COMP 551
Mini Project 3: Image Data Classification

Adharsh Mahesh Kumaar, Manoj Krishna, Nithilasaravanan Kuppan *McGill University*

***Abstract*—In this final Mini Project of the course, the performance of two Deep learning algorithms were compared with an objective to classify image data; specifically the CIFAR - 10 data set. The input data was augmented to support better predictions and fed into the models. Along with various regularization techniques, the models were also iteratively tested with different types of optimizers as well as their performance was noted with respect to varying model parameters. As expected, CNN proved to be the superior technique for image classification. Morever, additional experiments were performed with both the models, one of which gives us a peek into how machines visualize the world.**

I. INTRODUCTION

Multilayer perceptron, colloquially referred to as the *vanilla* neural network is one of the older classes of the umbrella, collectively known as artificial neural networks. It has a minimum of an input, hidden and an output layer. Each node is a neuron that uses a nonlinear activation function (except the input nodes). It has all the characteristics of fully connected layers, where each perceptron is connected to one another. There are a number of disadvantages to MLP, making it an insufficient tool for modern computer vision tasks - the very high number of parameters, which is both inefficient and redundant and its disregard for spatial information.
Convolutional Neural Networks are one of the best classes of solution available to image and video processing related problems. The process revolves around the creation of feature maps, with the help of kernels or filters. In contrast to MLP, CNN is able to capture the spatial features from an image; additionally, it also incorporates parameter sharing i.e., a single filter is applied across different parts of the image to engender a feature map [1,2,3,4].

A. The Task

The primary objective of this project is to create models to classify image data and compare their performance; specifically two models a) a Multilayer Perceptron and b) a Convolutional Neural Network. The MLP was implemented from scratch but the *base code* for the CNN was borrowed from the PyTorch tutorial [5].

B. Related Work

Multilayer Perceptrons have always been useful in academia and research because of their stochastic problem solving approach. They were a popular choice for speech recognition, as illustrated by the work presented by *A. Ahad, et al* in which the objective was to recognize Urdu digits from zero to nine from a mono speaker database. The team preprocessed the speech data with FFT and a particular set of filters before being fed into an fully connected 3 layer MLP with back-propagation error algorithm, for classification [6]. A similar, but more complex research was carried out by *N. Morgan & H. Bourlard* in their paper on continuous speech recognition. They used MLP with an integrated Hidden Markov Model to contrast over performance with simple maximum-likelihood probabilities [7].

MLP has also been used by researchers (*A. Patry & P. Langlais*) in Machine Translation to estimate the probability that a target word appears in the translation of a given input sentence [8].

In correlation to this project report, MLP has been extensively used for image and video processing / classification [9]. *Y. Hara et al.* used it to classify ground terrain types from fully polarimetric synthetic aperture radar (SAR) images using images from radar [10]. *U. Orhan et al.* used a MLPNN based model as a diagnostic decision support mechanism in the treatment for epilepsy [11].

Convolutional Neural Networks are responsible for major breakthroughs in images classification and presently are at the core of almost all computer vision based applications in the world. It is evident from the multitude of research areas that have leveraged CNN to achieve results. *P.Y. Simard et al.* at Microsoft established best practices on CNN's use for visual document analysis and established its dominance by testing it out on the MNIST set of English digit images [12]. *M. Liang & X. Hu* used a variation of CNN for object detection and tested it on the benchmark datasets - including CIFAR 10 [13].

CNNs were also used for face recognition where ConvNet was combined with local image sampling, and a self-organizing map (SOM) neural network by *S. Lawrence et al.* [14]. They were also used for semantic modelling of sentences, as demonstrated by *N. Kalchbrenner et al.* in their work that utilized Dynamic Convolutional Neural Network via dynamic k-Max Pooling [15].

In the following sections, we will briefly discuss the dataset used for this modelling exercise and deep dive into the actual model and their corresponding results.

II. Dataset

The CIFAR-10 is an established computer-vision dataset consisting of 60000 32×32 colour images in 10 classes, with 6000 images per class. There are 50000 training and 10000 test images in the dataset. These splits were used as is for this exercise. It was collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton [16]. The following image displays all the classes along with a few sample images in them.

