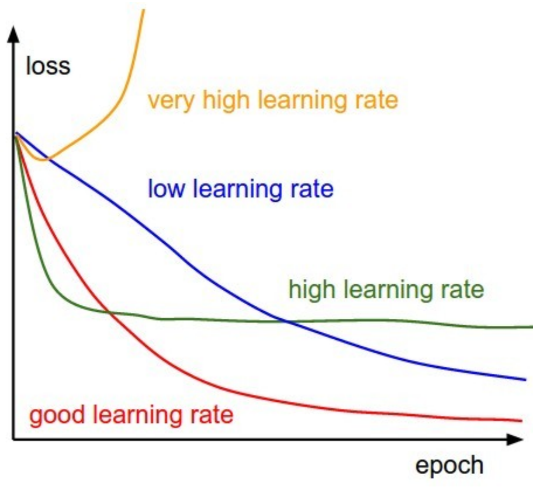


**Fig 5.1**

## 5.3 Mini-Batch Gradient Descent

Mini-batch gradient descent is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate model error and update model coefficients.

Implementations may choose to sum the gradient over the mini-batch or take the average of the gradient which further reduces the variance of the gradient.

Mini-batch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning.

In mini-batch gradient descent we must define hyper perimeter known as mini batch means who many images in a batch.

## 5.4 Epochs

One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE. Since one epoch is too big to feed to the computer at once we divide it in several smaller batches.

Now as we have limited dataset to train the neural network we pass the dataset more than one time so that we can update weights more than a single pass. So, we can remove underfitting problem in which the network can recognize the image. But while using epoch we should not overdo it then the network will be overfitting to the dataset and it will not predict the solution to a new data. This hyper perimeter is also found by trial and error, but I believe is that when accuracy plateaus out while an epoch training then the epoch should be reduced because the network is not creating error and is predicting accurately shown in Fig. 5.2.

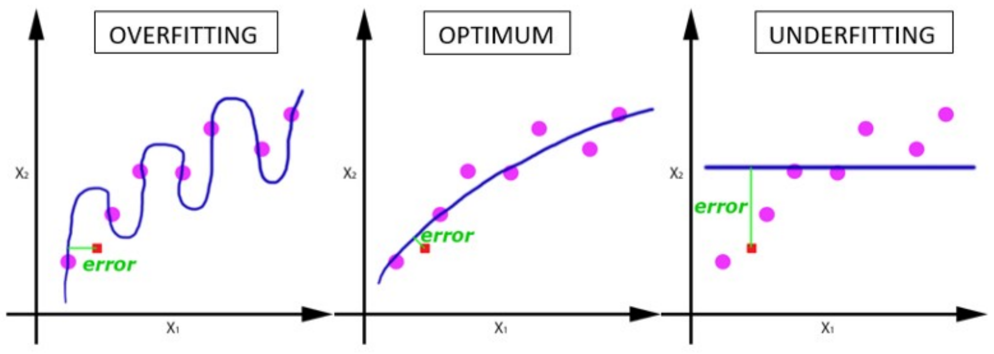


**Fig. 5.2**

## 5.5 Batch Size

Total number of training examples present in a single batch.

## 5.7 Iterations

Iterations is the number of batches needed to complete one epoch._We can divide the dataset of 2000 examples into batches of 500 then it will take 4 iterations to complete 1 epoch.

# Chapter 6

# Networks

## 6.1 Axiom CNN

Four object classification of airplanes, ship, automobile, and truck.

In the following code first, the categories are defined of the output neurons. And the root folder of which contains the images is kept in sperate folder in name of categories.

```
categories = {‘airplane’,’automobile’,’truck’,’ship’};

rootFolder = 'cifar10Train';

imds = imageDatastore(fullfile(rootFolder, categories), ...

   'LabelSource', 'foldernames');
```

The imds creates the memory allocation of images with their folder name as their category and it preprocess the dataset for training.

```
varSize = 32;

conv1 = convolution2dLayer(5,varSize,'Padding',2,'BiasLearnRateFactor',2);

conv1.Weights = gpuArray(single(randn([5 5 3 varSize])*0.0001));

fc1 = fullyConnectedLayer(64,'BiasLearnRateFactor',2);

fc1.Weights = gpuArray(single(randn([64 576])*0.1));

fc2 = fullyConnectedLayer(4,'BiasLearnRateFactor',2);

fc2.Weights = gpuArray(single(randn([4 64])*0.1));
```

The varsize is the image size. Conv1 crates a convolution layer of 5*5 filter with 3 depth and 32 of them and of padding of 2 pixel to input image and the weights of conv1 can be changed from there base value and the 0.0001 defines how close the random weight for convolution layer.

