Pattern Recognition and Machine Learning Project Topic: comparative classification of CNN,MLP and SVM-SGD for cifar-10 dataset

M K Laksath (B19CSE045), Himanchal Sharma (B19CSE039), Keshav (B19CSE042)

I. Introduction

MAchine learning is becoming a more integral part of society every day as more and more systems grow to depend upon data and feedback to further optimize or change. One of the most popular forms of data under more than exponential growth is the image/video data. The data containing pixels or pixels per frame in case of videos.The amount of information that can be gained from these pixels even for just a small 32x32 image is tremendous, and how much a set of just 10,000 images can help you recognise,organise or predict is also beyond bounds. In times such as these various models with mind blowing mathematics and tricky practical fixes are becoming the trend, models keep getting better, smarter and faster and so do the methods of making and using them. There are three models designed to categorise a certain kind of object from an image namely: Stochastic Gradient Descent, Multi Layer Perceptron, Convolutional Neural Networks. We attempt to do a comparative study of these three models based on various parameters to judge the best model for predicting on Cifar-10 dataset.

II. Multi-layer Perceptron

Multi layer perceptrons were the predecessors of the present day popular Convolutional Neural Networks. These were the first runs at trying to mimic the structure of the neurons of the brain and impart the model the ability to learn the best representation of the input data and find the most suitable way to relate it to output, in a way to generate a mathematical mapping between the two. MLP is a single layer neural network that is precursor to multi layer more complex models.

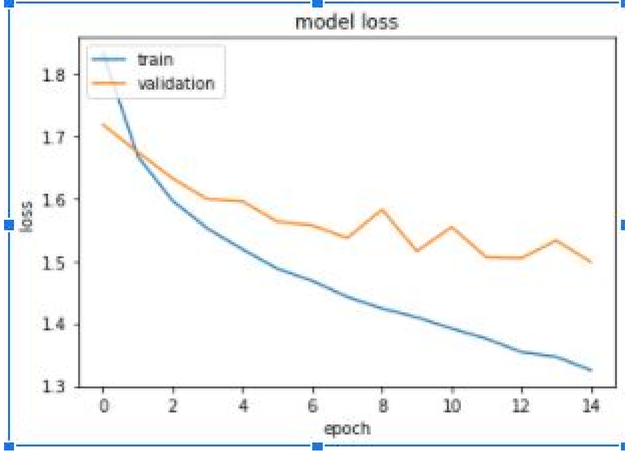
*1) **Methodology**:*

1) Principle: What the MLP tries to achieve is make a network of neurons with multiple layers where each layer contains multiple neurons with their own weights,biases and activation function(each layer has one). The idea is to assign a random set of weights biases and a chosen activation function to neurons in layers and then adjust them throughout the training process to best map them to the output.
2) Specifications: In order to run the Cifar-10 dataset through a neural network a bunch of standard practises were applied first. The feature set was converted into categorical integers representing each class in the dataset. The training and testing feature sets were reshaped from numpy arrays containing pixel info as 32*32*3 where 32*32 were pixel values of the 32*32 image and 3 were the RGB values, into a single 50000,3072 shaped array with all the pixel information and the RGB info is contained in one row of the array not 3arrays within array. This was done to suit the format in which keras(tensorflow library) designed sequential model accepts input.
3) Building: A sequential model was chosen as those are the preferred type of models with plain layers containing exactly one input and output tensor. Two 'dense' form of layers were added to the network. Dense layers are the most common form of layers which perform the following function on the input: output = activation(dot(input, kernel) + bias) And returns the output.
4) Here for the first two layers, the number of neurons were chosen to be 256 with the activation function chosen as the most common type "Rectified Linear Unit" function which "is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero." The first layer also contains the dimensions of the input the layer will receive which will be (Batch-size,3072) where batch size will be given at the time of running. The final layer contains 10 neurons and the activation function is chosen as softmax because "softmax is used as the activation function for multi-class classification problems where class membership is required on more than two class labels." and our final output has to classify into 9 types.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