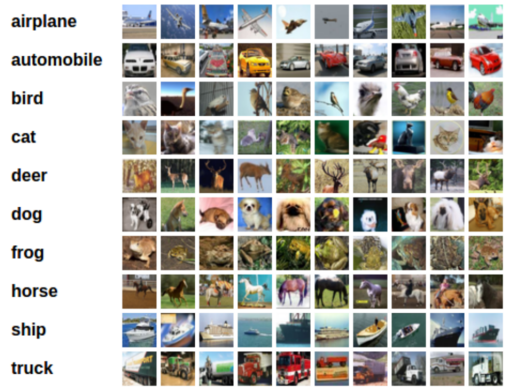

Figure 1: All classes in the dataset, with example images

A. Data Preprocessing

In MLP the data is flattened into a 3072 dimensional array and the data standardized to achieve 0 mean and unit standard deviation. Similarly the labels are converted to a one dimensional array that holds the categorical value of the output. This dataset has 10 categorical output. It is later one hot encoded to get a 10 dimensional array.

And in CNN, before the data was fed into the multi layered network, the input data was augmented to artificially enlarge the training set. Data augmentation assists in making the models more robust and makes it possible to achieve higher accuracy without the need for a large number of training example by preventing over-fitting.

RandomHorizontalFlip() will horizontally flip the image with a 0.5 probability. *RandomCrop()* crops and pads the images as per the given parameters. It is imperative to perform these steps before converting the images to tensor because these functions apply only to images. After this, the training set was normalized based on the mean and std dev over all the three channels.

1) Data Loading: For the MLP based model, 20% of training data was separated for validation set and remaining data was used to train the model.
In case of the CNN model, the data was loaded in batches of 4, and a separate validation set was set aside *(20% of the training data)* for validation testing.

III. Model Description

In this section we shall briefly describe the models and the associated parameters before jumping on to the results section

A. Multilayer Perceptron

Multilayer perceptron is a neural network with one or more hidden layers. Each layer has one or more neurons linking the previous and next layer. In this model, all layers are densely connected, i.e, each neuron in a layer is connected to all the neurons in the previous layer. The choice of width and depth of the neuron determines the performance of the model. After diligent analysis and experiments, a 2 hidden layer network of 512 parameters each is decided to be a good choice for this application. All weights of the neural networks are randomly initialized with 0 mean and $1/\sqrt{dim}$ standard deviation for every layer. Also, the biases of the neural network are randomly initialized with mean 0 and standard deviation 1. The output labels are one hot encoded.

The dataset is shuffled and categorized into mini batches of size 50. Stochastic Gradient Descent is performed for each mini batch and 20 epochs are run to train the model, optimizing the cost function.

Backpropogation algorithm is used in this case to compute the gradient of cost function for every mini batch and the weights are updated in each iteration. Every layer (excluding the output) uses ReLU as activation function and the output layer is activated by a softmax function. The gradient descent is augmented with L2 Regularization and it efficiently decreases variance of the model.

Finally, the trained model is then used to compute the training data accuracy and validation set accuracy. Hyper parameters such as the learning rate and L2 regularization parameter is tuned by evaluating the trained model against the validation set. The training accuracy, validation accuracy and the cost is computed and stored for evaluation.

B. Convolutional Neural Network

The CNN class consists of three blocks of Convolutional layers, with a kernel size of 3×3 and *ReLu* as the activation function for all the layers. The padding has been set to 1 on all the layers to avoid losing details from the original input images. The first block of CNN consists of two layers - the first of which has 3 *in channels* and 32 *out channels* and the other has 32 *in channels* and 64 *out channels*. It was only logical to select the most common variation of Max Pooling with a kernel size of 2×2 and stride of 2 which down-sampled the input to facilitate assumptions about the features.

Apart from this, the team chose Batch Normalization, a regularization technique that is used in maintaining mean activation close to 0 and a standard deviation close to 1 by applying appropriate transformation.

