Image Classification of CIFAR 100

CMPE 257 – Machine Learning

**Team Name -Group 1**

Team Members:

Chaithanya Reddy Bogadi - 013803998

Naga Sindhu Korlapati - 012467832

Nandini Puppala - 013752713

Prathusha Koouri - 013710658

Sindu Ravichandran - 013769824

Venkata Anil Kumar Thota - 012525734

**Best Accuracy obtained with ML Algorithm: CNN with 86.92%**

**Google Collaboratory Link:** <https://colab.research.google.com/drive/1X-TI0AZSEJXOB0tcCAxScVk24Pf-T4e8?authuser=1>

# 

# Work Collaboration Table Summary

|  |  |  |
| --- | --- | --- |
| Data Cleaning | We had to check for the right images, map and separate the images accordingly. | Naga Sindhu Korlapati, Prathusha Koouri, Sindu Ravichandran |
| Data Preparation | Normalise the data, checked the distribution and correlations. | Nandini Puppala, Venkata Anil Kumar Thota, Chaithanya Reddy Bogadi |
| List of Algorithms to be used. | we came up with a list of ML algorithms that work well with image classification data. | Chaithanya Reddy Bogadi, Nandini Puppala, Sindu Ravichandran, Naga Sindhu, Venkata Anil Kumar Thota, Prathusha Koouri |
| Traditional Machine Learning Algorithms | Worked with all the classic classification algorithms, fine-tuned parameters. | Venkata Anil Kumar Thota, Chaithanya Reddy Bogadi, Prathusha Koouri |
| Ensemble Algorithms | Tried improving the accuracy with the ensemble methods | Sindu Ravichandran, Nandini Puppala, Naga Sindhu |
| CNN | We worked on CNN tuning hyperparameters and regularization parameters to get better results. | Prathusha Koouri,  Nandini Puppala, Sindu Ravichandran, Naga Sindhu, Venkata Anil Kumar Thota, Chaithanya Reddy Bogadi, |
| Report | We documented the whole project work in a report. | Chaithanya Reddy Bogadi, Nandini Puppala, Sindu Ravichandran, Naga Sindhu,Venkata Anil Kumar Thota, Prathusha Koouri |
| Powerpoint Presentation | We worked on PPT together and created the content. | Nandini Puppala, Sindu Ravichandran, Naga Sindhu, Venkata Anil Kumar Thota, Chaithanya Reddy Bogadi, Prathusha Koouri |

**Contents**

[Work Collaboration Table Summary 2](#_Toc7379156)

[**Introduction:** 4](#_Toc7379157)

[**Software and libraries used:** 4](#_Toc7379158)

[**Software:** 4](#_Toc7379159)

[**Libraries:** 4](#_Toc7379160)

[**Dataset:** 5](#_Toc7379161)

[**Data Cleaning:** 5](#_Toc7379162)

[**Data Validation:** 7](#_Toc7379163)

[**Machine Learning Algorithm Summary:** 9](#_Toc7379164)

[**Logistic:** 9](#_Toc7379165)

[**K-NN:** 10](#_Toc7379166)

[**SVM:** 10](#_Toc7379167)

[**Random Forest:** 11](#_Toc7379168)

[**Gradient Boosting:** 11](#_Toc7379169)

[**XG Boosting** 12](#_Toc7379170)

[**MLP with Sklearn and NN with Keras** 13](#_Toc7379171)

[**CNN** 14](#_Toc7379172)

[**Reference Online links:** 17](#_Toc7379173)

# **Introduction:**

Image classification is a supervised learning problem and is one of the complex tasks for machines, which can be solved using Machine Learning and Deep learning algorithms. Sometimes it is difficult even for humans to differentiate between images of different species. Recent advances in deep learning made tasks such as Image and speech recognition possible. Image classification is performed by taking an image as input and producing an output class as: Binary or MultiClass or their Probabilities. The emphasis is placed on the summarization of major advanced classification approaches and the techniques which are used for improving classification accuracy.