The fc1 is of 64 neuron and is full connected to fc2 layer which is of 4 neuron which is equal to the output categories.

The layers function crates the layers defined in it as one input layer of 32*32*3. And the average polling to2 layer is used to average out the element in the filter size of the polling layer.

```
layers = [
    imageInputLayer([varSize varSize 3]);

    conv1;

    maxPooling2dLayer(3,'Stride',2);

    reluLayer();

    convolution2dLayer(5,32,'Padding',2,'BiasLearnRateFactor',2);

    reluLayer();

    averagePooling2dLayer(3,'Stride',2);

    convolution2dLayer(5,64,'Padding',2,'BiasLearnRateFactor',2);

    reluLayer();

    averagePooling2dLayer(3,'Stride',2);

    fc1;

    reluLayer();

    fc2;

    softmaxLayer()

    classificationLayer()];

opts = trainingOptions('sgdm', ...

    'InitialLearnRate', 0.001, ...

    'LearnRateSchedule', 'piecewise', ...

    'LearnRateDropFactor', 0.1, ...

    'LearnRateDropPeriod', 8, ...

    'L2Regularization', 0.004, ...

    'MaxEpochs', 10, ...

    'MiniBatchSize', 100, ...

    'Verbose', true,...

'Plots','training-progress');
```

The learning rate is 0.001 that is multiplied wile modifying the weight and after each 8 epoch the learning late is decreased by 0.1 and the verbose is used to create the table of accuracy, loss, learning rate, ect.ant plots poltes the training progress.

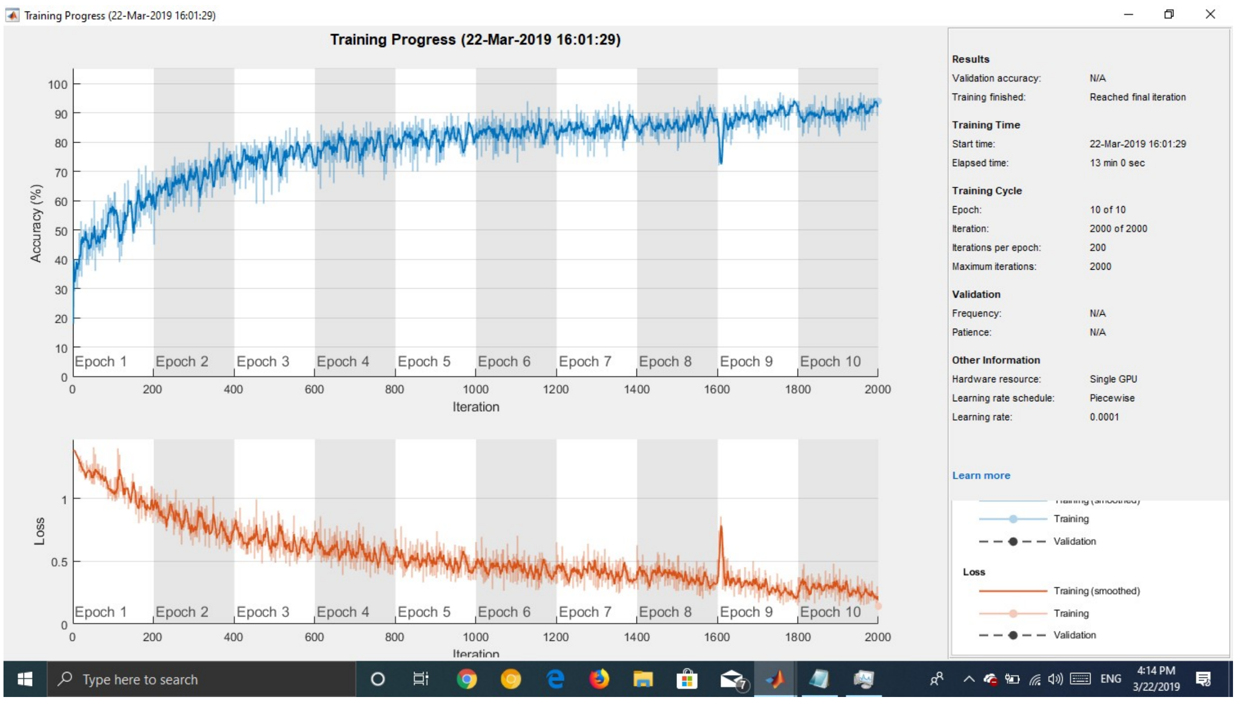
*[net, info] = trainNetwork(imds, layers, opts);*