The second block of CNN also consists of 2 layers with the first layer consisting of 64 *in channels* and 128 *out channels*. The second layer has 128 *in channels* and 256 *out channels*. Max pooling with a kernel size of 2×2 and a stride of 2 along with Batch Normalization is utilized. Another regularization technique known as Drop out is used which randomly drops points with a probability less than 0.05. The 2 layers in the 3rd block of CNN have 128 *in channels*, 256 *out channels*, and 256 *in & out channels* respectively along with Batch normalization and Max pooling with a stride of 2.

The outputs are flattened and are then passed on to 2 fully connected layers to give 10 outputs one belonging to each class. The model is run for 15 epochs, experimenting with different optimizers which are discussed in the next section. A confusion matrix has been plotted to show that the model clearly distinguishes each of the classes.

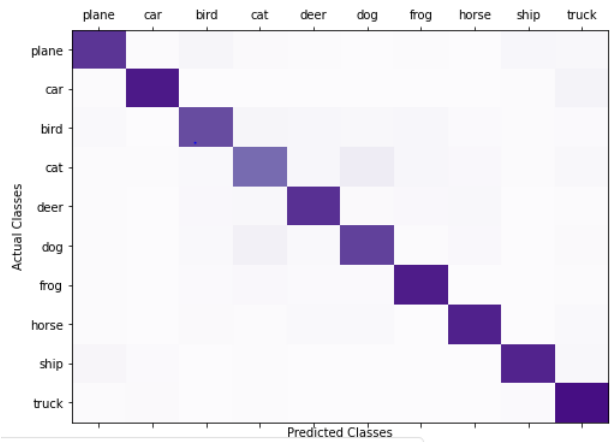

Figure 2: Confusion matrix for CNN

IV. Experiments & Results

A. Training and Test performance

The table below compares the accuracy of the both MLP and CNN. The reported accuracy is obtained after tuning the hyper parameters and selecting the best model.

Table I: Model performance comparison

	MLP	**CNN**
Training Accuracy	75.768%	88.782%
Validation Accuracy	52.990%	84.940%
Test Accuracy	52.540%	84.670%
Optimizer used	SGD with L2	SGD with momentum

From table I we can observe that, CNN gives a much better performance on the test data than MLP. We can also see that there is a very little difference in training and validation performance which shows that there is no over fitting.

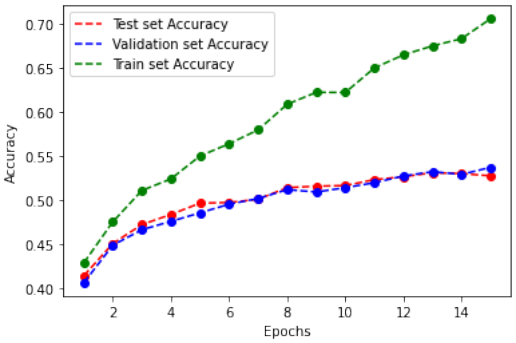

Figure 3: Accuracy of MLP vs No of epochs

From figure 3 it is evident that, the model begins to overfit gradually after 5 epochs and MLP suffers from high variance if epochs are high. L2 regularization helps reducing the variance but it is less effective in achieving a good bias variance trade-off.

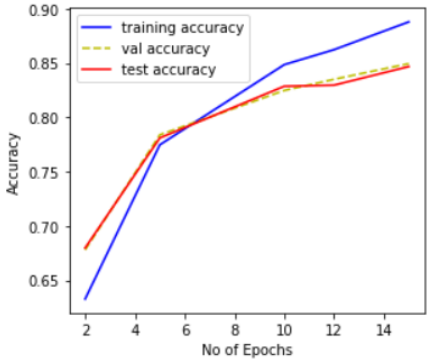

Figure 4: Accuracy of CNN vs No of epochs

From figure 4 we can observe that the accuracy of CNN in validation set is approximately same as the test set for the chosen best model (*SGD with momentum = 0.9*). This shows that the generalization error on the validation set can be well approximated on the test set.

