Machine Learning Engineer Nanodegree

Capstone Project Report

Image Recognition using CNN

Sai Chavali

June 25th,2018

I. Definition

Project Overview

Image recognization is the ability of computer to interpret a thing from images, documents and various other sources. The core idea came from the computer vision(how do a computer recognize whether its a car or human or dog).Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification, localization and detection. Recent developments in neural network approaches have greatly advanced the performance of these state-of-the-art visual recognition systems.

GPU-based implementations of this approach won many pattern recognition contests, including the IJCNN 2011 Traffic Sign Recognition Competition,the ISBI 2012 Segmentation of neuronal structures in EM stacks challenge,the ImageNet Competition and others. The dataset used is cifar10 which is subset to cifar100 dataset.Both are them serve well for the beginner in deep learning pretty well.

Reference: www.en.wikipedia.org/wiki/image-recognization

Problem Statement

We have to predict what is there in an image. The dataset contains images of frogs,automobiles,horse and 7 others images which are used to train the data. This is a supervised learning problem.It falls under classification problem containing 10 classes(0-9).Any classification algorithm(like SVM, randomforest, DecisionTree)may be used to the problem.Our model is tested by its ability to correctly predict different real world and unseen data.I have chosen A convolution neural network because we have large number of features in the dataset(every pixel is considered as feature).If there are more features and we have to get best accuracy and visual imagery is involved always better go with neural nets.

Metrics

The performance metric I used is accuracy because it measures the ratio of correct predictions to the total number of cases evaluated.In our case it means the ratio between correctly predicted images to the total number of images examined.Its range varies from 0 to 1 where 0 implies worst and 1 implies best performance.The most used classification metrics are accuracy and f1_score but f1_score is mostly used for binary targets which is not corresponding to our project.So I preferred accuracy.

Accuracy=number of samples predicted correctly/total num. of samples evaluated

II. Analysis

Data Exploration

The cifar10 dataset consists of 10 categories of images as follows:

1. Aeroplanes
2. Automobiles
3. Birds
4. Cats

5. Deer
6. Dogs
7. Frogs
8. Horses
9. Ships
10. Trucks

It consists of 50,000 training samples and 10,000 testing samples.It can be said as subset to cifar100.The images in the dataset are sized normally(32*32).The dataset contain colored images(channels =3).The dataset can be loaded using keras module by running the following code

```
from keras import cifar10

(X_train,y_train),(X_test,y_test)=cifar10.load_data()
```

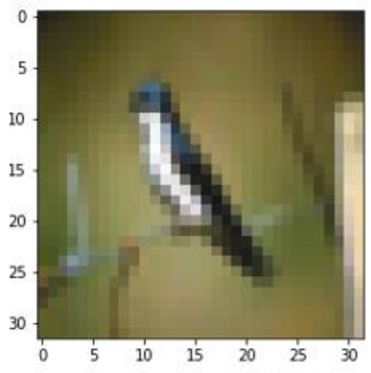
- X_train,X_test:unit8 array of RGB image data with shape(num_samples,3,32,32)
- y_train, y_test:unit8 array of categoriacal labels(integers in range(10))with shape(num_samples).

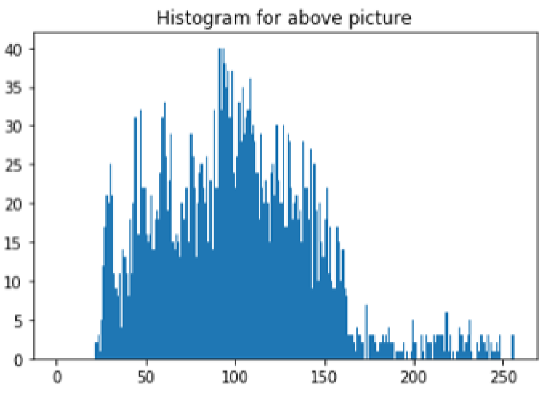
We train our model with pixel-values of image as it’s features.We can measure the performance of the model by testing with X_test and y_test.

