Image Classification Using Convolutional Neural Network:-

Suyash(1711MC13)

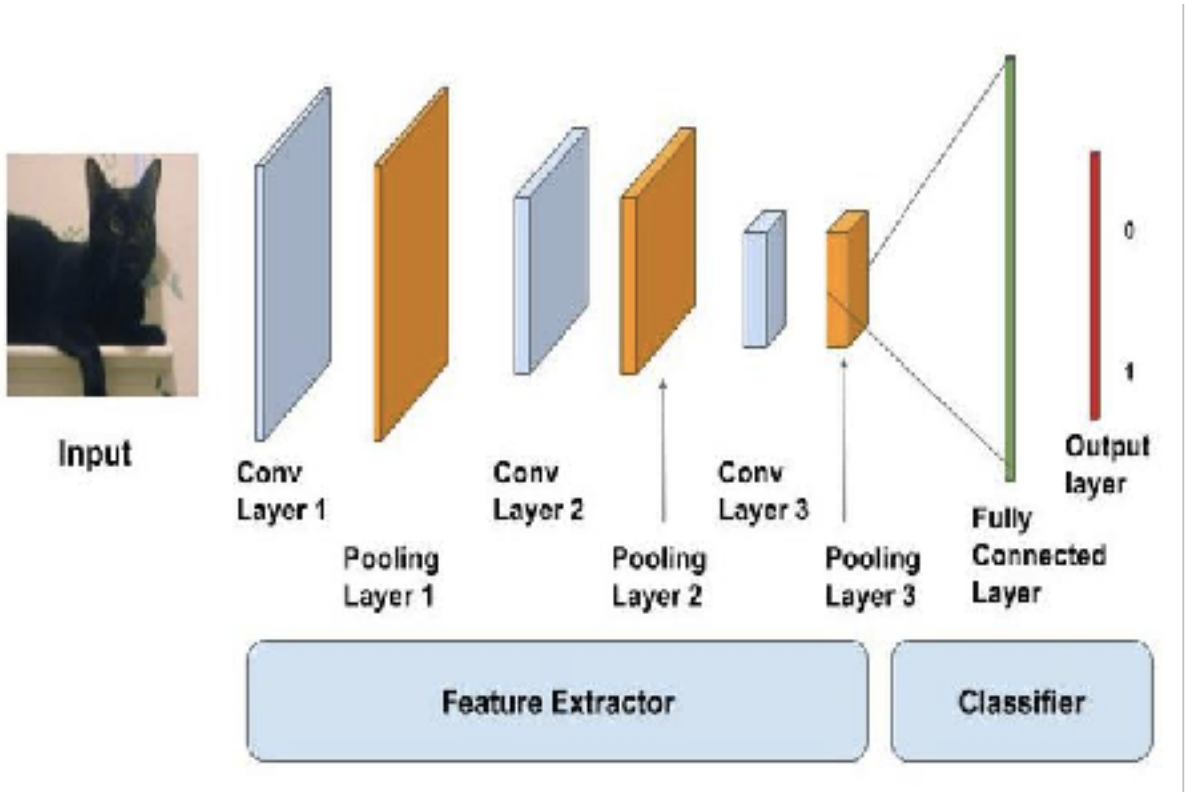
Abstract:- With images becoming the fastest growing content, image classification has become a major driving force. If we use a **Multilayer perceptron** for image classification, performance could be good as we can achieve good accuracy but there is a problem with this approach. If in our training dataset, all images are centered and test set are off-centered, then the MLP approach fails miserably. We want the network to be **Translation-Invariant.** So, either we have to train separate MLPs for different locations or we have to make sure that we have all these variations in the training set as well, which is quite difficult. The second option i.e. **Fully connected network** tries to learn global features or patterns. It acts as a good classifier but the number of parameters increases very fast since each node in layer L is connected to a node in layer L-1. Both the above problems of overfitting and translation are solved to a great extent by using Convolutional Neural Networks. So, we will be using CNN having multiple layers to classify images in our project.