In this study, we are considering two super classes from CIFAR 100 dataset, they are Household Electronics and Household Furniture and our task here is to consider all the 5 subclasses as one class and perform a binary classification on 2 super classes. We applied multiple machine learning algorithms to accomplish this task and finally were able to achieve excellent results with Convolutional neural networks as they can take the image data as input in 3 dimension and process through the image part by part, using filters which other algorithms cannot.

# **Software and libraries used:**

## **Software:**

* **Python**: Python comes with a huge amount of inbuilt libraries. Many of the libraries are for Artificial Intelligence and Machine Learning frameworks.
* **Tableau**: It is mainly used for interactive data visualization purpose.
* **Microsoft Excel:** It is spreadsheet that allows user to do calculations, graphing tools, pivot tables.

## **Libraries:**

* **Numpy:** It is mainly used for scientific computing and supports large, multidimensional arrays and matrices.
* **Pandas**: It is one of the most widely used tool in data wrangling/munging.
* **Seaborn:** It is a data visualization library.
* **Matplotlib:** It is used to create 2D plots and graphs using python library.
* **Sklearn:** It is a robust library that provides wide range of supervised and unsupervised learning algorithms.
* **Tensor flow:** It is a fast-numerical computing library that can be used to create deep learning models.
* **Keras:** It is a high-level API built on top of Tensorflow. It is more user-friendly and easier to implement compared to Tensorflow.

# **Dataset:**

CIFAR 100 dataset is available in Keras library. It can be imported from Keras datasets. In this project we worked on CIFAR 100 dataset. This dataset consists of 60000 colored images with dimensions 32\*32\*3, partitioned by 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 super classes. Each image comes consists of two label\_modes:

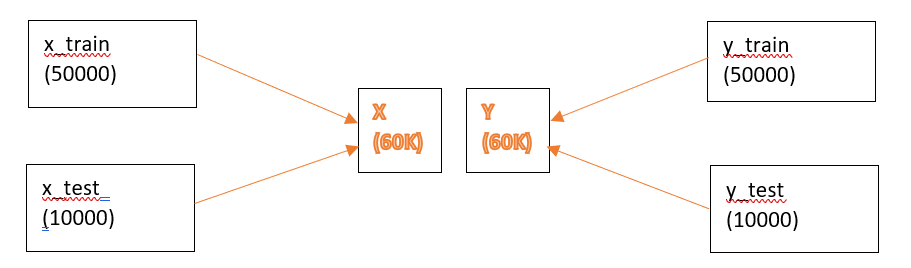
1. Coarse Labels - Images are partitioned with Superclass label names. (eg: Household electronics, Furniture)
2. Fine Labels - Images are partitioned with Class label names. (eg: lamp, keyboard, chair, etc.)

In this project, data is loaded with fine labels.

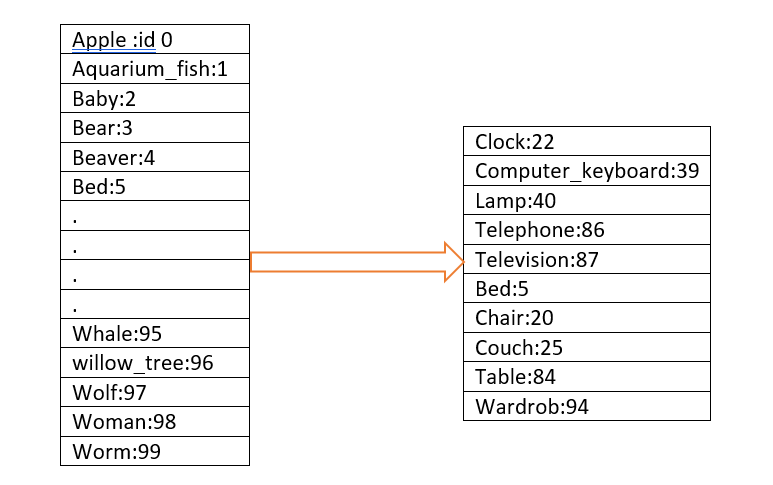
For our analysis, we worked on two super classes i.e, Furniture and Electronics.