Reference: https://www.kaggle.com/c/cifar-10/data

Exploratory Visualization

The image of the dataset looks as below.It contains 32*32 pixels as represented on x and y-axis.The imae is rgb image i.e,it contains three color channels where each values ranges from 0-255





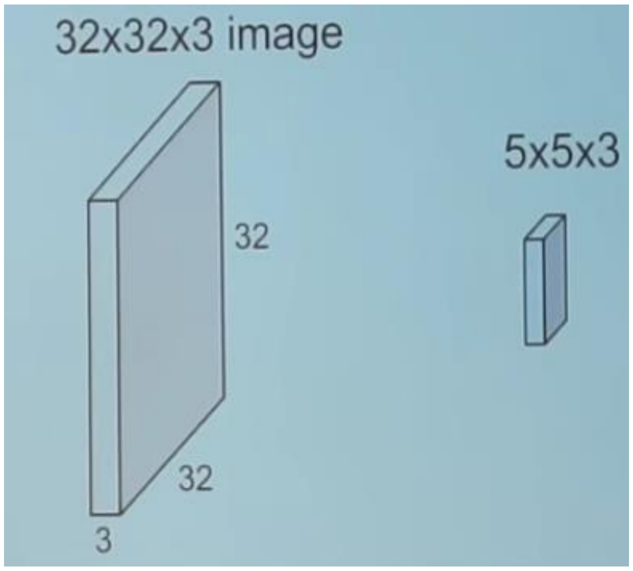
An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. Histogram is a graphical representation of the intensity distribution of an image.Histogram quantifies the number of pixels for each intensity value.The observation that can be made by looking at above histogram there are lot of pixels with the intensity of 50-150 peaking at 100.

Algorithms and Techniques

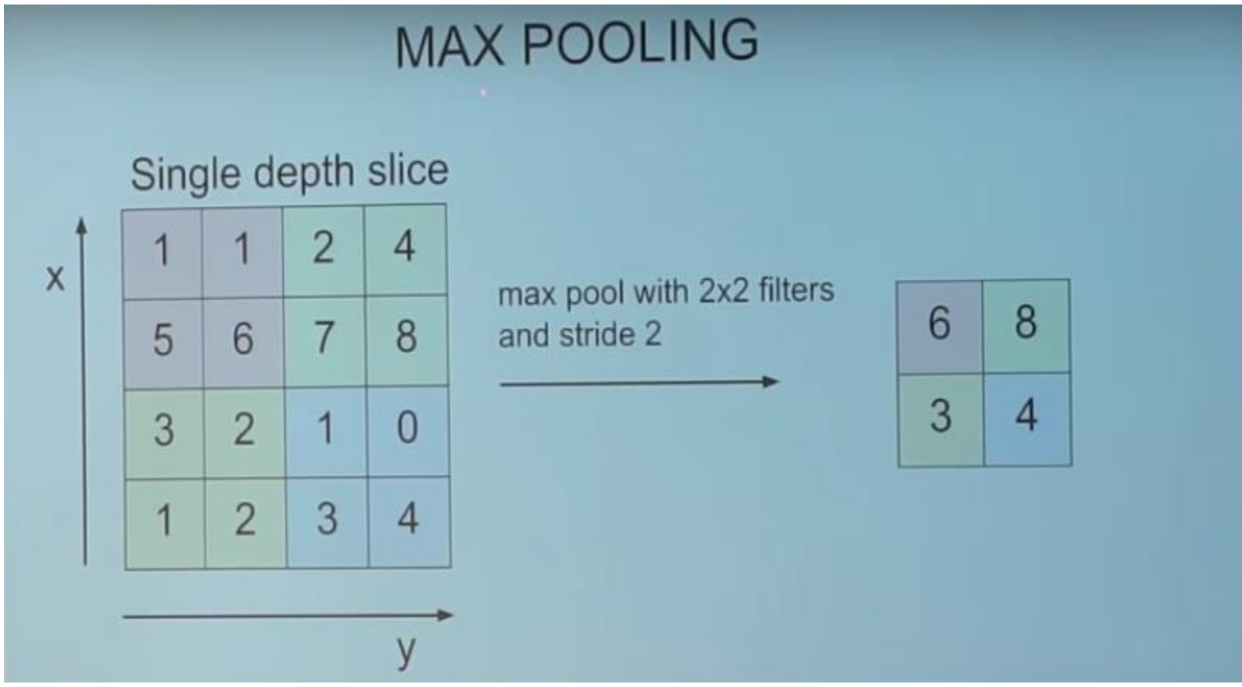
As I have mentioned it earlier as the problem belongs to supervised learning and particularly classification.Lot of algorithms are available for classifying the data.I have chosen Convolution Neural Network as my model