Introduction:- To demonstrate how to build a convolutional neural network based image classifier, we

shall build a 6 layer neural network that will identify and separate images even of dogs from that of cats.Classification algorithms typically employ two phases of processing: *training* and *testing*. In the initial training phase, characteristic properties of typical image features are isolated and, based on these, a unique description of each classification category, *i.e. training class*, is created. In the subsequent testing phase, these feature-space partitions are used to classify image features.
We are going to teach the computer to recognise images and classify them into one of the 10 classes we have. For this we need to teach the computer how each and every class looks like, before it being able to recognise a new image. This is what we call as supervised learning. The more images of a particular class the computer sees, the better it starts recognising that class. But we have to stop training before it gets overfitted. Hence in CNN, the computer will start learning the global features.

Keras and Tensorflow:- We are going to implement this using Python version 2.7 and a high level neural network API written in Python i.e. Keras which enables fast experimentation and supports CNN and RNN and even combination of both. Keras has Tensorflow in backend and comes with many inbuilt libraries which are directly being used in our project.
Core data structure of Keras is a model, which is a way to organise layers. Simplest type is SEQUENTIAL which is a linear stack of layers and we will be using it.

Convolutional Neural Networks are a form of feedforward neural networks. In feature extractor part, we have convolutional and pooling layers. In the second part we have fully connected layer which performs non-linear transformations of the extracted features and acts as the classifier. CNN are a special type of neural networks having convolution layer at the beginning.



Dataset:- We have total 10 classes in our dataset and those are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. We are using CIFAR-10 dataset consisting of 60,000 (32*32) coloured images with 6,000 images per class. They have been divided into 50,000 training images and 10,000 test images. The images are small and clearly labeled, which are noise free and

hence making the dataset efficient for correct classification.

Splitting Our Dataset:- Since we are having very huge dataset, we are using mini batch stochastic gradient descent. We are using mini batches of size = 256 each and training each for 50 epochs. We are randomly selecting 50,000 images for training purpose and 10,000 for testing.

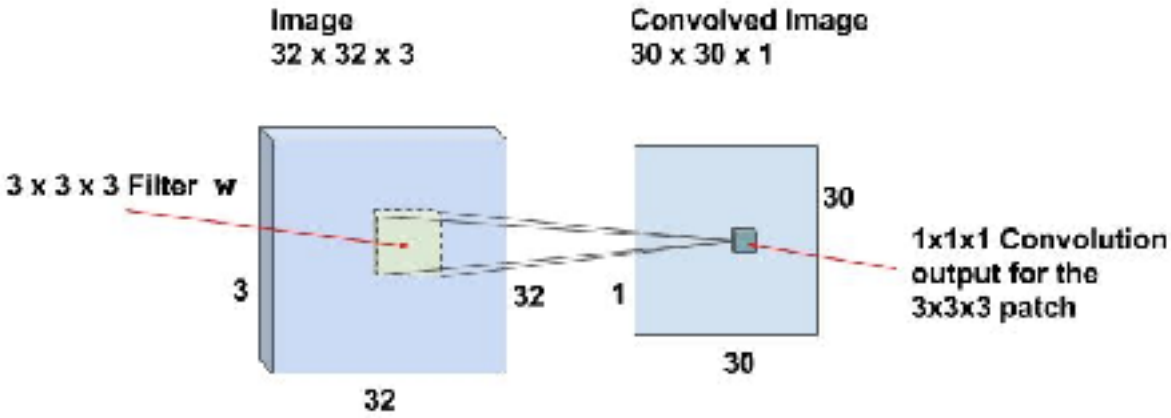
(A). Using RGB Values:-

We are having all coloured images in our dataset so whenever computer interprets an image, it extracts the RGB values (0-255) of each pixel and stores the result in an array. The computer then allots confidence scores for each class. The class with the highest confidence score is usually the predicted one.