## **Data Cleaning:**

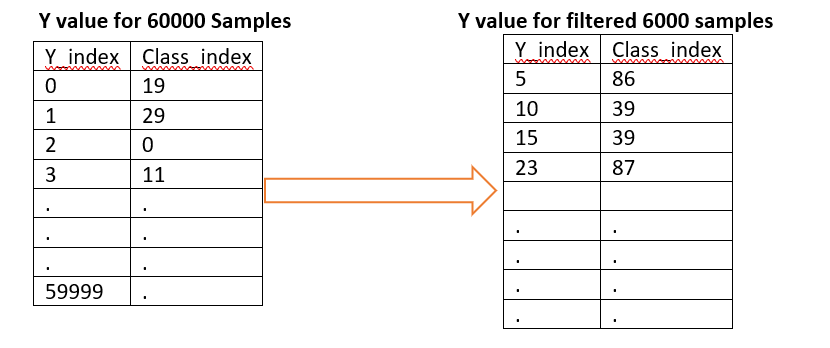
The data from keras is loaded as x\_train,y\_train,x\_test,y\_test. All this dataset are combined to ‘x’ and ‘y’ dataset to filter the required superclass data.



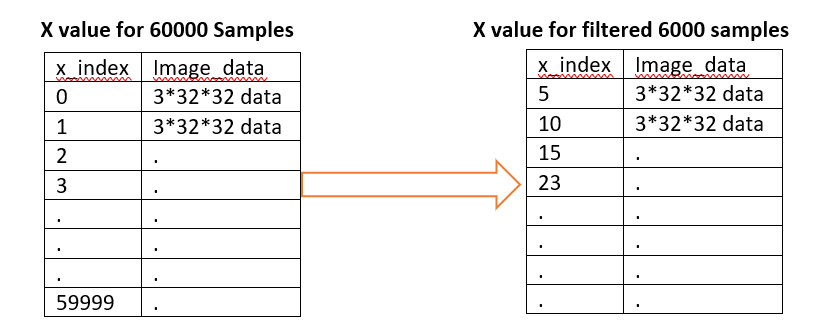
The fine labels of the data are stored in alphabetical order. We created dictionary with all the labels in correct order to filter the class index for the super class ‘Household\_Electronics’ and ‘Household\_Furniture’.



With this class index, we filtered out the index value of the samples in the ‘y’ dataset which belongs to the superclass- Household\_Electronics and Household\_Furniture. 6000 samples are filtered out from the original 60000 samples.

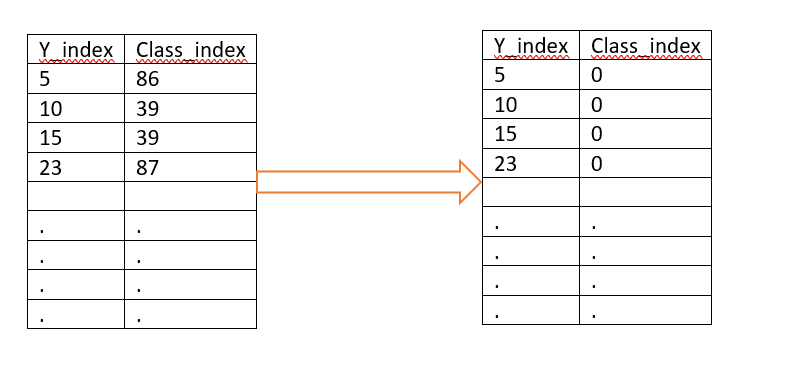


With the filtered ‘y\_index’ value, we were able to filter the ‘x’ dataset also.



