# Convulational Neural Networks and simple Multi-Layer Percepton for Image classification

Furaha, Damien
McGill University

Ujjwal, Kumar
McGill University
{furaha.damien, ujjwal.kumar, }@mail.mcgill.ca

## Abstract

**Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain[4]. Several forms of Neural Networks have been modelled for different applications. In this study, we developed a simple multi-layer perceptron neural network(MLP) and a Convolutional Neural Network(CNN) and gauged their performance on classification of the CIFAR10 image dataset. By developing the neural networks from scratch, we were able to vary several parameters of the Networks. We tried different activation functions and observed that for both models the Rectified Linear Unit (ReLU) activation function performed better than others. We also observed that the performance of the network varied with the number of layers. Overall, we observed that for image classification, Convolutional neural Networks performed better that a simple multi-layer perceptron for image classification. Using different evaluation metrics we, for example, observed that CNN had a 87.06% classification accuracy while MLP had 27% accuracy, both on their respective test sets.**

## 1 Introduction

### 1.1 motivation

Machine Learning models for image classification have been studies by many researchers. This is one of the key tasks in image processing. The goal of image classification is to predict the categories of the input image using its features[5]. This is an important task in many aspects. Qing Li, et al.[6] for example observed that image classification is used in medical imaging applications like lung cancer detection. Various approaches have been used for this application. Our study focuses on artificial neural networks. Specifically we try to understand on MLP and CNN perform on this task. Through various experiments, we aim to see if by adjust some components of the respective neural networks we can get to improve their performance.

### 1.2 Experimental Approach

The two image classification approaches were implemented from scratch in this experiment. This allowed us to study the impact of changing different characteristic of the models on their performance. We varied different activation function used, optimization approaches and the number of layers in then networks. We also increase the number of epoch used in the forward-backwards iterations to increase the efficiency of the networks. To study the performance of the developed models, the image dataset CIFAR10[7] was used. Several pre-processing techniques where applied to the dataset. One-hot encoding and normalization were used to flatten the features.

our investigation indicated that overall Convolutional Neural Networks had a better performance when tasked with image classification as compared to a simple Multi-Layer Perceptron. Increasing the number of number of layers and epoch used in training the MLP did not improve its performance relative to CNN. with an evaluation classification accuracy of 87.06%, CNN had a better performance compared to 27% obtained by MLP on the same dataset.

## 2 Related Works

### 2.1 Image classification by Multi-layer Perceptron

The task of image classification aims to predict the categories of input images using their features. Researchers have developed and trained various artificial neural network structures for image classification. Sahoo et al. (2006)[8] studied multi-layer perceptrons and observed that multi-layer perceptrons are a class of universal approximators. This means that provided sufficiently many hidden units are available, networks with as few as one hidden layer using arbitrary squashing functions can give an approximation with any desired degree of accuracy. G. M. Foody (2004)[9] proposes a multi-layer perceptron structure trained with a back-propagation for image classification. Benediktsson et al. 1990, observes that this approach is able to classify imagery data more accurately than a variety of other widely used approaches[10]

### 2.2 Image classification by Convulational Neural Network

## 3 DataSet

We are using CIFAR-10 dataset which is a subset of the 80 million tiny images dataset that were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class[1].The class labels and their standard associated integer values are list below:[aiplane , automobile, bird ,cat ,deer, dog, frog , horse , ship , truck]

# 4 Convolutional Neural Network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems,image classification, medical image analysis, natural language processing, and financial time series.
CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.[2]

## 4.1 Base Line model : VGG

- **VGG neural networks** Previous derivative of AlexNet focused on smaller window sizes and strides in the first convolutional layer, VGG addresses another very important aspect of CNNs i.e. depth of the network.
- **Input**VGG takes in 224x244 pixel RGB image.
- **Convolutional layers** The Convolutional Layers in VGG use a very small receptive field ( 3x3, the smallest possible size that still captures left/right and up/down). There are also 1x1 convolutional filters which act as a linear transformation of the input, which is followed by a ReLU unit. The convolution stride is fixed to 1 pixel so that the spatial resolution is preserved after convolution.
- **Fully Connected Layers** VGG has Three fully-connected layers: the first two have 4096 channels each and the third has 1000 channels, 1 for each class.
- **Hidden layers** All of VGG's hidden layers use ReLU as activation function. VGG does not generally use locally response normalization, as LRN increase memory consumption and training time with no particular increase in accuracy.
- VGG is an innovative object-recognition model that supports up to 19 layers. Built as a deep CNN, VGG also outperforms baselines on many task and datasets. VGG is now still one of the most used image-recognition architecture.[3]