Convolution neural network is a type of feed-forward artificial neural network most commonly applied for visual imagery.As our problem belong to visual imagery we can use CNN here.Most commonly used layers to architect a CNN are:

1) Convolution layer : The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting.It consists of set of learnable filters.Every filter is small spatially ,but extends through the full depth.During the forward pass,we convolve each filter across the width and height and compute dot products between filter entries and input.AS we convolve,we will produce a 2-dimensional activation map.Intutively,the network will learn the filters that activate when they see some type of visual feature.



2) Pooling layer : It's a function to progressively reduce the spatial size to reducing the amount of parameters and the computational complexity.Pooling layer operates on each map independently.The most common approach is max pooling.



3) ReLU layer :It's a non-linearity which is applied similar to neural network.It applies an element wise activation function like max thresholding.
4) Dense layer :It computes the class scores.Neurons in fully-connected layer have connections to all activations in the previous layer.Their activations can hence be computed with a matrix multiplication followed by a bias offset.
5) Dropout :It is meant for dealing with overfitting.The core idea is to drop some connections in the network.
6) Softmax layer:It is final output in a neural network that does multi class classification.It outputs a normalized class probabilities.In softmax classifier the function stays constant,we interpret the scores as unnormalized log probabilities and replace the hinge loss with cross-entropy loss that has the following formula:

$$L_i=-\log(e^{f_i}/\sum_j e^{f_j})$$

f_j :class scores f of j-th element in the vector.

The softmax function has the following formula:

$$F_i(x)=e^{z_i}/\sum_k e^{x_k}$$

The loss functioin (cross-entropy) between a true distribution and an estimated probability q is defined as:

$$H(p,q)=-\sum_x p(x)\log(q(x))$$

The softmax classifier actually minimizing the loss function where all probabilities masses are correctly classified(normalizing).In leyman terms the cross-entropy wants the predicted distribution to have all its mass o the correctly classified one.

Adaptive Moment Estimation Optimizer(Adam):It computes adaptive learning rates for each parmeter. Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

Stochastic gradient descent maintains a single learning rate(alpha)for all weight updates and the learning rate doesnot change during training.

The main advantages of adam is its adaptive grading algorithm that maintains a per-parameter learning rate that improves performance on problems with sparse gradients .The learning rates are adopted based on average of recent magnitudes of gradients for weight.

References:

http://cs231n.github.io/convolutional-networks/#fc

https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

Benchmark

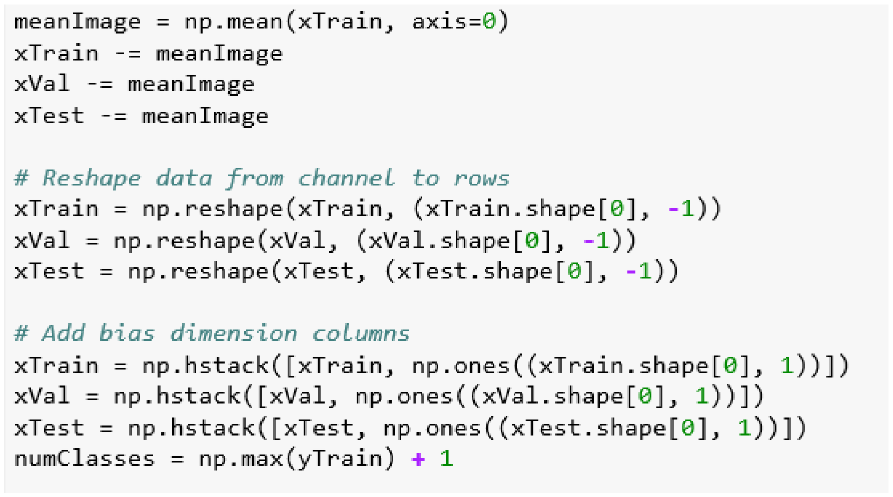
Initially I used SVM model as a benchmark model and fitted the model with our training data and calculated the accuracy score which I got about 0.30. More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback. So we can apply SVM for our dataset.Keeping this a benchmark accuracy I tried to build my model in such a way that it would give an accuracy score more than that model.