The filtered ‘x’ and ‘y’ value contains 6000 images, 600 images for each of the 10 classes. Our main intention is to perform binary classification with the images. Hence, we wanted to update the Target variable ‘y’ to binary format.

* Household\_Electronics : 0 (22,39,40,86,87)
* Household\_Furniture : 1 (5,20,25,84,94)



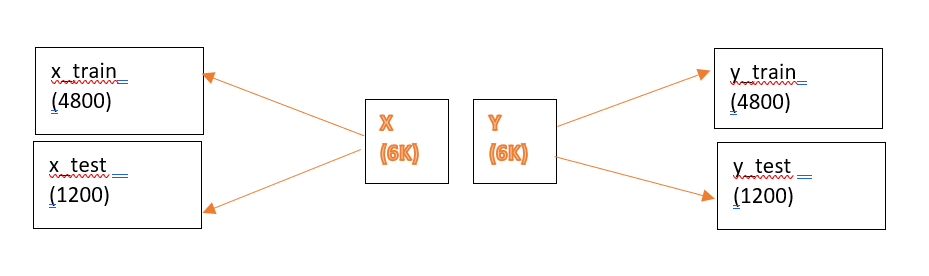
## **Data Validation:**

To validate that the filtered images belongs to the super class Household\_Electronincs an Household\_Furniture, we have displayed 40 images, 4 from each class along with their fine and coarse labels. Below is the grid of images we plotted for validation.

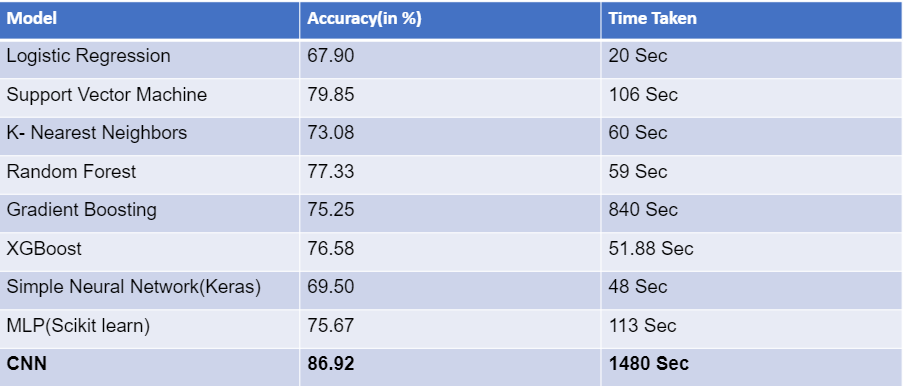


Now the data is ready for applying machine learning algorithms. We use 20% data for testing. Out final train- test dataset shape was,

* X\_train, y\_train : 4800
* X\_test, y\_test: 1200



# **Machine Learning Algorithm Summary:**



## **Logistic:**

Logistic regression is the standard solution for binary classification problems. “Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a [logistic function](https://en.wikipedia.org/wiki/Logistic_function)” .Hence we choose logistic as our first algorithm. It is pretty similar to linear algorithm with a key difference that output value modeled to binary values either 0 or 1. It also gives predicted probability as output. These probabilities can be converted into class predictions. With this model we could get an accuracy of **67.90** in **20** seconds.

## **K-NN:**

K-NN is a supervised machine learning algorithm. It is one of the popular techniques used while handling large datasets. KNN uses three types of distance functions: Euclidean, Manhattan and Minkowski. In this project, we used Euclidean distance.

KNN classifies a new instance based on the voting criteria. Applying KNN algorithm to this dataset with default hyper parameters we got an accuracy of **66.5**, after hyper parameter tuning, we got **73.08** with **13** n\_neighbors.

## **SVM:**

A support vector machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. When labeled training data is given as an input the algorithm outputs an optimal hyperplane. The advantage of SVM is that once a boundary is established most of the training data is redundant. So, we utilized this algorithm for our classification problem.