(B). LAYERS :-

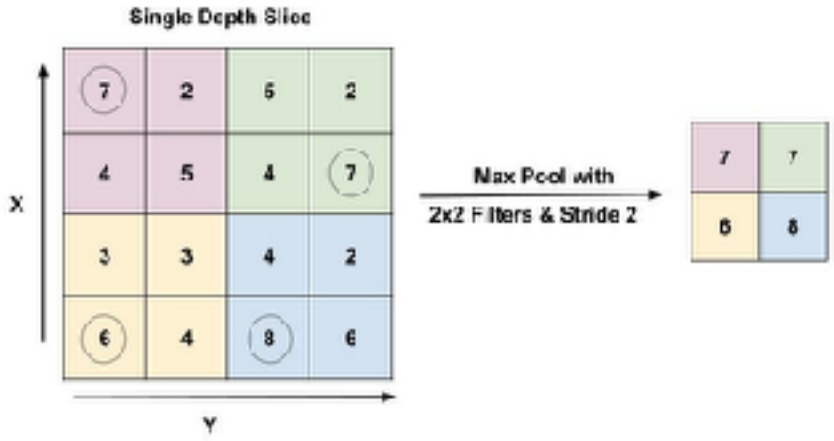
Different types of layers are :- Convolutional, Pooling, Dropout, Dense and Flattening.

1. **Convolutional Layer** :- The neurons in this layer look for specific features. If they find the features they are looking for, they produce a high activation. Suppose, the input image is of size 32x32x3. This is nothing but a 3D array of depth 3. **Any convolution filter we define at this layer must have a depth equal to the depth of the input.**