The above line trains the neural network with imds data on layers with opts options.

*Axion = net*

*Save axion*

The training progress of Axion is shown below: -



**Results**

| | |
|---|---|
| Validation accuracy: | N/A |
| Training finished: | Reached final iteration |

**Training Time**

| | |
|---|---|
| Start time: | 22-Mar-2019 16:01:29 |
| Elapsed time: | 13 min 0 sec |

**Training Cycle**

| | |
|---|---|
| Epoch: | 10 of 10 |
| Iteration: | 2000 of 2000 |
| Iterations per epoch: | 200 |
| Maximum iterations: | 2000 |

**Validation**

| | |
|---|---|
| Frequency: | N/A |
| Patience: | N/A |

**Other Information**

| | |
|---|---|
| Hardware resource: | Single GPU |
| Learning rate schedule: | Piecewise |
| Learning rate: | 0.0001 |

Learn more

**Fig. 6.1**

## 6.2 OCRV3 CNN

In this program I have created a convolutional neural network that can classify the 0-9 and A-Z optical character .

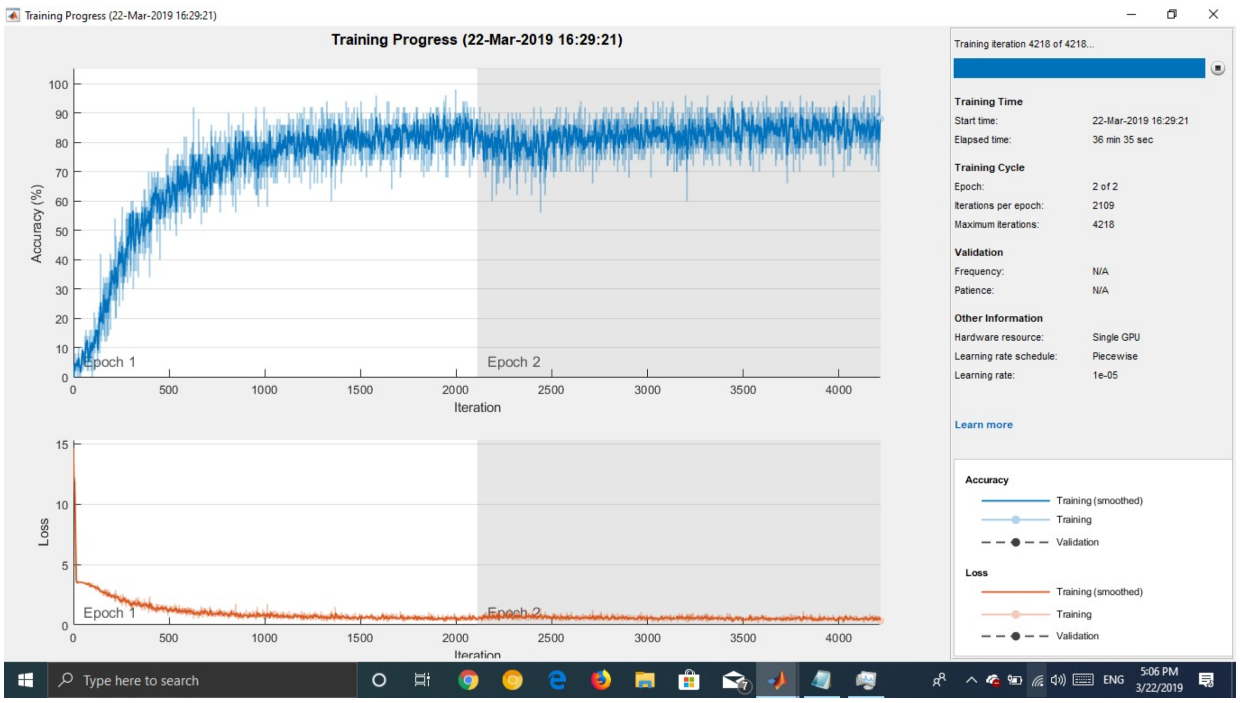
*categories= {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};*