There are three tuning parameters in SVM classifier. They are:

1)Regularization parameter

2)Kernel and

3)Gamma.

Regularization parameter(C) tells SVM how much we want to avoid misclassifying each training example. For large value of parameter C optimization will choose smaller margin hyperplane,on the other hand for the smaller value of parameter C optimization will choose larger margin hyperplane. Kernel is a method of using a linear classifier to solve a nonlinear problem. These kernel functions can be of different types. For example: linear, nonlinear, polynomial, sigmoid and radial basis function(rbf). gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’.

Using GridsearchCV we got a set of best parameters. Using these parameters, we got an accuracy of **79.58%** which is better than the accuracy with default parameters. The best set of parameters are:

C: 1, kernel: rbf and gamma = 0.01.

After working on traditional ML algorithms, we wanted to check if the accuracy improves with ensemble methods which generally reduce bias, variance, noise which will eventually reduce the gap between actual and predicted values.

## **Random Forest:**

Random Forest is one of the popular ensemble technique.It is widely accepted winning algorithm in many data science competitions. Random Forest is a supervised Machine Learning algorithm which is built by picking up features randomly from the data and builts decision trees in making predictions on classification and regression problems.

The main tuning parameters with Random Forest were n\_estimators and max\_depth. Since a forest is built on picking up features from the data. It is important to set how many features, it should pick randomly and how deep it should build the tree. A deep decision trees are prone to overfitting problem. Hence, it is important to look in to that as well.

As we went with more estimators and depth but score is saturated.Hence, we finalized our parameters with no.of estimators as 500 and Max.depth as 300 and obtained an accuracy of **77.33%** .

## **Gradient Boosting:**

There are various boosting techniques in practice which are giving very good results like Gradient boosting, XGboost and Adaboost. In this method more emphasis is on the data which gives wrong predictions in order to improve the accuracy.

Boosting is an ensemble technique which selects the predictors sequentially rather than independently. Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Main logic behind this algorithm is to focus more on residuals and modify the model to predict residuals of the previous models. In the end, all the predictors are combined by giving some weights to each predictor.

Parameter tuning in gradient boosting algorithm can be done in two different levels,

* Tree based parameters:
  + Min\_samples\_split – minimum no.of samples required in a node to split
  + Min\_samples\_leaf – minimum samples required in leaf node
  + Max\_depth - max depth of a tree
  + Max\_features – no of features to be considered while searching for best split.
* Boosting Parameters:
  + Learning\_rate – determines the impact of each tree in the output
  + N\_estimators – no of sequential trees to be modeled
  + Subsample – fraction of observation to be selected for each tree
* Miscellaneous Parameters:
  + Random\_state – random number seed
  + Warm\_start – to add additional trees to the previous fit of the model

Here, in our model we have used the above listed parameters to improve the accuracy from 71.20 to **75.25**.

## **XG Boosting**

XGboost: Xtreme gradient boosting, actually refers to the engineering goal to push the limit of computational resources for boosted tree algorithms. XGBoost makes the best use of available resources to train the model. Advantages of XGBoost:

* Sparse Aware (handling missing data automatically)
* Block Structure (supports parallelization of tree construction)
* further training (to further boost an already fitted model on new data)
* Out-of-core computing**:** This feature optimizes the available disk space and maximizes its usage when handling huge datasets that do not fit into memory.

We got a list of best parameters using GridSearchCV. By tuning the following hyperparameters

1. n\_estimators: 50
2. nthread: 4
3. objective: binary-Logistic,
4. learning\_rate: 0.05

we obtained the accuracy of 71.20 % (default) and improved the accuracy to 75.25% using tuned hyper parameters.

* <https://medium.com/datadriveninvestor/an-introduction-to-grid-search-ff57adcc0998>
* <https://medium.com/@kesarimohan87/model-selection-using-cross-validation-and-gridsearchcv-8756aac1e9d7>

## **MLP with Sklearn and NN with Keras**

A multilayer perceptron (MLP) also called Feedforward Neural Network is a [deep, artificial neural network](https://skymind.ai/wiki/neural-network). They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. MLPs with one hidden layer are capable of approximating any continuous function.