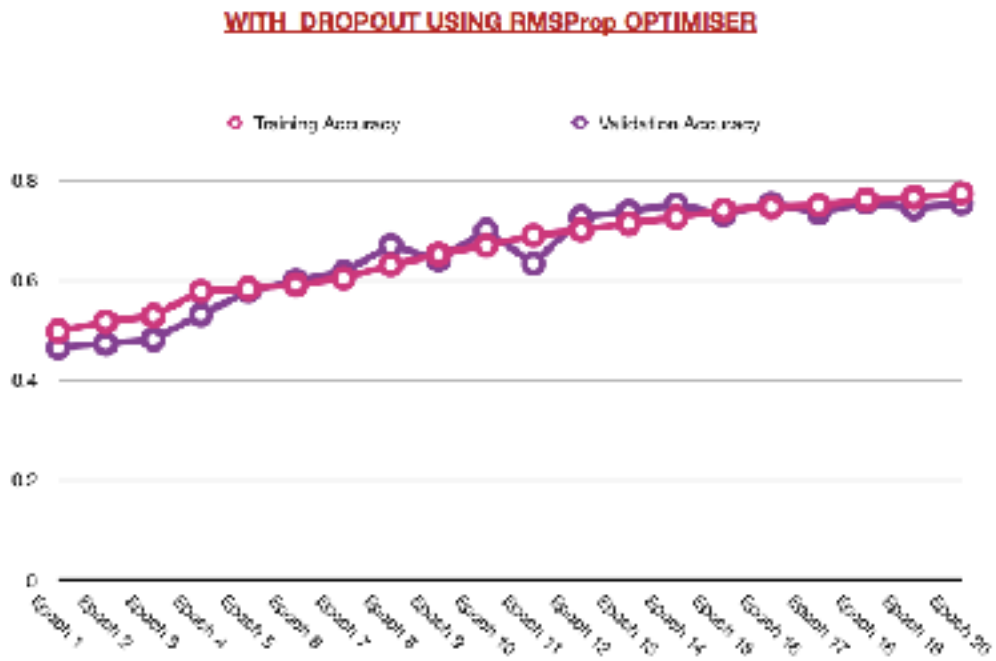


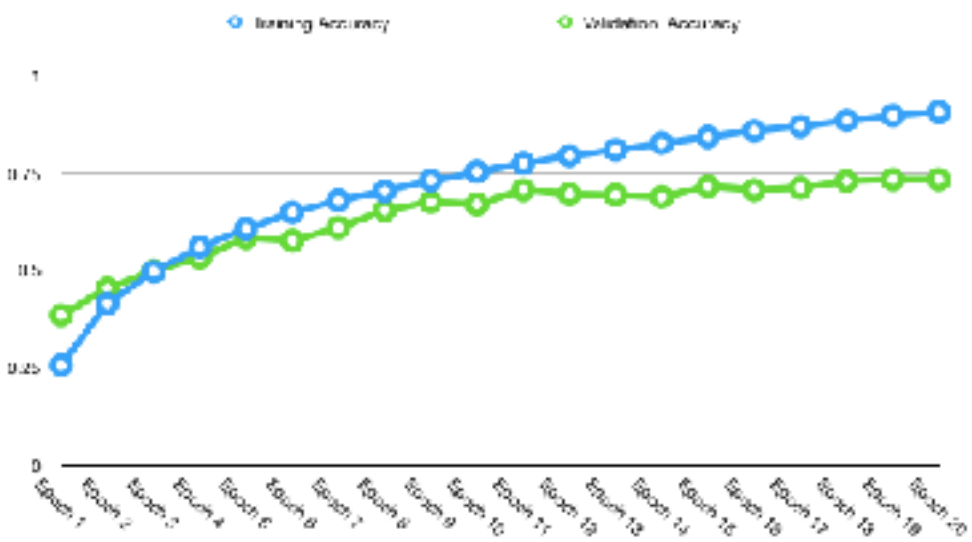
So, we can choose convolution filters of depth 3 (e.g. 3x3x3 or 5x5x3 or 7x7x3 etc.). We are picking a convolution filter of size 3x3x3.

2. **Pooling Layer:**- Pooling layer is mostly used immediately after the convolutional layer to reduce the spatial size (only width and height, not depth). This reduces the number of parameters, hence computation is reduced. Using fewer parameters avoids overfitting. We are using **Max pooling** where we take a filter of size (2*2) having stride of 2.
These defined parameters will now reduce input size to half as shown below.



3.**Dropout Layer:-** Dropout is a regularisation technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. The term "dropout" refers to dropping out units (both hidden and visible) in a neural networks.
More technically, at each training stage, individual nodes are either dropped out of the net with probability 1-p or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed.
Here we are using a dropout ratio of 0.25,0.25 and 0.5 at subsequent layers.





4. <u>Dense Layer</u>:- A linear operation in which every input is connected to every output by a weight .Generally followed by a non-linear activation function.

output = activation(dot(input, kernel) + bias)
Our kernel size is (3*3).

5. **<u>Flattening Layer:-</u>** The Output of a convolutional layer is a multi-dimensional Tensor. We want to convert this into a one-dimensional tensor. This is done in the Flattening layer. We simply use the reshape operation to create a single dimensional tensor by multiplying length, breadth and height.