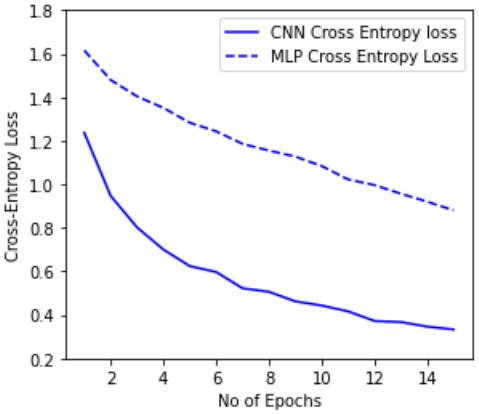

Figure 5: Cross Entropy loss vs No of epochs

Figure 5 shows that the cross-entropy loss of CNN is much lesser than in MLP. This proves the fact that CNN performs

much better than MLP on the test data set. This fact is also evident from table I.

B. Optimizers

For MLP, Mini Batch Stochastic Gradient Descent (SGD) with a batch size of 50 is used to train the model. Along with that, L2 regularization is used.

For CNN, different optimizers such as SGD, SGD with momentum, ADAM and AdaGrad have been used and their training accuracy, validation accuracy vs no of epochs have been plotted. From figure 6, we can observe that the validation accuracy is maximum for SGD with momentum and that is taken as the final model to predict the accuracy of test data.

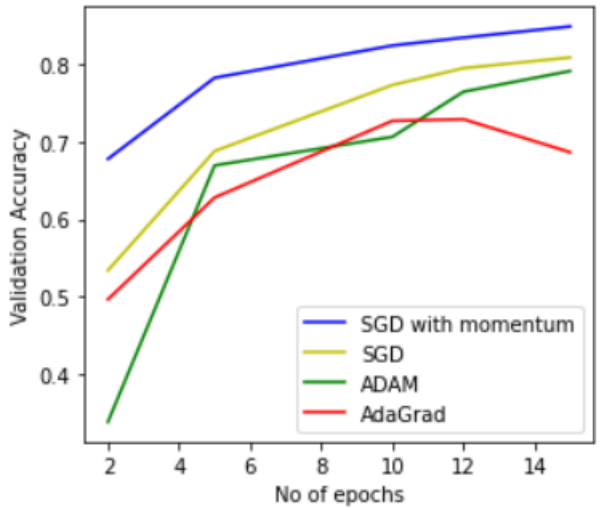

Figure 6: Optimizers validation accuracy vs No of epochs

C. Visualization of Filters and Classes

The idea of this experiment is inspired by how machines visualize the world [17]. This experiment shows how CNN sees the images we feed them.

The first layers primarily encode the direction and the color from the images which get combined into a basic grid texture. The kernels get more incisive and complex as they start accumulating different information from the images as the algorithm proceeds. In the penultimate set of layers, the textures and patterns similar to the ones in the images can be seen forming.

Figure 7 shows the images of filter 9, 10, 11 and 12 out of the 32 filters present in one layer of CNN.

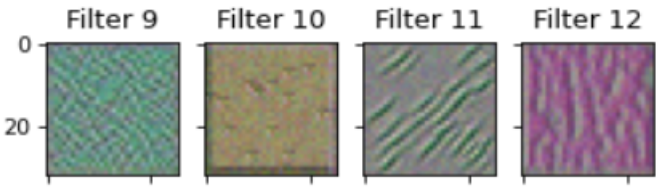

Figure 7: Filter images

Figure 8 shows how CNN visualizes the images of each of the classes in the data set. If these images are fed as inputs to CNN then the output accuracy will be approximately 100%. From this experiment, we can see how the model visualizes the images in a different way to that of humans.