We tried many hidden layers at the beginning and could not arrive at any better result than when we tuned the model to use single hidden layer with 1000 neurons, which gave an accuracy of 74.33% and it took 960 seconds to execute.The Relu(Rectified linear unit) worked best as the activation function, which will convert all negative values to zeros and give better results.

We used the Sequential model in Keras which is same as feedforward neural network. Each Dense layer represents a hidden layer and the final Dense layer represents an output layer. Here, Unlike MLP with SKlearn, we can use different activation functions for different layers and also, we can use an activation for output layer too.

After implementing and trying different layers and activations, our final model which gave a better accuracy of **69.5%**, where we used following parameters:

One hidden layer with 1000 neurons and activation used is ‘relu’, with an input shape of 3072 neurons.

Output layer consisting of 2 neurons and activation ‘softmax’ (It is similar to sigmoid which squashes the output between 0 and 1 but also makes sure that all the outputs sum is equal to 1).

When using Keras we need to compile the model in which we used 3 parameters: optimizer: ‘Adam’

Loss: Categorical\_crossentropy(As we have 2 categories of classes in output)

Metrics: Accuracy

We model.fit to build a model where we used following other parameters apart from X and Y training data:

Epochs: 20

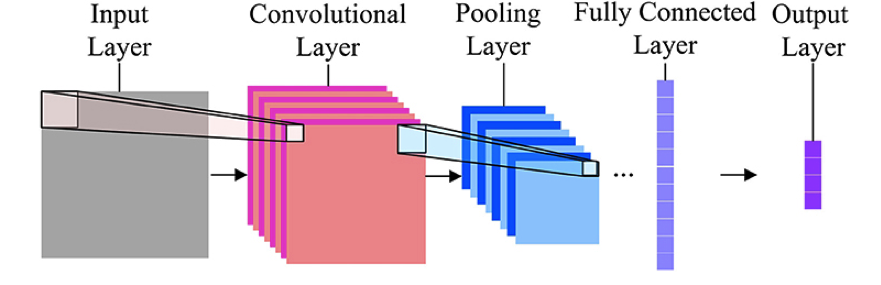
Early Stopping with patience 2: If the loss increases after 2 epochs the model stops executing.

We used model.evaluate to check for accuracy/score of the model.

## **CNN**

Convolutional Neural Network also known as Convnets or CNN is an artificial neural network, which is good at analyzing spatial and temporal dependencies in an image. It is used in various data analysis and classification problems as well. Wide applications like image recognition, image classification and pattern recognition etc are mostly implemented using CNN.

CNN is made up of several layers that process and transforms an input and produces an output. Technically, CNN model takes each input image and passes through series of convolution layers with filters (kernels), pooling, fully connected layers, and apply activation function to classify an object with probabilistic values between 0 and 1. The complete process is shown in the below diagram.



**Convolution layer:** It is mainly used to extract features in an image like multiple edges and corners etc. In each convolution layer we need to specify the number of filters to have.

**Pooling layer:** It is a kind of down sampling or sub sampling method, which reduces the dimensionality of each map but retains the important information. Methods like max, average and sum pooling are used.

**Fully Connected layer:** In this layer, we flattened our matrix into vectors and feed it into FC layer like neural network, where each and every node is connected to the next layer.

Based on requirements we can apply striding and padding concepts in our model.