*rootFolder = 'OCR';*

*imds = imageDatastore(fullfile(rootFolder, categories), ...*

```
    'LabelSource', 'foldernames');
cnn1=convolution2dLayer(5,10);
cnn1.Weights = gpuArray(single(randn([5 5 3 10])*0.0001));
cnn2=convolution2dLayer(5,20);
cnn2.Weights = gpuArray(single(randn([5 5 10 20])*0.001));
cnn3=convolution2dLayer(5,40);
cnn3.Weights = gpuArray(single(randn([5 5 20 40])*0.001));
fc1= fullyConnectedLayer(300);
fc1.Weights = gpuArray(single(randn([300 5760])*0.1));
fc1.Bias = gpuArray(single(randn([300 1])*0.1 + 1));
fc2= fullyConnectedLayer(36);
fc2.Weights = gpuArray(single(randn([36 300])*0.1));
fc2.Bias = gpuArray(single(randn([36 1])*0.1 + 1));
layers = [
    imageInputLayer([128 128 3])
    cnn1
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    cnn2
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    cnn3
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)

    dropoutLayer(0.25)

    fc1

    reluLayer

    fc2

    softmaxLayer

    classificationLayer]

opts = trainingOptions('sgdm', ...

    'InitialLearnRate', 0.01, ...

    'LearnRateSchedule', 'piecewise', ...

    'LearnRateDropFactor', 0.001, ...

    'LearnRateDropPeriod', 1, ...

    'L2Regularization', 0.004, ...

    'MaxEpochs', 2, ...

    'MiniBatchSize', 50, ...

    'Verbose', true,...

    'Shuffle', 'every-epoch',...

    'Plots','training-progress');

[net, info] = trainNetwork(imds, layers, opts);

OCRV3 = net

Save OCRV3
```

The training progress of OCRV3 is shown below: -



Training iteration 4218 of 4218...

**Training Time**

| | |
|---|---|
| Start time: | 22-Mar-2019 16:29:21 |
| Elapsed time: | 36 min 35 sec |

**Training Cycle**

| | |
|---|---|
| Epoch: | 2 of 2 |
| Iterations per epoch: | 2109 |
| Maximum iterations: | 4218 |

**Validation**

| | |
|---|---|
| Frequency: | N/A |
| Patience: | N/A |

**Other Information**

| | |
|---|---|
| Hardware resource: | Single GPU |
| Learning rate schedule: | Piecewise |
| Learning rate: | 1e-05 |

Learn more

**Fig. 6.2**

To see how to use Axion and OCRV3 network in to convert image to text see appendix 1.

# Chapter 7

# Conclusion

Convolutional neural network has been present for a while. Before Convolutional neural network, fully connected layer was used to convert images to text. Now using convolutional principle, we can extract the most relevant feature and then use these optimal features to convert image into text using fully connected layer as their supreme power in classification or mapping the input parameters to output. This project is using Convolutional neural network to separate the various shapes in the images to convert into text using webcam, four objects were detected in section 1 while in section 2 numbers from optical images of numbers from 0 to 9 and characters from A to Z (Capital) were detected using image segmentation through Convolutional Neural Network.

# Chapter 8

# Future scope

This is just the begin of something extricating in the field of AI and using convolutional neural network we can not only convert the shapes in images to text but if we flip the network and give input as text and using deconvolution we can generate image and if we have book then using deconvolution we man make continues video by using only text.

convolutional neural network can also be used to decrease the flackeries in a video due to low framerate like 15 to increase it to 30 or a video of 30 to 60 we can use convolutional and deconvolutional layer after it except fully connected layer the we can create inter frame between two images do every odd frame is present and every even frame will be given by the network.

convolutional neural network and Deconvolutional neural network can also be used to increase the image resolution form 2k to 4k in the most optimal way.

Deep learning is very prominent right now, but the future is one short learning in which we need small dataset to learn from the image.