Reference: https://en.wikipedia.org/wiki/Support_vector_machine

III.Methodology

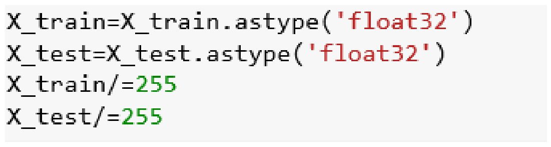
Data Preprocessing

I have done preprocessing two times one for benchmark model(SVM) where we don't have 3d image representation so we should preprocess in a different way which involves normalize the data,reshaping data from channel to rows and adding bias dimension columns and the other



```
meanImage = np.mean(xTrain, axis=0)
xTrain -= meanImage
xVal -= meanImage
xTest -= meanImage

# Reshape data from channel to rows
xTrain = np.reshape(xTrain, (xTrain.shape[0], -1))
xVal = np.reshape(xVal, (xVal.shape[0], -1))
xTest = np.reshape(xTest, (xTest.shape[0], -1))

# Add bias dimension columns
xTrain = np.hstack([xTrain, np.ones((xTrain.shape[0], 1))])
xVal = np.hstack([xVal, np.ones((xVal.shape[0], 1))])
xTest = np.hstack([xTest, np.ones((xTest.shape[0], 1))])
numClasses = np.max(yTrain) + 1
```

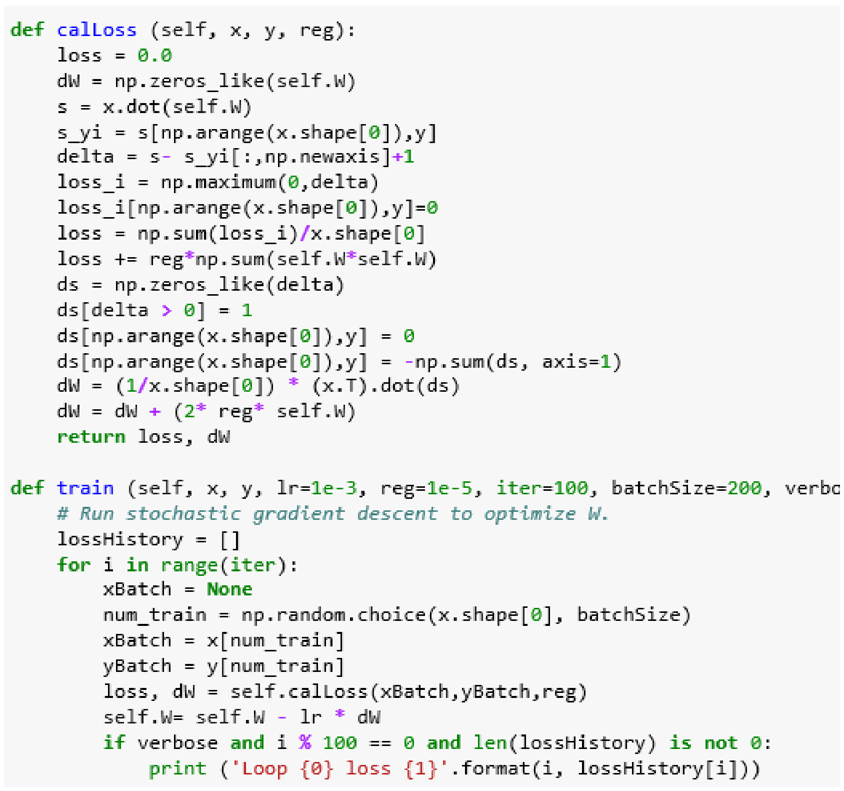
for my CNN where we preprocess the data in a go the color channel of each pixel varies from 0 to 255,so inorder to reduce computational load we divide every pixel by 255 such that it ranges between 0-1



```
X_train=X_train.astype('float32')
X_test=X_test.astype('float32')
X_train/=255
X_test/=255
```