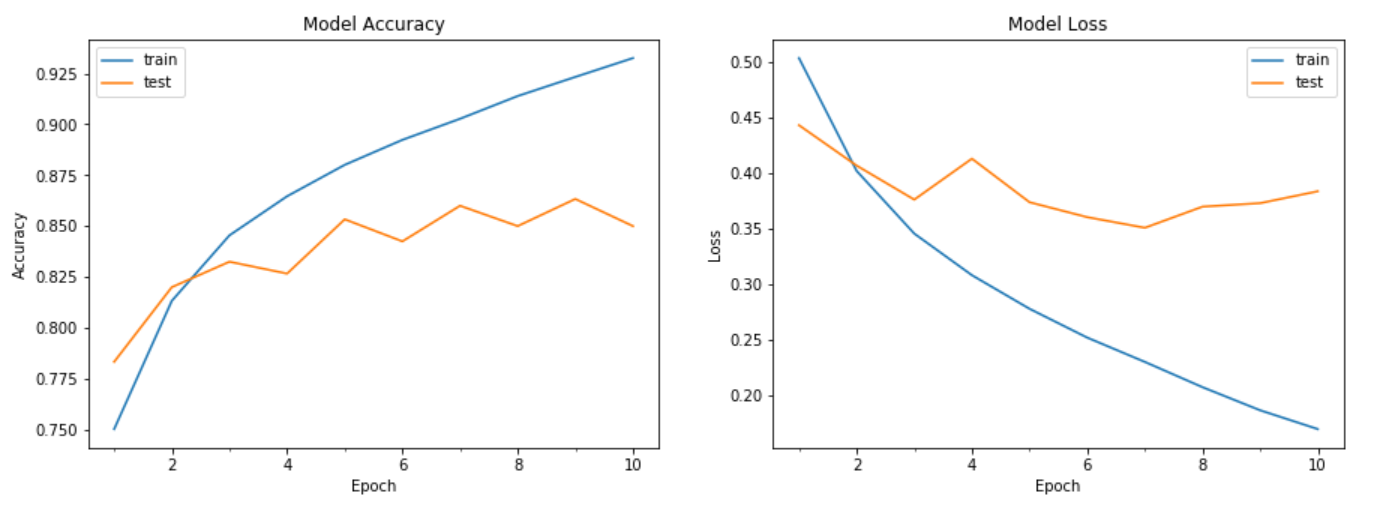
**Hyper Parameters Tuning in CNN**

* **Learning rate:** It controls the update of weight in each optimization algorithm like SGD, Adam, and RMSprop etc.
* **Number of epochs:**It is the number of times entire training set has to pass through neural network.
* **Batch size:** Convnet is sensitive to batch size, and a mini batch ranges from 16 to 128 is a good choice.
* **Activation Function:** It introduces non-linearity to the model. Usually, rectifier works well with convnet. Other alternatives are sigmoid, tanh etc.
* **Number of hidden layers:**It is usually good to add more layers until the test error no longer improves. Less number of layers can also lead to problem of under fitting. **Drop out for Regularization:** Dropout is a preferable regularization technique to avoid overfitting in deep neural networks. The method simply drops out units in neural network according to the desired probability. A default value of 0.5 is a good choice to test with.

We used below links for implementing CNN model.

1. <https://github.com/tatsuyah/CNN-Image-Classifier>
2. <https://github.com/sebsquire/Dogs-and-cats-image-classification-CNN>
3. <https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8>

After reading several research papers, we got an idea on specific parameters to implement in the CNN model. Popular architectures suggested that as the depth of the network increases, the number of filters should be increased. Keeping that in our mind, we have built CNN model using various parameters into consideration. Initially our model gave an accuracy of **80%**. After running several experiments by changing parameters like number of filters, activation function, learning rate and epoch, finally we achieved an accuracy of **85%**. But their does occurred a problem of over fitting. In order to resolve this issue, we implemented regularization concepts in our model.