Using convolutional neural network, we can create even a rover that can do in deep space exploration that don’t need human and just by visual understanding to make decision.

One step above this project we can detect multiple objects from a frame.

Using convolutional neural network, we can automatically detect censers cell from endoscopes and destroy it without doctors and not only censer, but any imaging system used in medical like retina scan and finger out many data out of it.

# Appendix 1

## 1 Axion

To use axion to convert airplane, truck, automobile, and ship using the laptop camera.

```
camera = webcam;
nnet = Axion;
while true
picture = camera.snapshot;
picture = imresize(picture,[227,227]);
label = classify(nnet, picture);
image(picture);
title(char(label));
drawnow;
end
```

## 2 OCRV3

To use OCRV3 and the images are perfectly spaced the by using the flowing code we can recognize the 10 character  s stored in a folder.

```
r = i(:,:,1); % Red channel

a1=r(1:128, 1:128);
a2=r(1:128, 129:256);
a3=r(1:128, 257:384);
a4=r(1:128, 385:512);
a5=r(1:128, 513:640);
a6=r(1:128, 641:768);
a7=r(1:128, 769:896);
a8=r(1:128, 897:1024);
a9=r(1:128, 1025:1152);
a10=r(1:128, 1153:1280);

a1 = cat(3, a1, a1, ac1);
a2 = cat(3, a2, a2, a2);
a3 = cat(3, a3, a3, a3);
```

```
a4 = cat(3, a4, a4, a4);
a5 = cat(3, a5, a5, a5);
a6 = cat(3, a6, a6, a6);
a7 = cat(3, a7, a7, a7);
a8 = cat(3, a8, a8, a8);
a9 = cat(3, a9, a9, a9);
a10 = cat(3, a10, a10, a10);

load ocrv3

l1 = classify(ocrv3, a1);
l2 = classify(ocrv3, a2);
l3 = classify(ocrv3, a3);
l4 = classify(ocrv3, a4);
l5 = classify(ocrv3, a5);
l6 = classify(ocrv3, a6);
l7 = classify(ocrv3, a7);
l8 = classify(ocrv3, a8);
l9 = classify(ocrv3, a9);
l10 = classify(ocrv3, a10);
x= {l1,l2,l3,l4,l5,l6,l7,l8,l9,10}

image(i);
title(char(x));
drawnow;
```

And if the chatters are randomly placed then the flowing program is used: -

```
y=50;
z=50;

i=imread('t11.jpg');
i2=imbinarize(i);

[h,w]=size(i2);
```

```
for j=1:h
for k=1:w
if(i2(j,k)==1)
i2(j,k)=0;
else
i2(j,k)=255;
end
end
end

L = bwlabel(i2,8);
[r, c] = find(L==1);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
a1=i2(Nw:Mw, Nl:Ml);
[R,C] = size(a1);
a1 = padarray(a1,[y z],0,'both');

[r, c] = find(L==2);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
a2=i2(Nw:Mw, Nl:Ml);
[R,C] = size(a2);
a2 = padarray(a2,[y z],0,'both');

[r, c] = find(L==3);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
a3=i2(Nw:Mw, Nl:Ml);
```

```
[R,C] = size(a3);
a3 = padarray(a3,[y z],0,'both');

[r, c] = find(L==4);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
a4=i2(Nw:Mw, Nl:Ml);
[R,C] = size(a4);
a4 = padarray(a4,[y z],0,'both');

[r, c] = find(L==5);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
a5=i2(Nw:Mw, Nl:Ml);
[R,C] = size(a5);
a5 = padarray(a5,[y z],0,'both');

[r, c] = find(L==6);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
a6=i2(Nw:Mw, Nl:Ml);
[R,C] = size(a6);
a6 = padarray(a6,[y z],0,'both');

[r, c] = find(L==7);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
```

```
a7=i2(Nw:Mw, Nl:Ml);
[R,C] = size(a7);
a7 = padarray(a7,[y z],0,'both');

[r, c] = find(L==8);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
a8=i2(Nw:Mw, Nl:Ml);
[R,C] = size(a8);
a8 = padarray(a8,[y z],0,'both');

[r, c] = find(L==9);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
a9=i2(Nw:Mw, Nl:Ml);
[R,C] = size(a9);
a9 = padarray(a9,[y z],0,'both');

[r, c] = find(L==10);
Ml = max(c);
Nl = min(c);
Mw = max(r);
Nw = min(r);
a10=i2(Nw:Mw, Nl:Ml);
[R,C] = size(a10);

a10 = padarray(a10,[y z],0,'both');
 a1= imresize(a1,[128,128]);
 a2= imresize(a2,[128,128]);
 a3= imresize(a3,[128,128]);
 a4= imresize(a4,[128,128]);
 a5= imresize(a5,[128,128]);
 a6= imresize(a6,[128,128]);
 a7= imresize(a7,[128,128]);
 a8= imresize(a8,[128,128]);
 a9= imresize(a9,[128,128]);
 a10= imresize(a10,[128,128]);

 invert_sp_ch;

 a1 = cat(3, a1, a1, a1);
 a2 = cat(3, a2, a2, a2);
 a3 = cat(3, a3, a3, a3);
 a4 = cat(3, a4, a4, a4);
 a5 = cat(3, a5, a5, a5);
 a6 = cat(3, a6, a6, a6);
 a7 = cat(3, a7, a7, a7);
 a8 = cat(3, a8, a8, a8);
 a9 = cat(3, a9, a9, a9);
 a10 = cat(3, a10, a10, a10);

load ocrv3

l1 = classify(ocrv3, a1);
l2 = classify(ocrv3, a2);
l3 = classify(ocrv3, a3);
l4 = classify(ocrv3, a4);
l5 = classify(ocrv3, a5);
l6 = classify(ocrv3, a6);
l7 = classify(ocrv3, a7);
l8 = classify(ocrv3, a8);
l9 = classify(ocrv3, a9);
l10 = classify(ocrv3, a10);

x=[l1,l2,l3,l4,l5,l6,l7,l8,l9,l10];
image(i);
```

```
title(char(x));
drawnow;
```