5) Softmax Activation:
   Next we've defined the SGD optimizer as the optimizer for our network with the standard learning rate of 0.01 and momentum of 0.9. We have chosen nesterov to be true(What is Nesterov: "Much like Adam is essentially RMSprop with momentum, Nadam is Adam RMSprop with Nesterov momentum.") The model was compiled to use SGD as the

optimizer with categorical-cross-entropy as the loss function to be minimised and val-accuracy to be maximized. The model was trained for 15epochs i.e ran over the entire dataset 15 times and the chosen loss over epochs was plotted as follows:



A. *Conclusion*

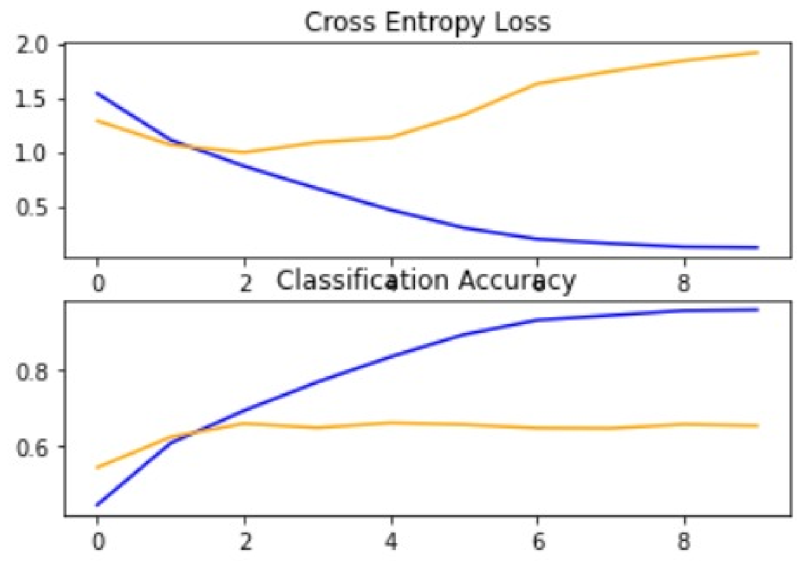
The final loss and accuracy score over the test set were found as : [1.4990483522415161, 0.4756999909877777]

III. Convolutional Neural Networks: CNNs

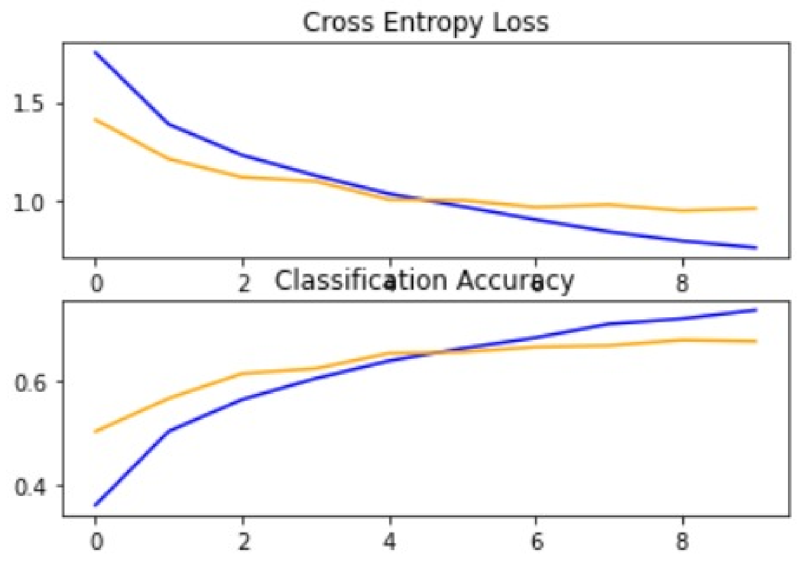
Convolutional Neural Networks are the backbones of modern day image processing Machine and Deep learning models. CNNs are the best attempt by far made at achieving complexity of the neurons in the brain by looking at various aspects of the image and getting stimuli from it to justify how much importance that part of the image(or a frame in case of videos) is to classify what we're seeing.