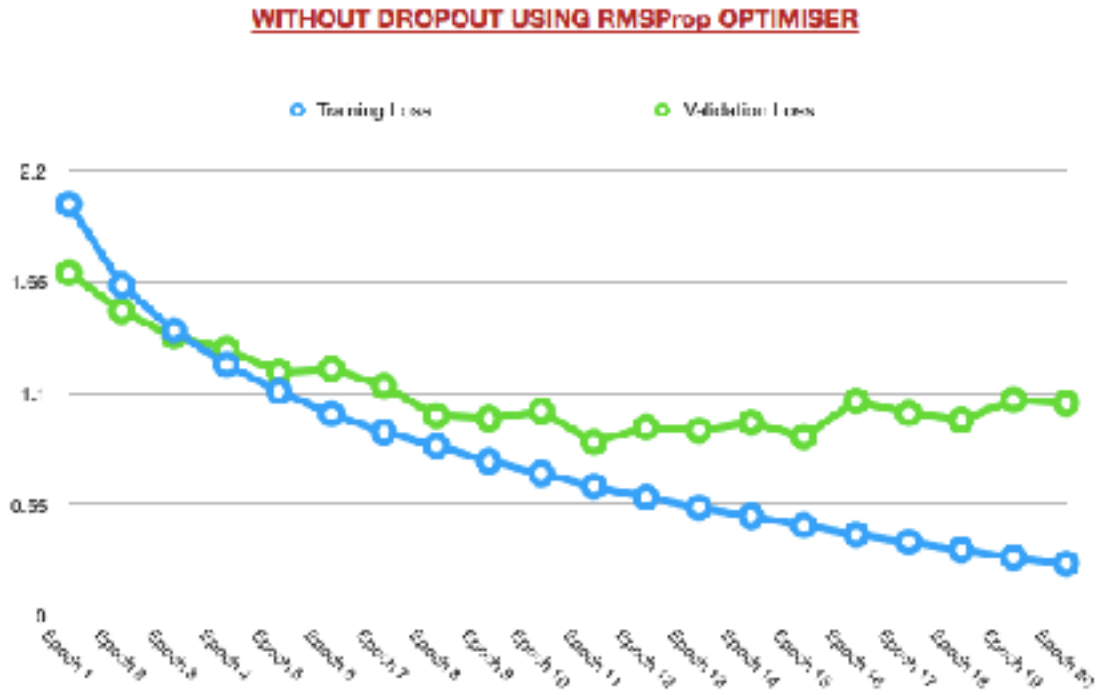
<u>(C). Loss Function :-</u>

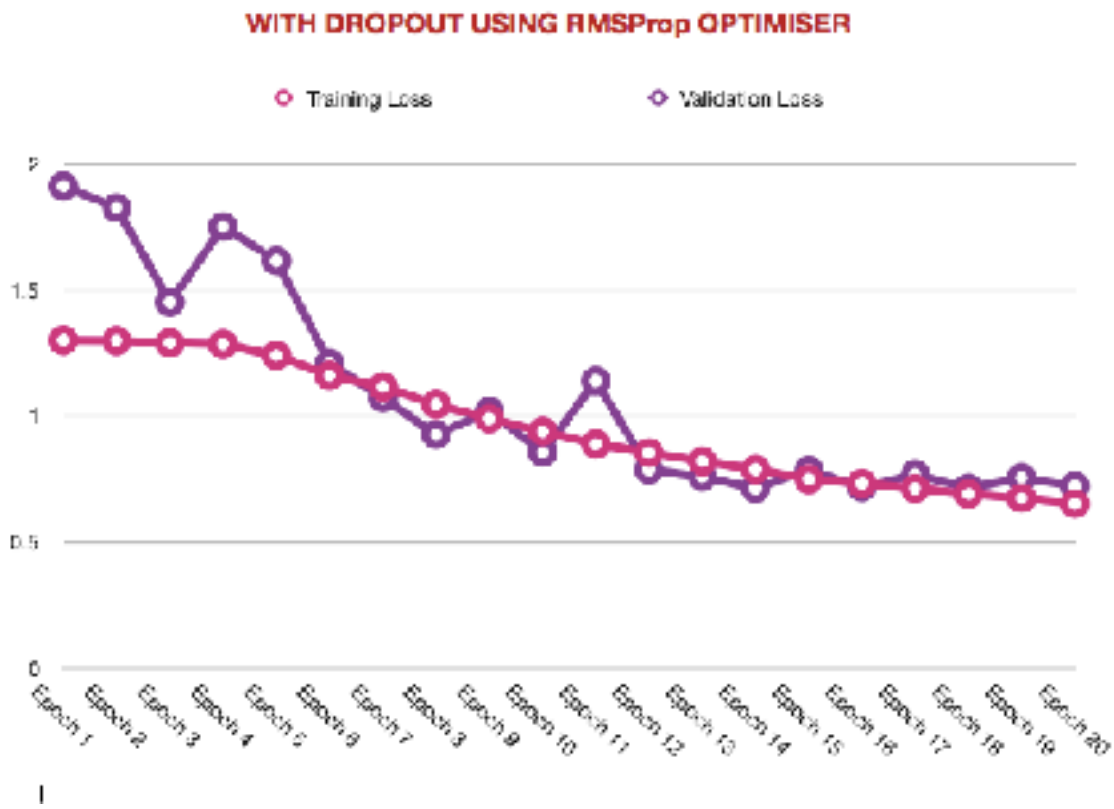
<u>Available loss functions are:-</u>mean_squared_error, mean_absolute_error,mean_absolute_percentage_error, mean_square_logarithmic_error, squared_hinge, hinge,

categorical_hinge, log cosh, categorical_crossentropy, sparse_categorical_crossentropy, poisson.

Parameters to be passed are:- (y_true, y_pred)
where y_true gives true labels and y_pred gives predicted labels.

We are using **categorical_crossentropy(y_true, y_pred)**. So, loss targets must be in categorical format. Example:- If we have 10 classes, target for each sample should be 10-dimensional vector that is all zeroes except for a 1 at index corresponding to class of sam





(D). Optimizers:-

Different optimisers available are:- Adadelta, Adagrad, Adam, Conjugate gradients, Gradient descent, Stochastic gradient descent, Momentum, Nesterov Momentum, Newton's Method and RMSProp.

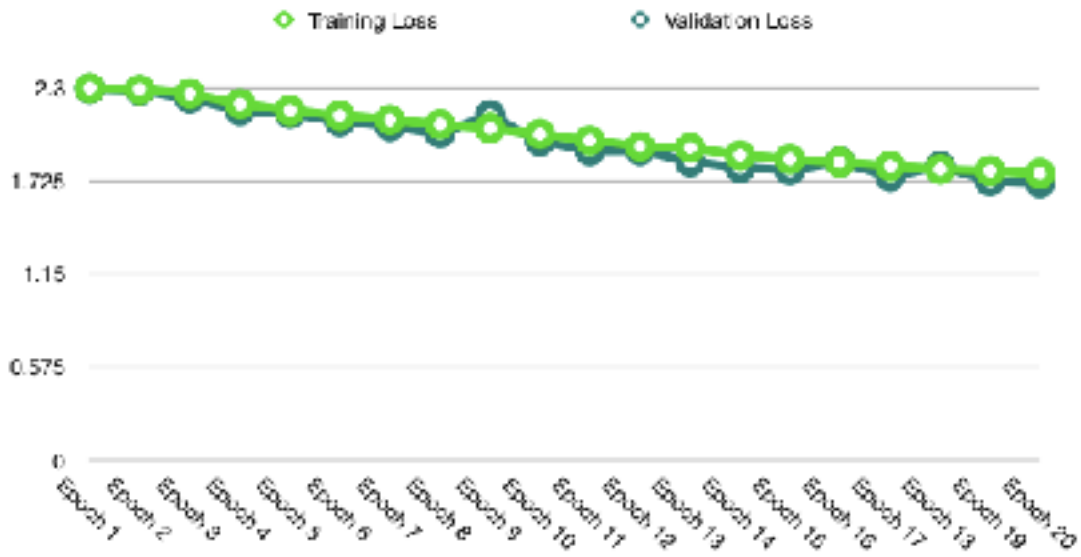
RMSProp:- Here we keep a moving average of squared gradient for each weight.

So dividing gradient by root(Mean_Square(w,t)) makes learning work much better.

It is recommended to leave parameters to this optimiser at their default values (except learning rate, which can be freely tuned.)

This optimiser is a good choice for recurrent neural networks.

WITH DROPOUT USING SGD OPTIMISER



References:-

1. Ciresan, Dan C., et al. "Flexible, high performance convolutional neural networks for image classification." IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Vol. 22. No. 1. 2011.

2. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012

Github Link:-

https://github.com/suyashsangwan/deep-learning