The invert_sp_ch function is shown below: -

```
for j=1:128
for k=1:128
if(a1(j,k)==0)
a1(j,k)=255;
else
a1(j,k)=0;
end
end
end

for j=1:128
for k=1:128
if(a2(j,k)==0)
a2(j,k)=255;
else
a2(j,k)=0;
end
end
end

for j=1:128
for k=1:128
if(a3(j,k)==0)
a3(j,k)=255;
else
a3(j,k)=0;
end
end
end
```

```
for j=1:128
for k=1:128
if(a4(j,k)==0)
a4(j,k)=255;
else
a4(j,k)=0;
end
end
end

for j=1:128
for k=1:128
if(a5(j,k)==0)
a5(j,k)=255;
else
a5(j,k)=0;
end
end
end

for j=1:128
for k=1:128
if(a6(j,k)==0)
a6(j,k)=255;
else
a6(j,k)=0;
end
end
end

for j=1:128
for k=1:128
if(a7(j,k)==0)
a7(j,k)=255;
else
```

```
a7(j,k)=0;
end
end
end

for j=1:128
for k=1:128
if(a8(j,k)==0)
a8(j,k)=255;
else
a8(j,k)=0;
end
end
end

for j=1:128
for k=1:128
if(a9(j,k)==0)
a9(j,k)=255;
else
a9(j,k)=0;
end
end
end

for j=1:128
for k=1:128
if(a10(j,k)==0)
a10(j,k)=255;
else
a10(j,k)=0;
end
end
end
```

# References

https://www.researchgate.net/publication/285164623_An_Introduction_to_Convolutional_Neural_Networks

https://web.stanford.edu/class/cs231a/lectures/intro_cnn.pdf

https://www.youtube.com/channel/UCWN3xxRkmTPmbKwht9FuE5A

https://www.youtube.com/watch?v=FTr3n7uBIuE

https://machinelearningmastery.com/what-is-deep-learning/

https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

https://in.mathworks.com/matlabcentral/fileexchange/60659-deep-learning-in-11-lines-of-matlab-code

http://www.human-memory.net/brain_neurons.html

https://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm

https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks

https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9

https://in.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html#mw_bac20c29-f95b-423a-816a-428fc8e0463e

https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

https://www.learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/

https://in.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html

https://in.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html
https://in.mathworks.com/help/deeplearning/ref/trainingoptions.html

https://in.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html#mw_bac20c29-f95b-423a-816a-428fc8e0463e