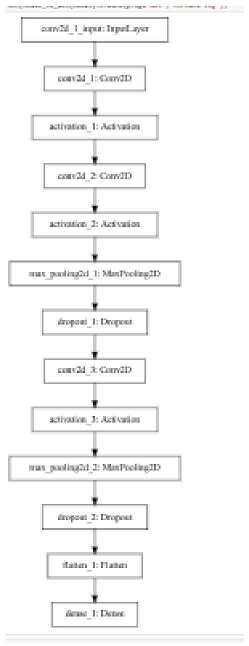
Implementation

First step I have done after preprocessing the data is feeding the data into SVM classifier and fitting the model.I learned how to train a image pixel data on SVM classifier.I have calculated the accuracy score of the SVM model which I have chosen as a benchmark.It gives an accuracy score of about 0.35 which in my opinion is bad.Then I started building my model architecture after preprocessing.My model looks like below.I have faced lot of difficulty in training the benchmark SVM as I haven’t worked with images in SVM.It took me hours to understand the calculateloss(calLoss) function and I have tried for codes but I didn’t understood most of them slowly by knowing it into deeper helped me and it clicked.



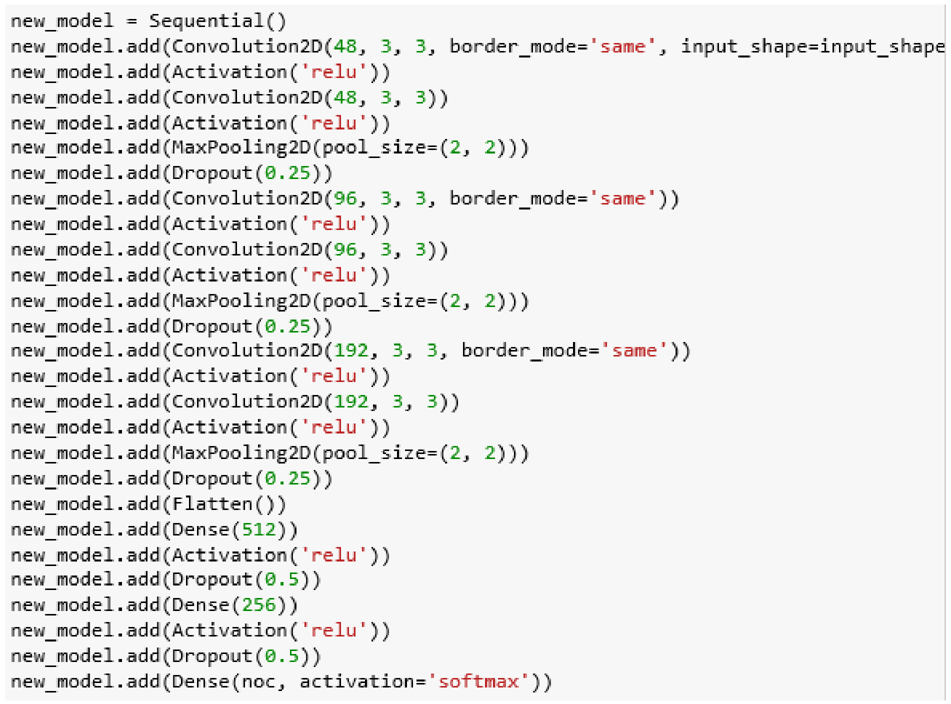
```python
def calLoss (self, x, y, reg):
    loss = 0.0
    dW = np.zeros_like(self.W)
    s = x.dot(self.W)
    s_yi = s[np.arange(x.shape[0]),y]
    delta = s- s_yi[:,np.newaxis]+1
    loss_i = np.maximum(0,delta)
    loss_i[np.arange(x.shape[0]),y]=0
    loss = np.sum(loss_i)/x.shape[0]
    loss += reg*np.sum(self.W*self.W)
    ds = np.zeros_like(delta)
    ds[delta > 0] = 1
    ds[np.arange(x.shape[0]),y] = 0
    ds[np.arange(x.shape[0]),y] = -np.sum(ds, axis=1)
    dW = (1/x.shape[0]) * (x.T).dot(ds)
    dW = dW + (2* reg* self.W)
    return loss, dW

def train (self, x, y, lr=1e-3, reg=1e-5, iter=100, batchSize=200, verbo
    # Run stochastic gradient descent to optimize W.
    lossHistory = []
    for i in range(iter):
        xBatch = None
        num_train = np.random.choice(x.shape[0], batchSize)
        xBatch = x[num_train]
        yBatch = y[num_train]
        loss, dW = self.calLoss(xBatch,yBatch,reg)
        self.W= self.W - lr * dW
        if verbose and i % 100 == 0 and len(lossHistory) is not 0:
            print ('Loop {0} loss {1}'.format(i, lossHistory[i]))
```

Initially I am taking 48 filters for the first convolution layer with kernel size=3,3 whose output is fed into input as ReLU layer which applies elemental activation and similar kind of layers are repeated by with maxpooling layer which downsamples the data.At last there is fully connected layer of size 10 which follows softmax activation layer for classification



I have used sgd optimizer and metrics as accuracy and compiled my model before I fitting it.The network is trained on training data evaluated against the test data.The accuracy score we have got is around 0.49

The best part of this project is knowing how well my model works on unseen data.So after the final model I have taken some images and tested whether my model correctly predicts it or not but before doing this we have to do preprocessing on the image uploaded by us.



```
new_model = Sequential()
new_model.add(Convolution2D(48, 3, 3, border_mode='same', input_shape=input_shape
new_model.add(Activation('relu'))
new_model.add(Convolution2D(48, 3, 3))
new_model.add(Activation('relu'))
new_model.add(MaxPooling2D(pool_size=(2, 2)))
new_model.add(Dropout(0.25))
new_model.add(Convolution2D(96, 3, 3, border_mode='same'))
new_model.add(Activation('relu'))
new_model.add(Convolution2D(96, 3, 3))
new_model.add(Activation('relu'))
new_model.add(MaxPooling2D(pool_size=(2, 2)))
new_model.add(Dropout(0.25))
new_model.add(Convolution2D(192, 3, 3, border_mode='same'))
new_model.add(Activation('relu'))
new_model.add(Convolution2D(192, 3, 3))
new_model.add(Activation('relu'))
new_model.add(MaxPooling2D(pool_size=(2, 2)))
new_model.add(Dropout(0.25))
new_model.add(Flatten())
new_model.add(Dense(512))
new_model.add(Activation('relu'))
new_model.add(Dropout(0.5))
new_model.add(Dense(256))
new_model.add(Activation('relu'))
new_model.add(Dropout(0.5))
new_model.add(Dense(noc, activation='softmax'))
```

My refined model code

Refinement

I added augmentation in data and improved zoom range to 0.2 using ImageDataGenerator function in keras .preprocesssing.image library and made my architect a bit deeper by increasing few more layers and adding dropout functions to ensure overfitting doesn't happen and chosen an adam optimizer to change the learning rate.Once the final model is built and compiled.It must be trained and after we evaluate the performance our model by calculating accuracy and I got around 0.82 which is good.

IV.Results

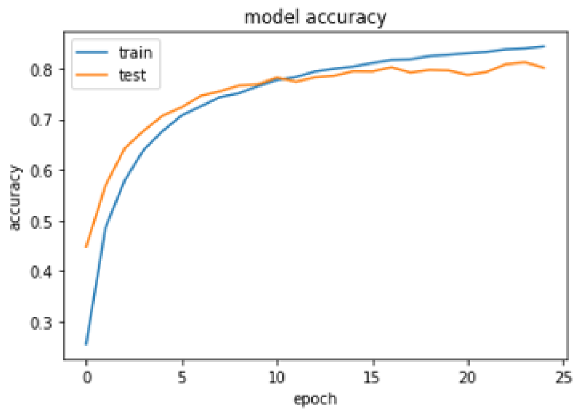
Model Evaluation and Validation

The model is evaluated against the test set.The final model is chosen in such a way that performs better(accuracy) than previous models.The accuracy is increased by changing the following parameters:

1) I have added few more layers to make my network deeper

2) I have changed the optimizer from sgd to adam to make changes to learning rate.
3) Data Augmentation.

The accuracy on training and testing data can be visualized by this graph



The model performed pretty well on the data.It have scored 84% accuracy which is a significant change.I think it doesnot overfit as we are augmenting data i.e,change image zoom value,rotations make the model learn rather than remembering.

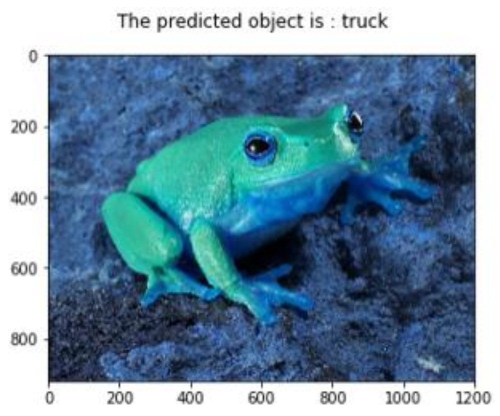
The robustness of the final model is evaluated by performing a test against the real world data.The observation made based upon that test out of 11 images it correctly classifies 7 images .This tells that our model is reliable and robust enough on real world data as well.Using dropout layers helped us to ensure small changes in the input doesnot effect the result.

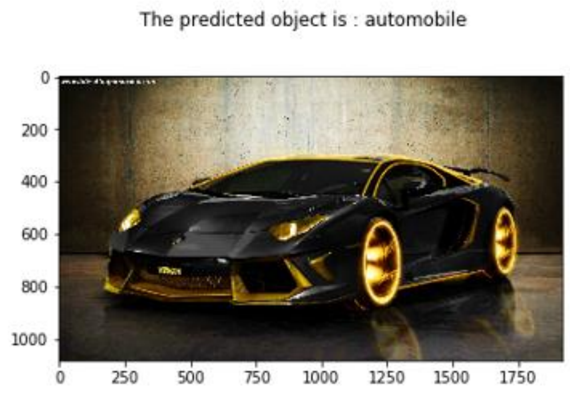
Justification

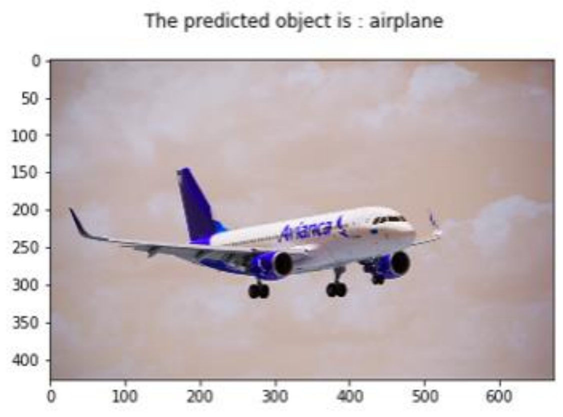
The accuracy score of our model is 0.84 and the accuracy score of my benchmark model is 0.37 which gives an evidence that our model clearly outperforms the benchmark SVM.This justifies that for image recognization Neural Nets are best suited .To make our justification stronger we tested our model with real world data and it also performs pretty well on unseen data.

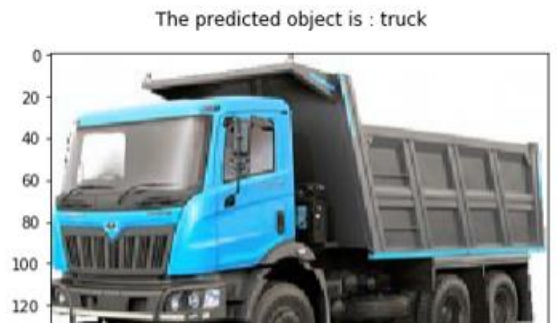
V.Results

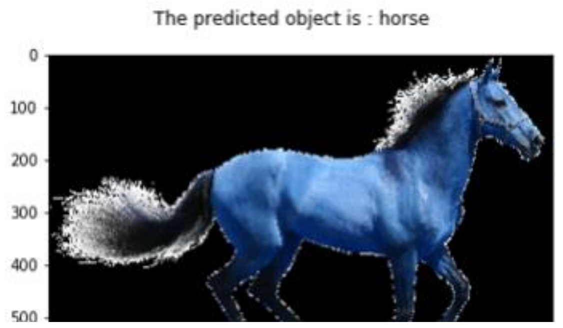
Free-Form Visualizations

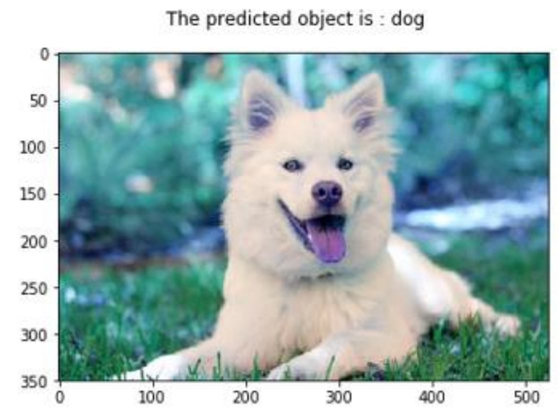


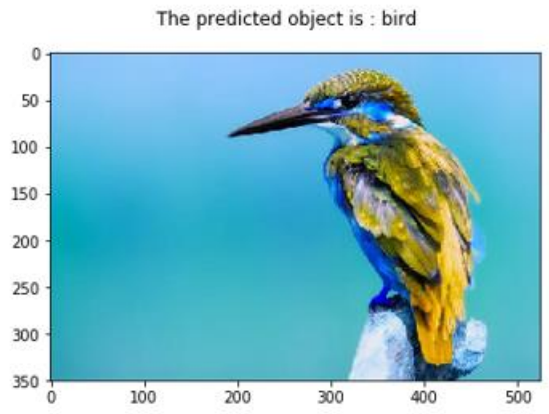












This is the unseen input fed to our model and our model performed well on real world unseen data(robust) and it is quite reliable due to high accuracy score .This reflects that our model is behaving correctly and I am satisfied with the output.

Reflection

I can describe this project summary as follows:

1) Importing the required modules
2) Loading the data,Data Exploration and Observation
3) Preprocesing the data according to the model which I am applying
4) Train the benchmark model(SVM) and find the accuracy score
5) Built our model using convolution neural network
6) Train the model using training set and evaluate the performance using testing data
7) Refine the model by increasing layers and changing the parameters inorder to find the optimal model architecture
8) A test is conducted against the refined model by loading the images which are unseen for the model and evaluate the performance of final model.

I started my work by importing various modules very importantly keras,numpy and loading the data and visualizing the data .Then I thought preprocesing the data to decrease the computatioinal complexity later I trained the SVM with the preprocessed data and calculated my benchmark model accuracy as I have already mentioned SVM gave me hard time and I don’t know even how to start i.e,how to fit the image pixel data into a SVM then I searched it online but haven’t understood anything then came back to the scikit documentation and observe SVC class again I looked for some code and tried to implement what I have understood so many codes I have seen along with documentations gradually I attained some knowledge about it and by implementing many times I got to know how my code is behaving and got the intent, then I built my own CNN , compiled,fitted the data and calculated accuracy score then I thought of refining my model a bit and successfully done it by adding more layers and changing the parameters.Then I finalized my model and tested with unseen data to check its reliability and performance on realworld and unseen data.I observed the contrast between my model and benchmark model and I was satisfied with my result.

Improvement

The one thing I thought of doing is to apply transfer learning with ResNext,Inceptionv3,VGG16 to the data and check the accuracy.I have done it partially but due to lack of resources(GPU) I had stopped it.I will comment my code with that implementation the ipynb file.Also we could improve accuracy by tuning hyper parameters using hyperas library.