# 5 Model Design

## 5.1 Libraries used

Keras, Tensorflow, Matplotlib , PlaidMl [For GPU acceleration on AMD GPU]

## 5.2 Pre-processing Dataset

In our Dataset we have 50000 examples in the training dataset and 1000 in the test dataset and images are square with 32x32 pizels and color with three channels.

- **One hot encoding :** There are 10 classes in the CIFAR-10 dataset represented as unique integers. Therefore, we use one hot encoding for class element of each sample transforming the integer into a 10 element binary vector with a 1 for the index of class value. For this purpose we use ***keras.to_categorical*** utility function provided by keras for one hot encoding.
- **Normalization :** Pixel value in the dataset are unsinged integers in the range between 0 and 255. To make efficiency, we re-scale the values between 0 and 1 by dividing the pixel values by 255 i.e. maximum pixel value.

## 5.3 Building Multi-Layer Perceptron

Multi-layer perceptron is a type of network where multiple layers of a group of perceptron are stacked together to make a model. The model takes inputs, multiplies them by weights and adds a bias to produce inputs.

$$f(X) = W.X + b$$

To add non-linearity to our network, we implemented Rectified Linear Units(ReLu) activation function. ReLU function is zero for any input value below zero and the same value for values greater than zero. The layers have both feed-forward and and backpropagation loops. The forward loop inputs data and generates output. The backpropagation loops on the other hand trains the model by adjusting weights and minimize the output loss. To quantify loss, we created a loss function. Since we are using probabilities to make predictions, for better numerical stability we used Log-softmax.

$$Log - softmax = -a_{correct} + log \sum_i e^{a_i}$$

Using the layers we built earlier one, we defined our Multi-Layer perceptron as a list of layers that feed input into each other. We built Mini-Batch Stochastic Gradient Descent from scratch. Using this, we split the data into min-batches and feed each of the batches into the network while updating weights and every iteration.

### 5.3.1 Experimental setting and Model Parameters

Efforts to experiment across parameter values on the dataset was made. We used different activation functions and studied their impact. For MLP, we ReLU and Sigmoid activation functions were used. We varied the size of mini-batches used for training. To study the effect of density and number of layers, we tried a network of 2 layers and another with 3 layers with varying numbers of units. The output layer had 10 units that used softmax activation function with stochastic gradient descent. The networks were trained multiple times, increasing the number of Epochs on every iteration and the performance of the models was recorded as seen in the Results section(Section 6). Over the course of training the models, we recorded the cross entropy loss and plotted it against the epoch number. We also plotted the accuracy against the respective epoch. We also experimented on the effect of batch size by varying the sizes on multiple runs. However, due to computational resources, we did not get to explore many batch size variations. We present the results of the experiment and best performing parameters in section 6(Results)

To optmize our model, we used mini-batch Stochastic gradient descent and initialized the weights for the non-linearity using Xavier initialization.

### 5.4 CNN models explored

After preprocessing the data, we now define our neural network model to be trained on our training set. Due to limited computation resources we don't perform cross validation and instead uses test dataset to test accuracy of our trained model. For training purposes, we provide an upper limit of 150 epochs and batch size of 64. Our models are trained on GCP cloud instance with TESLA T4 and TESLA k80 gpus.
For our image classification purpose we compare various models and optimization techniques. We choose VGG models as our baseline models. Choice of Baseline model is based on easy to understand architecture and performance enlisted.
Our network architecture involves stacking convolutional layers with small 3x3 filters followed by max pooling layer ,down sampling feature maps. Together, these layers form a block, and these blocks can be repeated where the number of filters in each block is increased with the depth of the network. Paddig is used on the convolutional layers to ensure the height and width of the output feature maps matches the inputs. For our activation function we compared mainly three activation functions for hidden layers. *Sigmoid, tanh , ReLU , elu.*

In our Model, the feature maps output from the feature extraction part of the model is flattened. We interpret them with one or more fully. connected layers, and then output a prediction. The output layers have 10 nodes for the 10 classes and uses the softmax activation function.

Model was optimized using **Stochastic Gradient Descent** *with 50 epochs, momentum=0.9 and learning rate = 0.001*

## 6 Results

### 6.1 VGG3

We compared 1, 2, 3 block VGG as our models for training purpose. All the models were trained for 50 epochs initially. VGG with 1 block performed worse with *training set accuracy of 99.9%* and *test set accuracy of 66.75%*. Similarly, VGG with 2 block attained accuracy score of *99.90% on training dataset* and *71.51% on test dataset*. Finally VGG with 3 block performed best with accuracy score of almost *100.0% on training dataset* and *73.86% on test dataset*.

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_20 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               262272
_________________________________________________________________
dense_8 (Dense)              (None, 10)                1290
=================================================================
Total params: 550,570
Trainable params: 550,570
Non-trainable params: 0
```

Figure 1: VGG3 learning curve

| Dataset | Accuracy |
|---|---|
| Training set | 100.0 |
| Testing set | 73.86 |

Table 3: 3block VGG model results

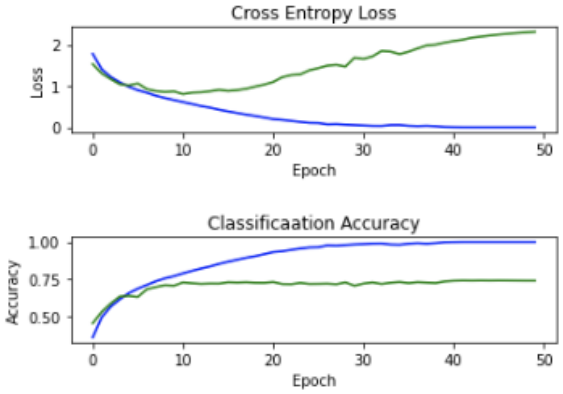


Figure 2: VGG3 learning curve

All VGG3 models continuous improvement on test dataset at least 50 epochs. Therefore models have sufficient capacity to learn the weights. By comparing result of all the three models we can conclude that our accuracy increases with increase in depth of the network. So if we increase the depth of the model we can see some drastic increase in the accuracy. That can be the reason why VGG16 and VGG19 tend to perform extremely well on image classification tasks.
Since VGG3 model starts to showcase the high variance trend(overfit) after 20 - 30 epochs we need to add regularization. One proposed solution is to slow down the rate of convergence by using techniques such as data augmentation, variable learning rate, different batch size , batch normalization, dropout technique.vFor further model improvement we are going to use VGG3 model just because deeper models tends to perform better on complex task such as image classification.

### 6.2 Variation of activation functions

For every layer in the network we choose to test different activation functions and comparison for activation function selection. We compared *'sigmoid' , 'tanh', 'elu' , 'ReLU'* activation function for our VGG3 Model.
for our test VGG3 model used 50 epochs and learning rate = 0.001 for better convergence without exhausting limited computation resources available.

| Activation Function | Test Accuracy |
|---|---|
| Sigmoid | 10.00 |
| tanh | 73.33 |
| elu | 72.94 |
| ReLU | 73.86 |

Out of all the activation functions tested on our VGG3 network *Sigmoid* tends to perform the worst out of all the activation functions with test accuracy of 10.00%. The reason for such a low accuracy can either be vanishing gradient problem or *Sigmoid* has slow convergence and might need high amount of epochs to converge. *'ReLU'* activation function outperformed *'tanh','elu'* with slight margin.

### 6.3 Regularization technique

we could try various regularization techniques that can helps with overfitting. Techniques that help in slowing down the convergence rate tends to help more in the case of high variance. We choose *'ReL*U' choose activation function and *'softmax'* function for classification problem to compute the probabilities for the classes.

#### 6.3.1 Dropout Regularization

The Key idea of dropout is randomly drop units ( along with their connections ) from the neural network during training process. This prevents