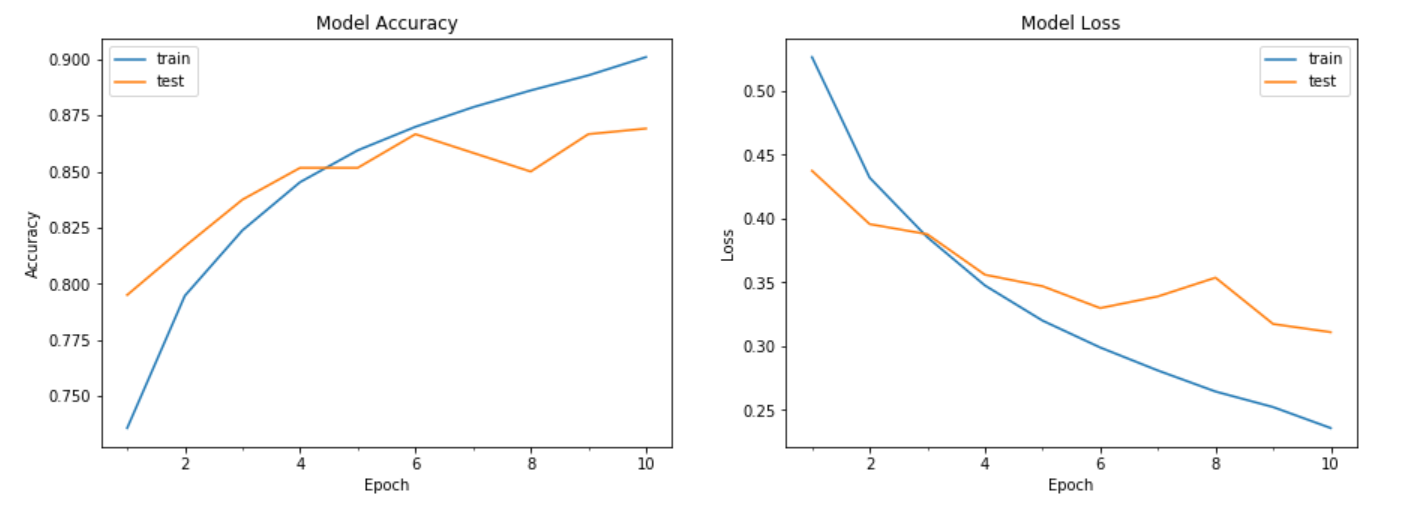


In the above figure, right side graph clearly depicts the problem of overfitting, We can observe that loss for the test data decreased to a certain extent and then it got increased, whereas for training data it was continuously decreasing w.r.t to epoch.

1. Drop out: In our model dropout of 0.5 addressed the main problem of overfitting.

2. More training example: Since we already implemented data augmentation, there is no chance we can improve this.

3. Weight regularization: we used L2 regularization, but didn’t resolve the problem



Here loss for both test and training data continuously decreased w.r.t to epoch.

From above, we can conclude drop out method does help in solving overfitting and after running nearly 100 experiments; at last we achieved an accuracy of **86.72%** using CNN.

**Conclusion and Future work**:

With the specific work we did in this project, comparing and contrasting the performance of various traditional ML algorithms with Deep learning neural networks with this dataset, we found that implementing a CNN for identification of label type of household dataset images gave promising results with regards to accuracy of 86.9% but the computational performance is a little low. However, we believe that with some further improvements of parameters we could improve the accuracy sufficiently. We will implement Grid search CV for CNN and further fine tune the parameters. We would recommend further investigations into the use of Convolutional neural networks for the Image classification project, as we believe that it has great potential, specially in dealing with image recognition problems.

For the next milestone, we will work on the two super classes mentioned above, but will try to split the training data and test data, by giving few sub classes as inputs for CNN and putting the rest in test set. This was we want to see analyse how robust the build Machine Learning model is and further tune the model.

# **Reference Online links:**

1. <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>
2. <https://towardsdatascience.com/a-walkthrough-of-convolutional-neural-network-7f474f91d7bd>
3. <https://skymind.ai/wiki/multilayer-perceptron>
4. <https://medium.com/@mohtedibf/in-depth-parameter-tuning-for-knn-4c0de485baf6>
5. <https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd>