Object Classification in CIFAR 10

Sona Praneeth Akula, Pratik Satapathy, Jash Dave, Jayant Golhar
IIT Bombay

May 2, 2017

Contents

Abstract

Unprecedented expansion and revolution of world wide web has made it very difficult for search engines to index data, more specifically images. Therefore, it has become increasingly important to classify images in order to store them, tag them and search them. With this objective in mind, we in our project try to develop a method to classify images in a well-known data-set i.e *CIFAR 10*. Although there are already proven efficient ways to achieve the objective we here try to implement various algorithms of machine learning and compare their behaviour and performance for a deeper understanding.

Keywords - *CIFAR 10, Histogram of Oriented Gradients (HOG), Cross validation, Support Vector Machine (SVM), Multilayer perceptron (MLP), Convolution Neural Network (CNN), Residual Network, Inception, Transfer learning*

1 Introduction

Since the rise of high performance systems, parallelized algorithms for large matrix computation and use of GPUs in data processing, machine learning has altogether taken a generation leap. With use of large neural networks cuber-some image processing tasks are being performed in a flash. The applications of machine learning and deep neural networks in computer vision has increased many folds. As per our problem statement we will concentrate on the computer vision segment of machine learning.

Computer vision deals with mainly three activities. Those are

- **Data acquisition:** Data gathering is performed mostly by sensors. In the case of images digital camera is used to capture images or we can use datasets publicly available on the internet.
- **Data representation:** This deals with the technique of storing the data, removing irrelevant parts of data, filtering more significant portions, amplifications and also loss-less representations.
- **Data classification:** Classification deals with the most difficult task i.e. decision making. In our case the decision is to classify objects e.g. separating planets from stars in space science.

Our work will be mostly concentrated on the part of object classification. Therefore, we intend to take a standardised existing dataset of images for our data acquisition phase. More details on dataset is present in section 2.1. We have used different classifier to train data and compare the accuracy of all the classifiers. A detailed discussion with methods will follow in section 4.

2 Work Done

2.1 Datasets

CIFAR 10 dataset [1] consists of 60000 $32 \times 32 \times 3$ color images of 3 channels (red, green and blue) pertaining to 10 classes, with each class having 6000 images. Out of the 60000 images, there are 50000 train images and 10000 test images in the standard dataset. The classes present in the dataset are *airplane, automobile, bird, cat, deer, dog, frog, horse, ship* and *truck*. Classes are mutually exclusive and there is no multi-class multi-label case for the test data.

Figure 1: CIFAR 10 train dataset sample

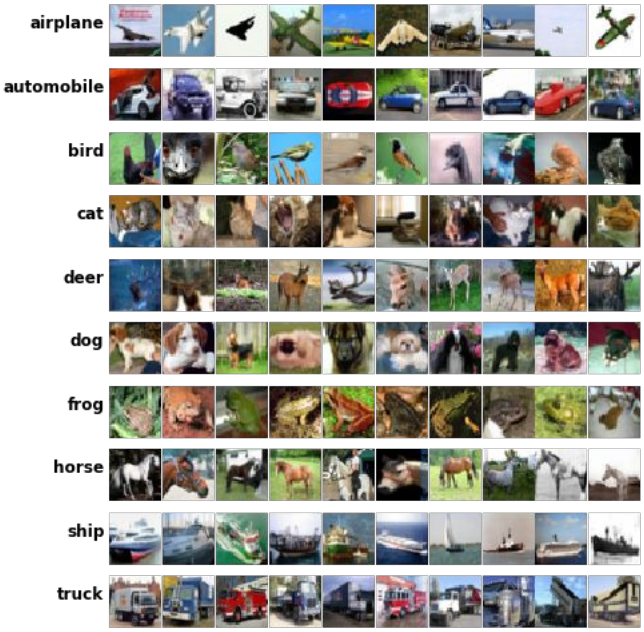

Figure 2: CIFAR 10 test dataset sample

2.2 Procedure

Every machine learning procedure goes through some specified steps that are discussed below:

2.2.1 Creating a train dataset

We have downloaded the CIFAR 10 dataset from the following link and used the original train test partition for all evaluations.

2.2.2 Extracting important features

We have used different feature extraction methods such as Histogram of Oriented Gradients (HOG) [2] and convolution. We have standard normalised our data before performing the training.

2.2.3 Training the classifier

We have used classifier algorithms namely

- Linear classifier
- Support vector machines
- Multilayer perceptron
- Convolutional Neural network

2.3 Features

For capturing information about images, we need to extract certain features which are distinctive to the image. We will discuss some features here.

2.3.1 Raw input

These features are nothing but the input image treated as a vector of pixels flattened as one dimensional array. For CIFAR 10 dataset where the image size is $32 \times 32 \times 3$, the feature vector would be of size 3072.

2.3.2 HOG features

Histogram of Oriented Gradients [2] has revolutionised the state of art human detection methods in 2005. This method was introduced by Dalal and Triggs and was used for human detection in images. In this algorithm, color image is converted to grayscale image. The image is divided into blocks of size $n \times n$ where n is usually in order of 2. Each block is then divided into cells of size $n' \times n'$ where $n' < n$. We calculate the gradients of these cells and organise them into k bins where each bin corresponds to the histogram of votes for that particular direction. Doing this would help in detecting a strong visual outline of the object. Finding the best n, n', k are vital for the HOG features to look better.

2.3.3 Image

Here we use the image as is without any modification

3 Algorithms Implemented

3.1 Linear classifier

Linear classification is based on the linear regression model i.e. $Y = Wx + b$ where W represents the weight matrix and b represents bias. x is the input features and Y returns output which is in our case a classification of images. We then calculate the softmax cross entropy of the obtained transformation Y and estimate the class labels as the one with highest probability. We also apply an L_2 regularizer to avoid overfitting.

3.2 Support Vector Machines (SVM)

Support Vector Machines [3] have been introduced by Cortes et al. in the year 1995. SVMs are classifiers which operate like a linear classifier but they strictly maintain a breathing space between the two classes and ensure that the gap stays at a maximum possible value. SVMs can perform non linear tasks as well by using the kernel trick, i.e mapping non linear features to a higher dimensional linear feature space.

3.3 Multilayer perceptrons (MLP)

Multilayer perceptron aka MLP is an extension of linear perceptron where non linear functions such as *sigmoid*, *tanh*, $ReLU$ etc., are used as activation functions in the intermediate neurons. MLP can classify data that are not linearly separable due to the non linear activation functions in the hidden layers. In the final layer we apply softmax function to predict the class corresponding to the input data. It has been observed that a single layer is equally powerful as a cascade of layers in this algorithm.

3.4 Convolution neural network

Convolutional neural networks are a special kind of network which are primarily used for data that are locally correlated like pixels of an image. The convolution steps reduce the input space by enabling weight sharing to make the whole input features tractable and meaningful for the neural network to learn.

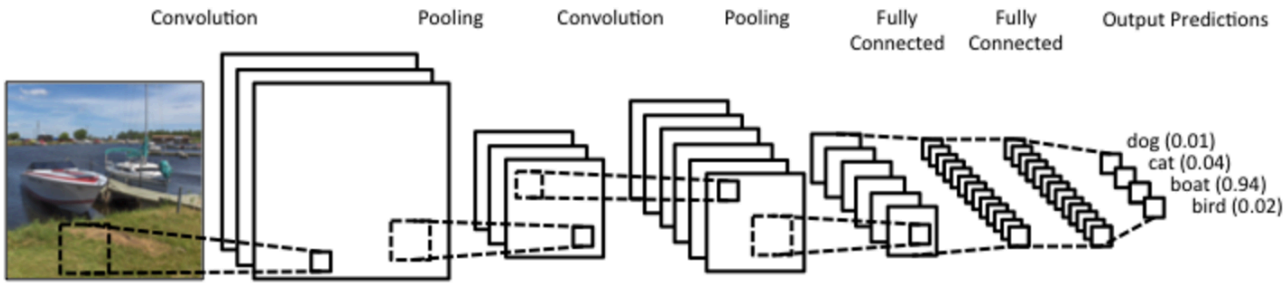

Figure 3: Convolution Neural Network [4]

3.4.1 Data Augmentation

Data Augmentation means adding new dataset to the training dataset and increasing the dataset size. We have used online process of augmenting dataset where we add new dataset for every image at every iteration. We have currently used random flip left right 14 and random rotation 15 of train dataset with maximum rotation of 20 degrees.
The figure 3 refers to a sample idealogy of CNN architecture. Usually a CNN has a convolution layer connected to input data layer followed by a pooling layer or a stack of more convolution layers. This process is repeated which finally ends in a bunch of fully conncted layers which can be seen towards the end of the diagram.

3.5 Residual network

Microsoft Research Asia comes up with 152 layer network called ResNet(Residual Network) [5]. Residual block involves blocks of conv-relu-conv series. In general, output from one layer is forwarded to next layer during forward propagation. But in ResNet, input from previous residual block is feed to the next block. Directly feeding output from one block to another does not keep any information about previous input going forward, so this is one advantage ResNet has over traditional CNN. Here the original input has a greater influence over the rest of the network.

3.6 Transfer learning

Transfer learning is the procedure where we need not train a convolution neural network from scratch but use an already trained model and use that model to train for our dataset. This method is successful as values learnt by convolution neural network in initial layers will tend to remain same whatever be the network. We only need to train the last fully connected layers to be able specific to our dataset.

3.6.1 Estimate accuracy on unseen test data

Using the trained model we predict the classes for unseen test data and compare it with the real test labels to check the accuracy of our implementation.

4 Experiments

We have performed our feature extraction using two types of features. One feature is that we directly feed the raw input pixels flattened as a vector to the classifier. Second kind of feature is called Histogram of Oriented Gradients (HOG) which was used for human detection. We have used a block size of 8×8, cell size of 4×4 and number of bins as 9 for our HOG feature as this gave the best results. All the code has been developed in Python 3 using *tensorflow*, *sklearn* & *tflearn*

4.1 Methods

4.1.1 Linear Classifier

Classifier pipeline: Raw input pixels/HOG features → Pre-process with StandardScaler → $Wx + b$ → Softmax with cross entropy → One hot labels → Class labels

We have applied L_2 loss as the regularizer and trained using gradient descent.

Best accuracy with raw input features 0.4083

Best accuracy with HOG features 0.5311

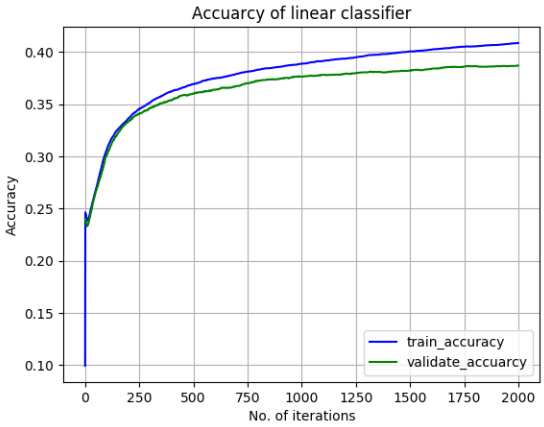

Figure 4: Accuracy of Linear Classifier

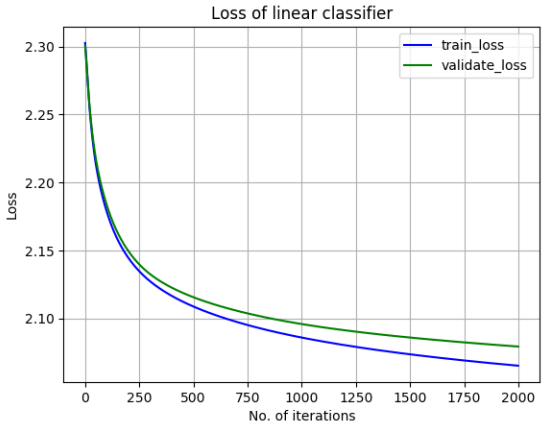

Figure 5: Loss of Linear Classifier

4.1.2 Support Vector Classifier

Classifier pipeline: Raw input pixels/HOG features → Pre-process with StandardScaler → Radial basis kernel → SVM → Class labels

Best accuracy has been obtained using radial basis function kernel indicating that the data is non-linear.

Best accuracy with raw input features 0.5432

Best accuracy with HOG features 0.6193

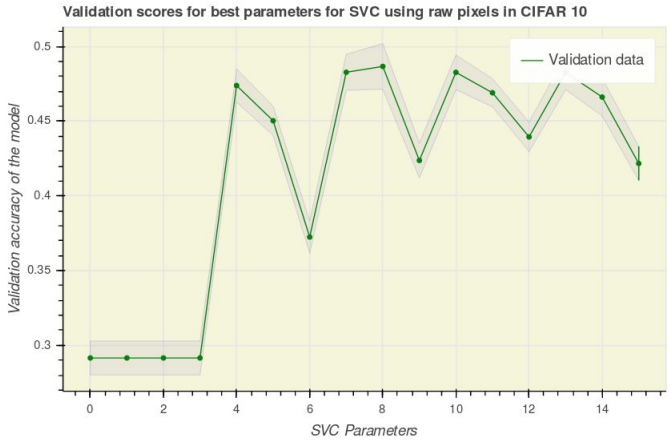

Figure 6: SVM classifier with raw pixel vector validation accuracy

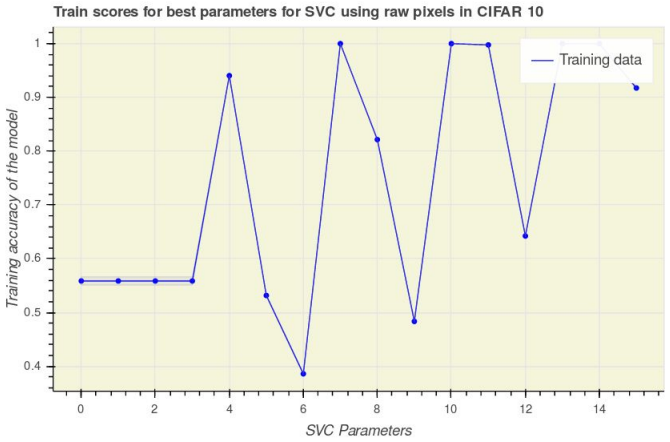

Figure 8: SVM classifier with raw features train accuracy

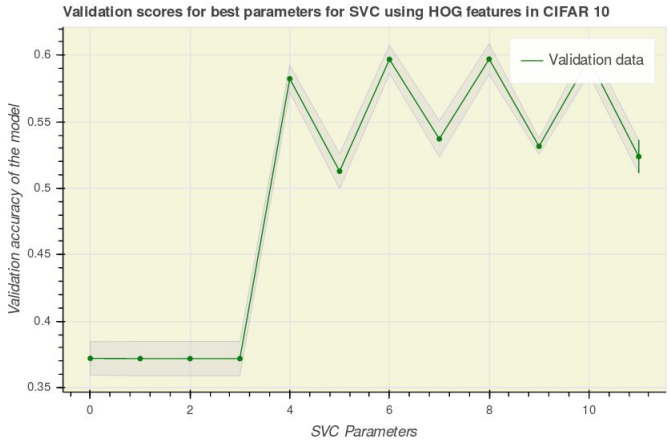

Figure 7: SVM classifier with HOG features validation accuracy

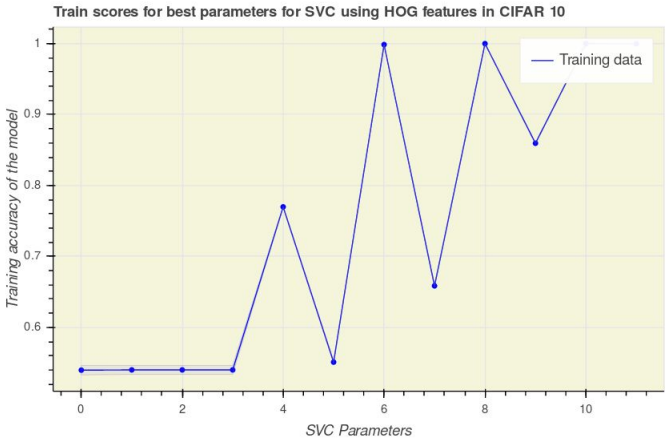

Figure 9: SVM with HOG classifier train accuracy

4.1.3 Multi Layer Perceptron

Classifier pipeline: Raw input pixels/HOG features → Pre-process with StandardScaler → Hidden layer 1 with 3072 hidden units → Hidden layer 2 with 3072 hidden units → Softmax with cross entropy → One hot labels → Class labels

Best accuracy with raw input features 0.5168

Best accuracy with HOG features 0.5621

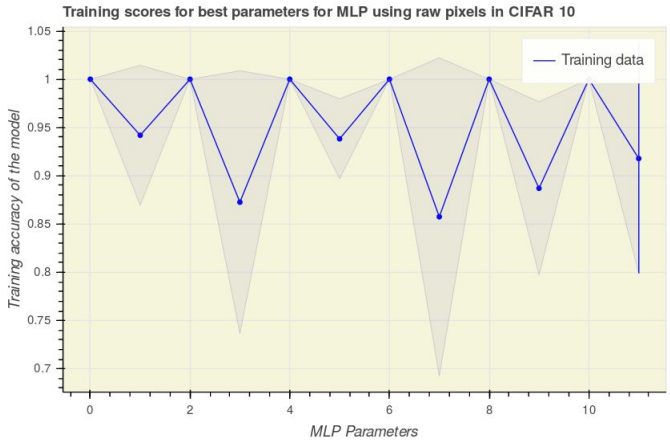

Figure 10: MLP classifier with raw pixels train accuracy

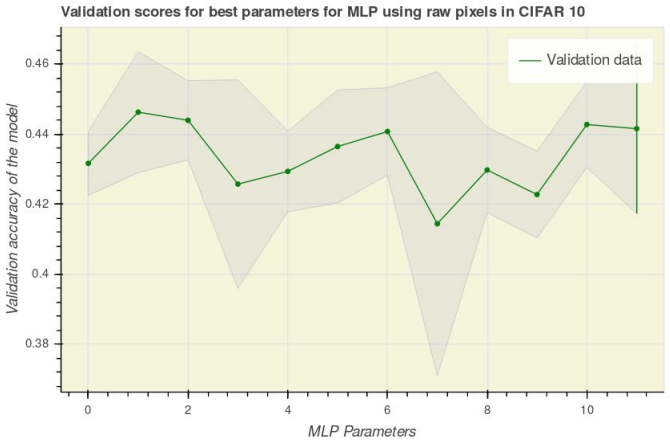

Figure 11: MLP classifier with raw pixels validation accuracy

4.1.4 Convolution Neural Network

Classifier pipeline: We've implemented a simple CNN architecture which can be seen in the figure 12. The architecture has been run with and without data augmentation. When augmenting new data to the train set we've flip left-right each image and added a random rotation of maximum 20 degrees to each train image. Input image data has been preprocessed with standard scaling method and fed to the classifier.

Best accuracy without data augmentation: 0.745

Best accuracy with data augmentation: 0.7861

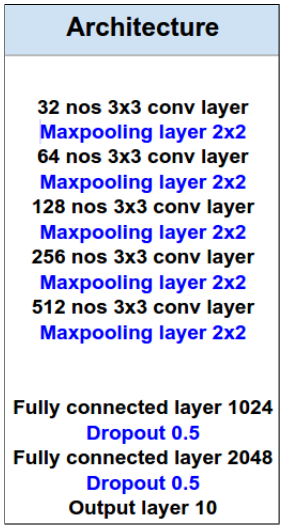

Figure 12: Simple cnn design

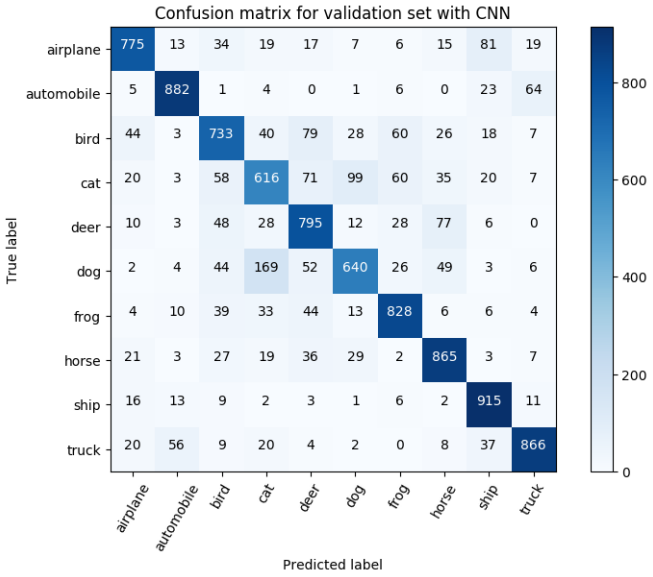

Figure 13: CNN Confusion Matrix

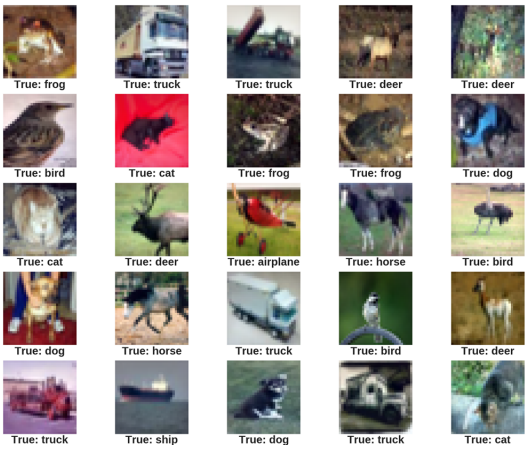

Figure 14: Random Flipping of train dataset

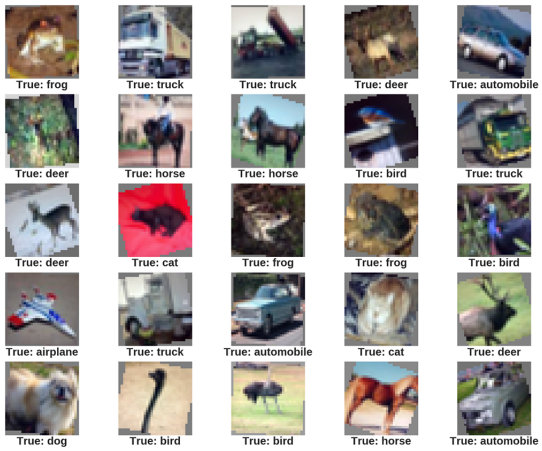

Figure 15: Random rotation of train dataset with a maximum angle of 20 degrees

Visualization of layers

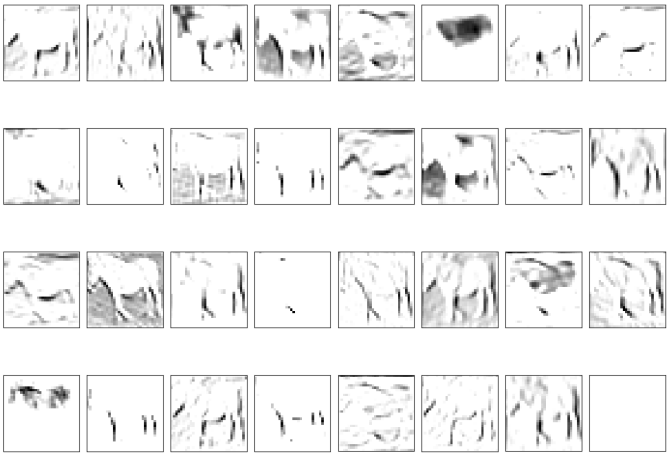

Figure 16: Convolution layer 1 visualization for a horse. 32 convolution layers have been plotted

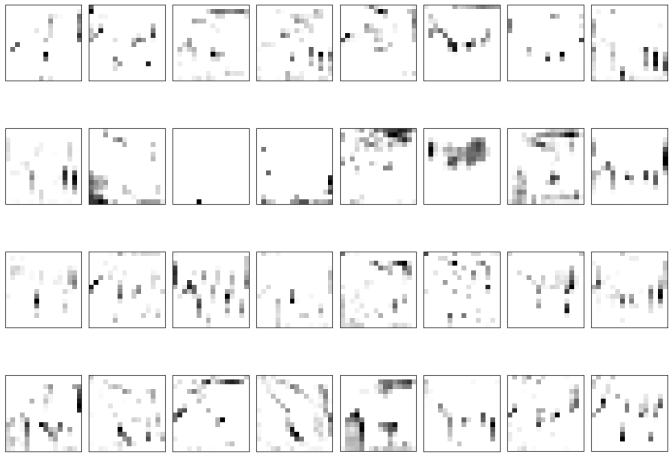

Figure 17: Convolution layer 2 visualization for a horse. 32 convolution layers have been plotted

4.1.5 Non Residual Architecture

With reference to the ResNet layer in section 4.1.6 we evaluate a simpler design with simple non residual conv-relu-conv design consisting of 4 nos of 32 (3x3) conv layer followed by 4 nos 0f 64 (3x3) conv layer with a 512 neurons fully connected layer and a 10 neuron fully connected layer. The intention of this experiment was to differentiate performance with the similar ResNet architecture.
Figure 18 shows our implemented residual architecture.

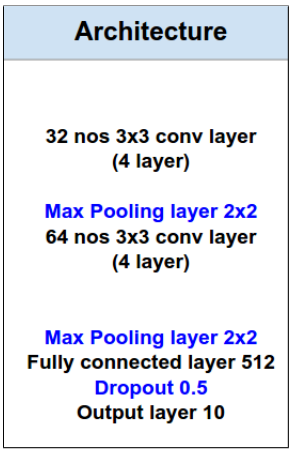

Figure 18: Smaller residual design [5]

Best accuracy 0.83

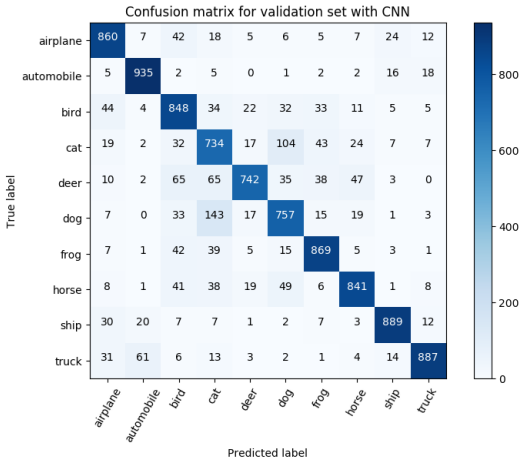

Figure 19: Non Residual Network Confusion Matrix

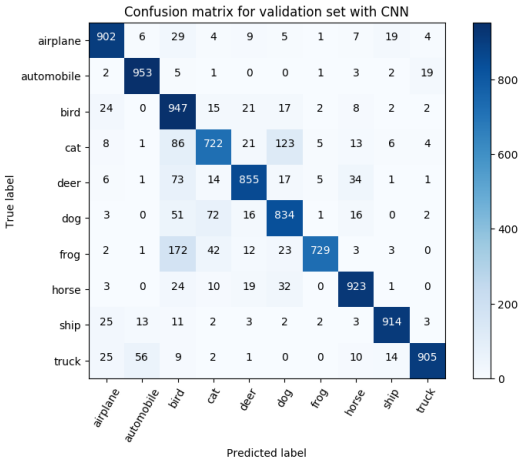

Figure 20: Residual Network Confusion Matrix

4.1.6 Residual Network

We formed a smaller ResNet CNN architecture with only 32 layers instead of the original 152 layer. As observed previously that data augmentation has increased accuracy, we've used data augmentation for our residual network also. Figure 21 shows a single residual block. Figure 22 shows our implemented residual architecture which is designed by using residual block implemented in the tflearn library of tensorflow.

Best accuracy 0.87

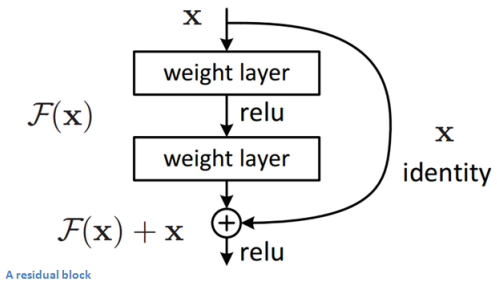

Figure 21: Residual Block [5]

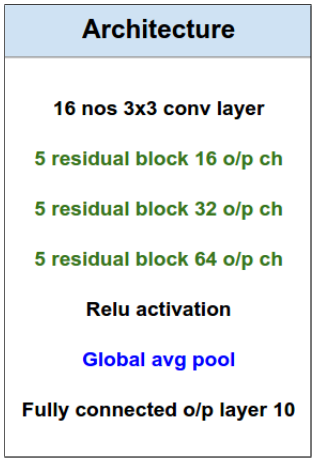

Figure 22: Smaller residual design [5]

4.1.7 Transfer learning with inception model

Inception model [6] is a convolution neural network developed by Szegedy et al. which has been trained on ImageNet dataset which contains 15 million images for 22,000 different objects. The final three layers are removed from inception model and output from transfer layer is feed to the fully connected neural network. So instead of retraining the network from scratch, we use the output from already trained inception model to get better accuracy. In Inception Model, we will take output from max pooling, convolution operation instead of just taking the output from only one. This will result in too many output, so use 1×1 filter for dimension reduction.

Best accuracy with transfer learning features 0.91

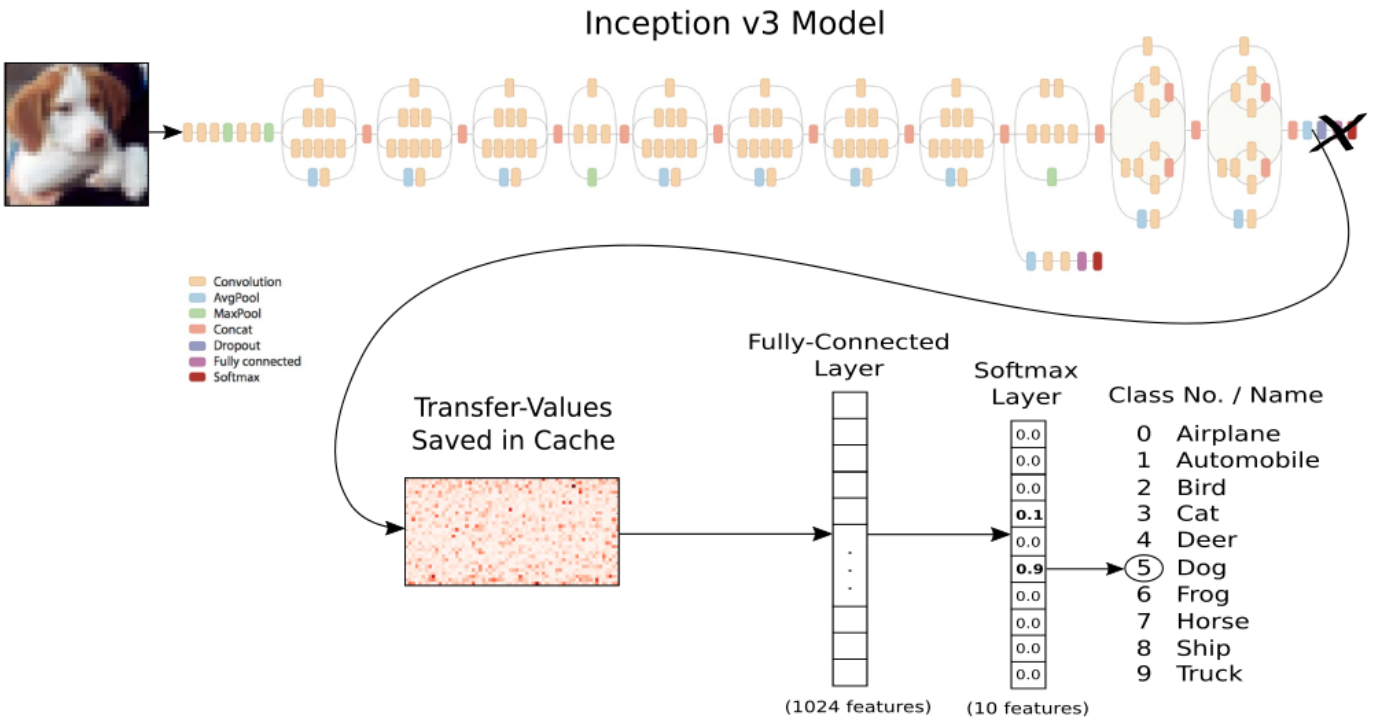

Figure 23: Inception Model [7]

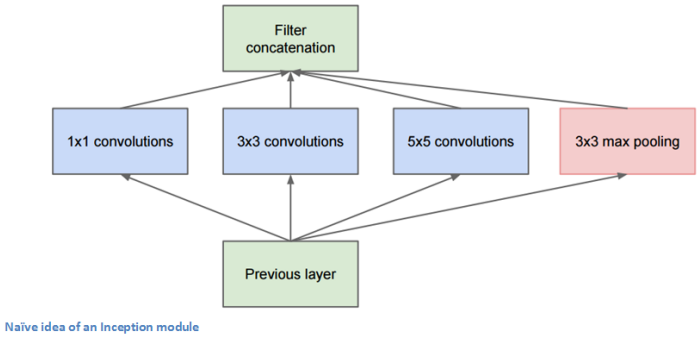

Figure 24: Naive Inception Model Block Diagram [6]

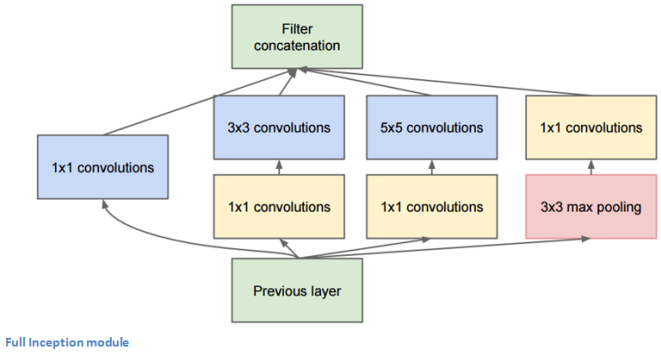

Figure 25: Inception Model Block Diagram [6]

5 Observations

- We have observed that using HOG features has improved test accuracy as opposed to using raw input pixels for traditional machine learning approaches.
- Residual network has provided the best accuracy (almost equivalent to transfer learning)
- Also we have observed that initialization of weights with a particular criteria has resulted in faster jump in accuracy values. We have used *random_normal* weights initialization with mean 0 and stddev 0.02.
- CNN's work much better when they're trained with augmented data

6 Conclusion

Here's a summary of experiments we have performed and results obtained.

Table 1: Summary of experiments

Exp. No.	**Input features**	**Classifier**	**Test accuracy**
1	Raw pixels	Linear	0.4083
2	HOG	Linear	0.5311
3	Raw pixels	SVM	0.5432
4	HOG	SVM	0.6193
5	Raw pixels	Multi layer perceptron	0.5168
6	HOG	Multilayer perceptron	0.5621
7	Raw image without data aug.	CNN	0.745
8	Raw image with data aug.	CNN	0.8058
9	Raw image with data aug.	ResNet	0.8684
10	Raw image	Inception	0.9015

7 Acknowledgements

We would like to express our deep gratitude to our TA mentors Abhishek Sethi and Rishabh Dhabral along with our professor Ganesh Ramakrishnan for guiding us steadily though this project. Their valuable suggestions and discussions helped in understanding the convolution neural networks better.

8 Contributions

- Sona Praneeth: Coding Libraries for dataset / accuracy / confusion mat /HOG features etc , SVM models, MLP models, Linear models, CNN models with TF, report preparation.
- Pratik Satapathy: CNN models using a TF-learn model, non residual TF-learn model, residual model (ResNet), presentation preparation, report preparation.
- Jash Dave: Transfer learning using GoogLeNet(Inception v3), presentation preparation, report preparation.
- Jayant Golhar: SVM various model validations, presentation preparation, report preparation.

References

[1] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[3] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[4] http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/11/screen-shot-2015-11-07-at-7.26.20-am.png.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[7] https://github.com/hvass-labs/tensorflow-tutorials/blob/master/08_transfer_learning.ipynb.