1) ***Methodology****:*

1) Principal: What CNNs try to do through some methods like running kernels(matrix dot products) over image pixels(matrix) is to extract some important information from the image like edges and shapes and then assign neural weights and biases to every such feature and make a neuron responsible for it, which based on chosen activation function will decide if the neuron will fire up every time the feature is is responsible for is detected. Using such combinations of neurons across layers CNNs assign probabilities to what the object in the image could be.
2) In order to build the CNN similar preprocessing as the above mentioned MLP model was done.
3) Building: A sequential model type was chosen. A 2D convolution layer of the shape 32,3,3 was applied on the input layer with Relu as the chosen activation function and in the first layer tha shape of the input was given as it is in the from 32*32*3. A second convolutional layer was added with the same Relu activation function and similar shape. A max pooling of the above convolutional layer was done. What Max pooling achieves is reduce the noisy activation values while performing a dimensionality reduction of the convolution to extract maximum from it while applying least computational power. Next the the output of this second layer was flattened so it can be sent to the next layer for implementing activation function and assigning weights and biases Next layer is defined as 256 neurons with ReLU as their activation function and the output of the layer above as there input The final layer is assigned with 10 neurons and softmax as the activation function for reasons same as MLP. SGD was chosen as the optimizer with the standard learning rate,decay rate and momentum with nesterov set to True. The model was compiled to minimize the categorical loss entropy loss function while trying to maximize the val accuracy. The model was ran for 10epochs with the validation split of 0.2 to determine the validation accuracy and the following accuracy was received on the test set: 64.460 The plot of loss and validation loss over each epoch as well as the plot of accuracy and validation accuracy:

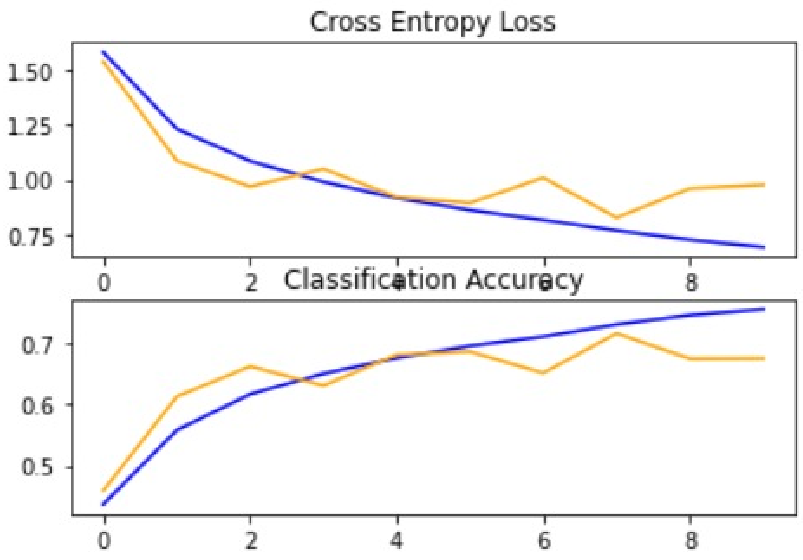


4) Improving: In an effort to improve the accuracy of the model two drop out layers were added after the max pooling layer and the first neuron assignment layer. What dropout does is “that it will randomly drop nodes out of the network. It has a regularizing effect as the remaining nodes must adapt to pick-up the slack of the removed nodes.” “The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting”. The model was again ran with the similar optimizer,epochs,loss function and batch size and following accuracy was received on the test set: 67.040
5) The plot of loss and validation loss over each epoch as well as the plot of accuracy and validation accuracy:



6) In an effort to further optimize the model batch normalisation was done after the second convolutional layer and before the second dropout layer. Normalization is done because “One potential purpose behind this trouble is the distribution of the inputs to layers somewhere down in the network may change after each mini-batch when the weights are refreshed. This can make the learning algorithm always pursue a moving target. This adjustment in the distribution of inputs to layers in the network has alluded to the specialized name internal covariate shift.The challenge is that the model is refreshed layer-by-layer in reverse from the output to the input utilizing an estimate of error that accept the weights in the layers preceding the current layer are fixed.”
7) The model was again ran with the similar optimizer,epochs,loss function and batch size and following accuracy was received on the test set: 65.810
8) As we can see the accuracy was decreased but primarily we believe it is because of the lack of computational power that we couldn’t run it for more than 10 epochs, generally to generate anything concrete atleast 50 runs are required but that would cost us 4 hours per run and we wouldn’t be able to do any experimenting at all with the parameters.

9) The plot of loss and validation loss over each epoch as well as the plot of accuracy and validation accuracy:



A. Conclusion

As we can see the accuracy was decreased but primarily we believe it is because of the lack of computational power that we couldn't run it for more than 10 epochs, generally to generate anything concrete atleast 50 runs are required but that would cost us 4 hours per run and we wouldn't be able to do any experimenting at all with the parameters. Highest accuracy obtained on test data while training for different parameters was 67.040

IV. Support Vector Machines: SVM

SVM aims to draw a decision boundary through linearly separable classes. When there are many lines possible to draw and differentiate between classes, We take the line that brings out the maximum margin between them.

Support vectors are just points that lie nearest to the decision boundaries. Support vectors are the ones that maximize the decision boundary margin.

Hard margins are the ones that allow no points to fall inside the decision margin. Whereas, soft margins allow some slack.That is, they allow some points to fall inside the margin.This is useful as it allows some class overlaps which could be needed in free cases. "hard margin" because there are no points that fall inside the decision margin. In many cases of SVM classification, it makes more sense to use a "soft margin", which would give "slack" to some points and allow them to encroach on the dividing region

$$\mathrm{h}(\beta) = \frac{1}{2}\|\beta\|^2 + C\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\beta x_i)\right)\right]$$

Here , a smaller value of C will lead to a softer margin .This formula is called hinge loss function.

Upon minimizing this function, we get a set of betas that look similar to linear regression function.

$$\hat{y}_i = \begin{cases} -1, & \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} < 0 \\ 1, & \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} \geq 0 \end{cases}$$

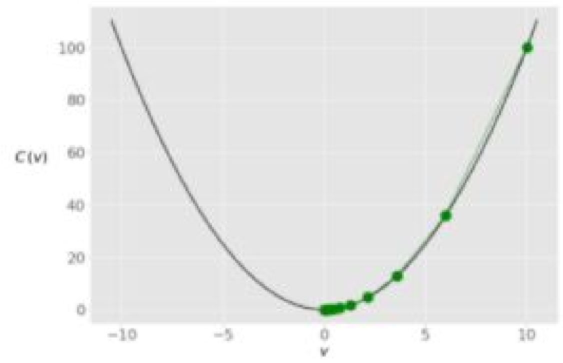
Prediction are made in the following way:

*1) **Stochastic Gradient Descent**:* Stochastic Gradient Descent also known as SGD is known for its solving of optimization problems.

Intuition behind gradient descent:

Let us take a large bowl which will be our cost function. Any random position on the surface of the bowl is the cost of the current value. The bottom most point of the bowl is where the cost is least and hence contains the best set of coefficients. Our target is to try different values of coefficients and to minimize the cost and reach the bottom of the bowl.

The figure below shows the movement of the solution through the iterations:



Stochastic gradient descent algorithms are a modification of gradient descent. In stochastic gradient descent, we calculate the gradient using just a random small part of the observations instead of all of them. A gradient is a vector that contains every partial derivative of a function.

SGD works by initializing the set of coefficients with random values and then calculating the gradient of the loss function at the given point in a dataset.Then we update the values of coefficients by taking some defined step from the direction opposite to the direction of the gradient. This way, the algorithm updates itself iteratively such that it moves away from the steepest hill (That is , from the maximized loss function).This way leads to moving towards the minimum of loss function.

SGD works by initializing a set of coefficients with random values, calculating the gradient of the loss function at a given point in the dataset, and updating those coefficients by taking a “step” of a defined size (given by the parameter) in the opposite direction of the gradient. The algorithm iteratively updates the coefficients such that they are moving opposite the direction of steepest ascent (away from the maximum of the loss function) and toward the minimum, approximating a solution for the optimization problem. The algorithm follows this general form:

```
$\beta$ = [random coefficient values]

for *epoch* in range *n_epochs*:
    for *instance* in range *X_length*:
        $x_i$ = $X$[random index]
        $y_i$ = $y$[random index]

        *gradient* = $\nabla_\beta$ f($\beta, y_i, x_i$)
        $\beta$ = $\beta$ - $\eta$ * *gradient*
```

Beta is the learning rate. N-epochs is no of iterations over the full dataset. f(, yi, xi) -¿ loss function. gradient -¿collection of partial derivatives.

$$\mathrm{h}(\beta) = \frac{1}{2}\|\beta\|^2 + C\left[\frac{1}{n}\sum_{i=1}^{n}\max\big(0, 1 - y_i(\beta x_i)\big)\right]$$

$$\downarrow$$

$$\mathrm{h}(\beta) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{2}\|\beta\|^2 + C * \max\big(0, 1 - y_i(\beta x_i)\big)\right]$$

Using SVM with SGD:

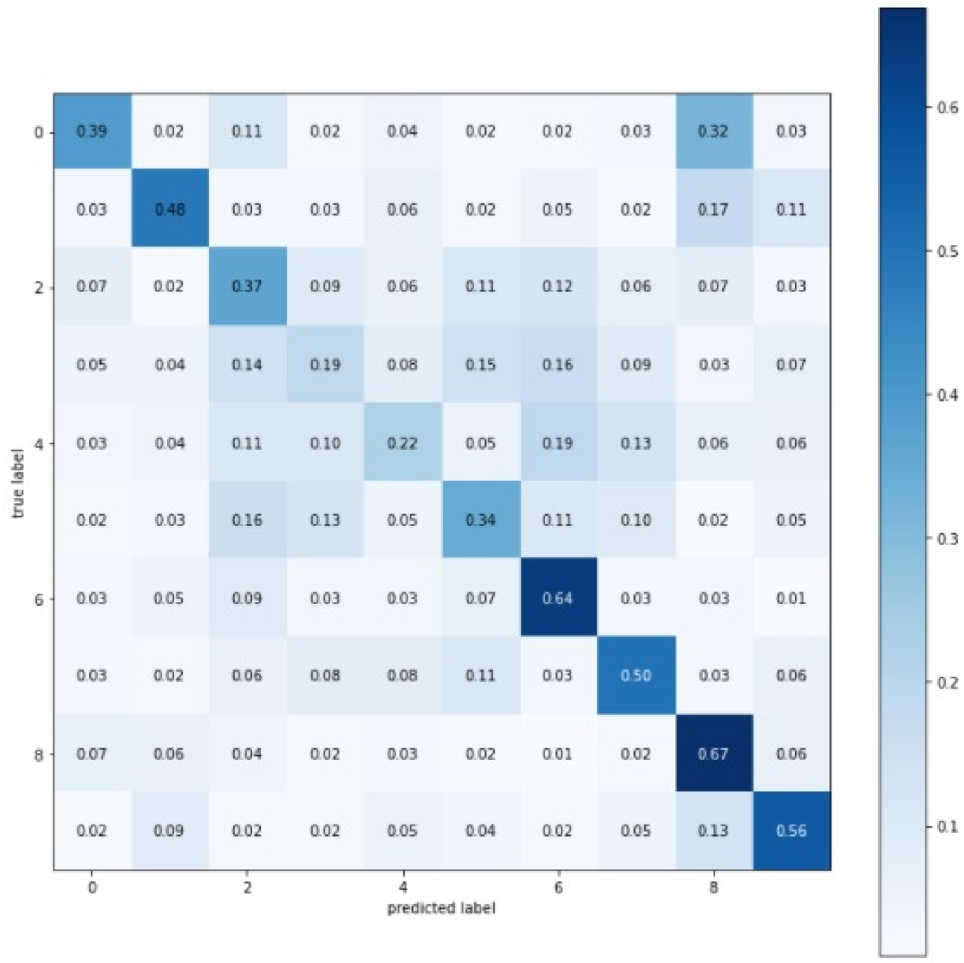
*2) **Methodology:***

1) TO use SVM with SGD, we need to find the hinge loss function’s gradient.Once we find that, we can substitute that value on The previous formula , we will be able to update the learning rate and minimize the loss function.
2) We have used cifar 10 dataset to demonstrate image classification between 10 images using SGD with SVM in the sklearn library.
   SGD in sklearn uses SVM with “ovo” by default. It also uses One vs rest by default. It is important to note that SGD is sensitive to feature scaling.

3) Initially, The training data received was in 32x32x3 (rgb format). It was further converted into a grayscale image using cv2. Hog image is a better way to preprocess but for that we need to convert it into grayscale first. Now , the image was converted into hog format. In here , there are several hyper-parameters such as pixels-per-cell, Cells-per-block,orientations that we manually have to alter for the improvement of the model. Before fitting the data, there is one more step that was required and that is to scalify the data. Upon this step, we conclude the preprocessing steps. Now instead of using gridsearchcv to choose the optimal parameters, we have written down 10 cases manually as we had a good idea about the optimal hyperparameters.
4) In an effort to further optimize the model batch normalisation was done after the second convolutional layer and before the second dropout layer. Normalization is done because “One potential purpose behind this trouble is the distribution of the inputs to layers somewhere down in the network may change after each mini-batch when the weights are refreshed. This can make the learning algorithm always pursue a moving target. This adjustment in the distribution of inputs to layers in the network has alluded to the specialized name internal covariate shift.The challenge is that the model is refreshed layer-by-layer in reverse from the output to the input utilizing an estimate of error that accept the weights in the layers preceding the current layer are fixed.”
5) The model was again ran with the similar optimizer,epochs,loss function and batch size and following accuracy was received on the test set: 65.810
6) As we can see the accuracy was decreased but primarily we believe it is because of the lack of computational power that we couldn’t run it for more than 10 epochs, generally to generate anything concrete atleast 50 runs are required but that would cost us 4 hours per run and we wouldn’t be able to do any experimenting at all with the parameters.
7) The plot of loss and validation loss over each epoch as well as the plot of accuracy and validation accuracy:

A. Conclusion

1) The fifth case was the best one which had an alpha value of about 0.0001 with hinge loss and l2 penalty. The accuracy score is: 0.4364 The precision score is: 0.4364 The recall score is: 0.4364 The f1 score is: 0.42721575562168485
2) A confusion matrix was plotted to identify if the majority of the classes were correctly classified or not.



0	0.39	0.02	0.11	0.02	0.04	0.02	0.02	0.03	0.32	0.03
	0.03	0.48	0.03	0.03	0.06	0.02	0.05	0.02	0.17	0.11
2	0.07	0.02	0.37	0.09	0.06	0.11	0.12	0.06	0.07	0.03
	0.05	0.04	0.14	0.19	0.08	0.15	0.16	0.09	0.03	0.07
4	0.03	0.04	0.11	0.10	0.22	0.05	0.19	0.13	0.06	0.06
	0.02	0.03	0.16	0.13	0.05	0.34	0.11	0.10	0.02	0.05
6	0.03	0.05	0.09	0.03	0.03	0.07	0.64	0.03	0.03	0.01
	0.03	0.02	0.06	0.08	0.08	0.11	0.03	0.50	0.03	0.06
8	0.07	0.06	0.04	0.02	0.03	0.02	0.01	0.02	0.67	0.06
	0.02	0.09	0.02	0.02	0.05	0.04	0.02	0.05	0.13	0.56

From the matrix, we can notice that in all classes that (predicted labels,correctly labelled) point has the highest percentage.
This concludes the model.

From the matrix, we can notice that in all classes that (predicted labels,correctly labelled) point has the highest percentage. This concludes the model.

V. Contributions

1) Himanchal Sharma: Studying and implementation of CNN based approach.
2) Keshav : Multi-layer Perceptron based implementation and documentation.
3) M K Laksath : Support Vector Machine including SGD with implementation, analysis and report.