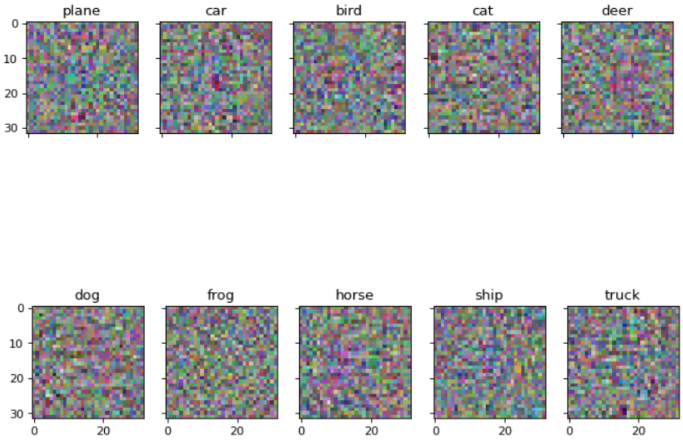

Figure 8: Images of different classes

D. Variation in Layer Sizes

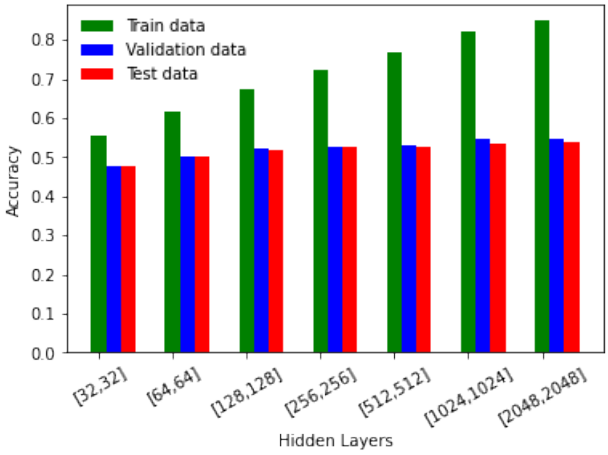

Figure 9: MLP Hidden Layer width vs No of epochs

For MLP, different layer sizes were experimented to test the variation in accuracy. When the layer depth is increased, the model performance on test data increases, but computation time is drastically increased. Also, it was evident that, when the width of the layer is increased, it increases the training accuracy but not the testing accuracy. This clearly explains that the model is over fit and suffers from high variance. The figure 9 illustrates the explained behaviour.

E. Variation in Activation function

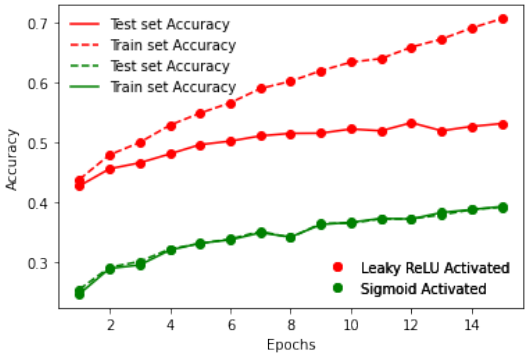

Figure 10: MLP Sigmoid vs Leaky ReLU activation

For MLP, three activation functions are experimented with, namely ReLU, Leaky ReLU and Sigmoid function. Evaluation of results showed ReLU is the best pick among the three. Leaky ReLU performed on par with ReLU but performance of sigmoid function was really poor. The comparison of Leaky ReLU and Sigmoid function can be seen in figure 10.

V. Discussion and Conclusion

The primary motive of this Mini Project was to understand, implement and test two complex models on Image classification. Even though the team was already aware that *Convolutional Neural Networks* would emerge as the superior algorithm, it is important to understand the workings of *Multilayer Perceptron*.

- For both the models, data was preprocessed before being fed into the classification models for MLP it was essential to flatten, standardize and finally one hot encoded and for CNN the data was augmented and normalized to prevent over-fitting
- For both the models, the Generalization error estimated on the validation dataset, which was collected from the training dataset (*20% split*)
- Mini batch SGD with L2 and SGD with momentum was selected as the best optimizers for MLP and CNN, respectively

From the results that the team obtained, it was clear that CNN (*with a best validation accuracy of 84.9% & a similar test accuracy of 84.7%*) outperforms MLP (*with a best validation accuracy of 53.0% & again a very similar test accuracy of 52.5%*) by a sizeable margin.

Apart from comparing the test and train performances of both the models, the team also tried out a couple of experiments. Different optimizers were tested to ensure that the model utilizes the best suited one for the final model. Additionally, inspired by the article written by *F. Chollet* in *The Keras Blog* the team tried exploring the machine side of ConvNet filters. The team also did an interesting experiment to see how the MLP's performance varies with a change in the layer sizes and activation function.

In order to contrast our model with the ones that achieved the best performance on CIFAR - 10 dataset, the team read about a research published by *M. Tan & Q.V. Le*, who proposed a novel scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective compound coefficient. The proposed model, known as *EfficientNet-B7*, achieved a staggering 98.9% accuracy [18].

The team strongly agrees that this exercise helped us understand the algorithms incisively, and that the project supplemented the lecture course very well. The team would love to experiment with more intrinsic/ advanced models with the same dataset to further expand their understanding of Deep learning methods for image classification.

VI. Statement of Contributions

The collaborative work of this team involved everyone contributing to both the code and the report. Adharsh worked on the analysis and experiments associated with *Convolutional Neural Networks* and documenting them on the report.

Manoj implemented *Multilayer Perceptron* from scratch and performed analysis by varying model parameters. He too assisted in documenting those results.

Nithilasaravanan modelled the CNN class and handled the overall report formation along with the intro & literature survey.

References

[1] Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. *MIT press.*

[2] Zhang, A., Lipton, Z.C., Li, M. and Smola, A.J., 2019. *Dive into Deep Learning*. Unpublished draft. Retrieved, 3, p.319.

[3] https://github.com/ageron/handson-ml - *Hands on ML*

[4] https://www.youtube.com/channel/UCcIXc5mJsHVYTZR1maL5l9w - *Deeplearning.ai*

[5] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html - *Training a Classifier*

[6] A. Ahad, A. Fayyaz and T. Mehmood, "Speech recognition using multilayer perceptron," *IEEE Students Conference, ISCON '02. Proceedings.*, Lahore, Pakistan, 2002, pp. 103-109 vol.1.

[7] N. Morgan and H. Bourlard, "Continuous speech recognition using multilayer perceptrons with hidden Markov models," *International Conference on Acoustics, Speech, and Signal Processing*, Albuquerque, NM, USA, 1990, pp. 413-416 vol.1.

[8] Patry, A. and Langlais, P., 2009. Prediction of words in statistical machine translation using a multilayer perceptron. *Proceedings of the twelfth Machine Translation Summit (MT Summit XII)*, Ottawa, ON, Canada, pp.101-111.

[9] Khotanzad, A., Chung, C. Application of multi-layer perceptron neural networks to vision problems.*Neural Comput Applic 7*, 249–259 (1998). https://doi.org/10.1007/BF01414886

[10] Y. Hara, R. G. Atkins, S. H. Yueh, R. T. Shin and J. A. Kong, "Application of neural networks to radar image classification," in *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, no. 1, pp. 100-109, Jan. 1994.

[11] Orhan, U., Hekim, M. and Ozer, M., 2011. EEG signals classification using the K-means clustering and a multilayer perceptron neural network model. *Expert Systems with Applications*, 38(10), pp.13475-13481.

[12] Simard, P.Y., Steinkraus, D. and Platt, J.C., 2003, August. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar* (Vol. 3, No. 2003).

[13] Liang, M. and Hu, X., 2015. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3367-3375).

[14] S. Lawrence, C. L. Giles, Ah Chung Tsoi and A. D. Back, "Face recognition: a convolutional neural-network approach," in *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98-113, Jan. 1997.
[15] Kalchbrenner, N., Grefenstette, E. and Blunsom, P., 2014. A convolutional neural network for modelling sentences. *arXiv preprint* arXiv:1404.2188.1997.
[16] http://www.cs.toronto.edu/%7Ekriz/cifar.html - *The CIFAR -10 Dataset*
[17] https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html- *How convolutional neural networks see the world*
[18] Tan, M. and Le, Q.V., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint* arXiv:1905.11946.