units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This is significantly reduces overfitting and gives major improvement over the regularization methods.[12]
dropout can be added to the model by adding new dropout layers, where the amount of nodes removed is specified as a parameter. First we tried to add dropout layers after each max pooling layer and after the fully connected layer, and used a fixed dropout rate of 0.2 i.e. we retain 80% of the nodes. Then we try to use variable dropout rate to overcome overfitting. Using dropout rate = 0.2 we achieved *training score of 83.68%* and *test score of 80.82%*.

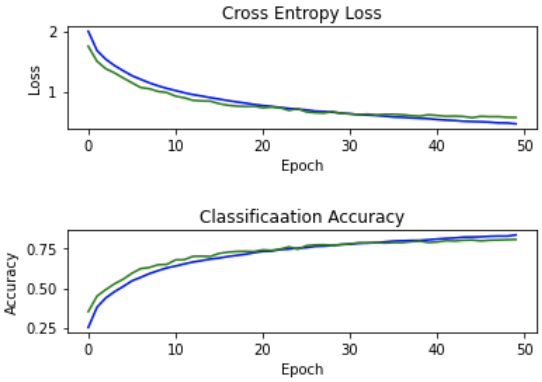


Figure 3: VGG3 model with droptout rate =0.2 (80% network retention)

And a Dropout rate of 0.5 achieved a score of *65.85% on training dataset* and *70.51% on test dataset*.

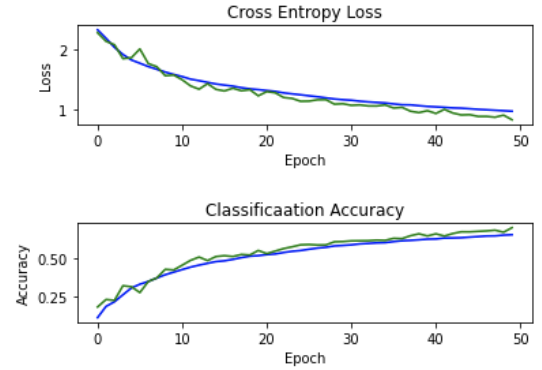


Figure 4: VGG3 model with droptout rate =0.5 (50% network retention)

We invested the effect of variations of dropout in model. we used dropout pattern of 0.2 dropout rate for first layer, 0.3 dropout rate for second layer and dropout rate of 0.4 and 0.5 for next two layer.[14]. using Variable dropout rate resulted in score of 75.31% on training dataset and 77.77% on test dataset.

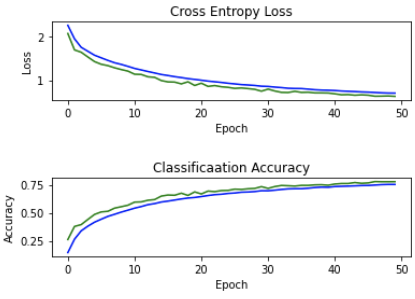


Figure 5: VGG3 model with variable Dropout rate

### 6.3.2 Weight Regularization

Weigh Decay is another technique out there to reduce the overfitting of a deep learning neural network model on the training data and improve the performance of the model on new data such as the holdout test set. Weight regularization involves updating the loss function to penalize the model in proportion to the size of the model weights. larger weights result in more complex and less stable model, whereas weights are often more stable and more general. Weight Regularization can be considered as weight shrinkage. For our model optimization we added weight regularization to the convolutional layers and fully connected layers. with l2 penalty=0.001 we achieved a *score of 76.00% on training dataset* and score of *70.10% test dataset*. From the above result we can conclude that weight regularization didn't effect the model's accuracy considerably.

### 6.3.3 Data Augmentation

Image data augmentation is quite useful technique used to increase the test dataset accuracy in image classification task. Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.
When we feed image data into a neural network, there are some features of images that we would like the neural network to condense or summarize into a set of numbers of weights. In the case of image classification, these features or signals are the pixels which make up the object in the picture. On the other hand, there are features of the images that we would not like the neural network to incorporate in its summary of the images( the summary is the set of weights). In case of image classification, these features or noise are pixels which form the background in the picture. One solution is to create multiple alterations of each image, where the signal or the object in the picture is kept invariant, whilst the noise or the background is distorted. These distortions include cropping, scaling and rotating the image , among others. Therefore, the network of neurons observes the in variance in the images and encodes this information or signal is the set weights which summarize the training data. [3]
There are different type of data augmentation that could be used. Our main goal during data augmentation is to preserve as much as information possible. The types of random augmentation that could be useful include a horizontal flip, minor shifts of image and 10% shifts in the height and width of the image.Data augmentation helped us to achieve score of 85.04 on training dataset and score of 81.85 on test dataset.

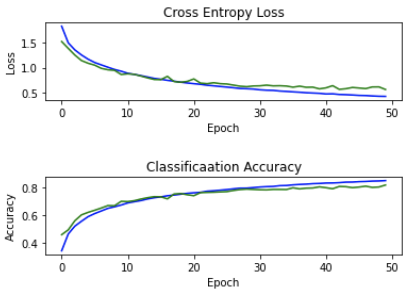


Figure 6: VGG3 model with Data Augmentation

### 6.3.4 Further Optimization

From the results , presented above, we concluded that dropout *rate = 0.2*, and data augmentation both helped in boosting testing the accuracy on test dataset and avoid any overfitting. We saw further boost in accuracy after combining both dropout and data augmentation techniques. Using Data augmentation and dropout rate of 0.2 resulted in *training accuracy of 64.87%* and *test accuracy of 67.21%*. Combining both regularization technique, Dropout and Data augmentation, resulted in higher accuracy on test dataset. Learning curves provide additional information about convergence of our model. The convergence behaviour of the model is overall better than either fixed dropout and data augmentation alone. Learning has been slowed without overfitting, allowing continued improvement. To further increase the model accuracy we can use the variable dropout with high dropout rate as we go deeper in network.

Finally we can use batch normalization[15] Batch normalization is a technique for improving the speed, performance, and stability of Neural Networks. It is used to normalize the input layer by re-centering and re-scaling.

For the final model, we used Stochastic gradient descent for better optimization of objective function, Variable dropout ( increasing dropout with increasing depth of the model ), Data augmentation and batch normalization and learning rate = 0.001 for better convergence and epoch = 200 , so that model has enough time to learn weights and finally using momentum of 0.9 to avoid getting stuck in local minima and speed up convergence of gradient descent.

| Final Model Results | |
|---|---|
| **Dataset** | **Accuracy** |
| Training accuracy | 85.94 |
| Test accuracy | 87.06 |

For

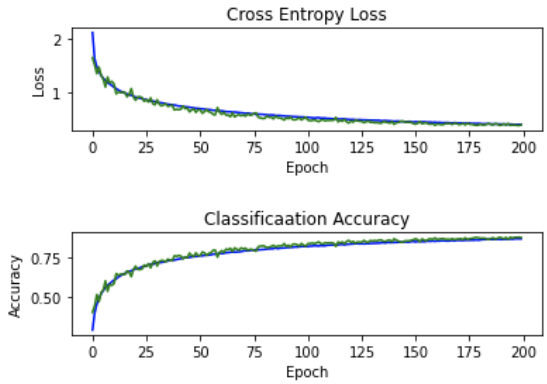


Figure 7: VGG3 with Data Augmentation + dropout rate=0.2

the final model training we increased the epochs from 50 to 200 in order to give a chance to model to learn. Reason of high accuracy after increasing epochs from 50 to 200 might be due to the fact that addition of all these regularization technique slowed down the learning process and hence need far greater number of epochs to regain 80%+ accuracy on test data. From learning curve we can confirm that models shows continued improvement for nearly 200 epochs. Therefore increasing epochs further might result in higher accuracy. For further improvement of the model we can add early stopping with with a patience set between 20 and 100. But due to limited resource and time we exhaust our model optimization with dropout , batch normalization, dataset normalization, SGD, and data agumentation with a final accuracy on test data equal to 87.06%.

## 6.4 Multi-Layer PERCEPTRON

### 6.4.1 Activation function against test set accuracy

| Activation function | Accuracy |
|---|---|
| Sigmoid | 27.0 |
| ReLu | 27.0 |

Sigmoid and ReLU where used for hidden layers while Log-softmax was used for output layers in all experiments

### 6.4.2 Number of hidden layers against accuracy

| Hidden Layers | Number of units | Training accuracy | Testing accuracy |
|---|---|---|---|
| 2 | 80, 30 | 57.80 | 20.0 |
| 3 | 80, 50, 30 | 95.5 | 25.0 |

### 6.4.3 Best performing parameters on training and validation sets

The following parameters had the best performance both on the training and testing sets: learning_rate = 0.01, epoch = 100, num_hidden layers = 2(80, 30), activation function = ReLU

| Set | Accuracy |
|---|---|
| Training | 95.5 |
| Validation | 27.0 |

### 6.4.4 Effect of learning rate and epoch number on cross entropy loss

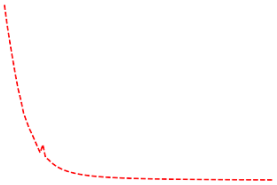

Figure 15: effects of learning_rate = 0.01 and number of epoche (x-axis) on cross entropy loss(y-axis).

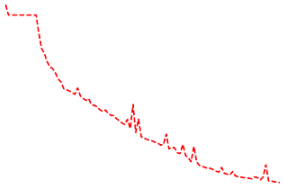

Figure 16: effects of learning_rate = 2.5 and number of epoche (x-axis) on cross entropy loss(y-axis).

# 7 Conclusion and Discussion

The intent of this experiment was to compare the performance of Convulational Neural Networks and a simple Multi-Layer Perceptron on classification of image data. We observed that for both models,increasing the density of hidden layers increased classification accuracy to some extent. Other factors like learning rate, number of layers and type of activation functions affected performance. For both models, ReLu non-linearity had higher performance than Sigmoid activation function. Despite not being able to explore the entire parameter space, Covulational Neural networks consistently showed an overall higher performance than a simple Multi-layer perceptron.

In our experiments, the lack of extensive model selection due to constraints of time and computational power may have caused comparatively lower performance across all tests. To better understand the performance of these models, future experiments aught to have both time and computing resources so that they can explore a wider parameter space

# 8 Statement of Contribution

Ujjwal worked on the Convulational Nerual Netork. he developed the model, tested, optimized and wrote its part of the report. Furaha worked on the Multi-Layer Perceptron. Likewise, he developed, tested and optimized it and wrote its report.

# References


[1] https://www.cs.toronto.edu/ kriz/cifar.html

[2] https://en.wikipedia.org/wiki/Convolutional_neural_network

[3] https://arxiv.org/pdf/1409.1556v6.pdf

[4] D Anderson, G McNeill - Kaman Sciences Corporation, 1992, ARTIFICIAL NEURAL NETWORKS TECHNOLOGY

[5] Le Hoang Thai et al., I.J. Information Technology and Computer Science, 2012, 5, 32-38 , Image Classification using Support Vector Machine and Artificial Neural Network

[6] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng and M. Chen, "Medical image classification with convolutional neural network," 2014 13th International Conference on Control Automation Robotics Vision (ICARCV), Singapore, 2014, pp. 844-848.

[7] Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

[8] Kurt Hornik, Maxwell Stinchcombe, Halbert White, Multilayer feedforward networks are universal approximators, Neural Networks,Volume 2, Issue 5,1989,Pages 359-366

[9] G. M. Foody (2004) Supervised image classification by MLP and RBF neural networks with and without an exhaustively defined set of classes, International Journal of Remote Sensing, 25:15, 3091-3104, DOI: 10.1080/01431160310001648019

[10] Yang, H, van der Meer, F, Bakker, W and Tan, Z. J. 1999. A back-propagation neural network for mineralogical mapping from AVIRIS data. International Journal of Remote Sensing, 20: 97–110.

[11] Xiangyu Zhang, Jianhua Zou, Kaiming He, Jian Sun (2015),Accelerating Very Deep Convolutional Networks for Classification and Detection

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research, 2014

[13] Daniel Ho, Eric Liang, Richard Liaw, 1000X faster data augmentation Berkely Artificial Intelligence Research, june 7, 2019

[14] Nitish Srivastava,Geoffrey Hinton,Alex Krizhevsky,Ilya Sutskever,Ruslan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting , Journal of Machine Learning Research 15 (2014) 1929-1958

[15] Sergey Ioffe, Christian Szegedy, Batch Normalization : Acceralting Deep Network Training by Reducing Internal Covariate Shift,arXiv:1502.03167 , 11 feb 2015