CSCI 5521 Introduction to Machine Learning
Visual Recognition System

Team Members:
Rudragouda Pharale - 5183804 - phara001@umn.edu
Yuxiang Wang - 4981113 - wang5350@umn.edu
Jayapriya Surendran - 5225959 - suren009@umn.edu

University of Minnesota, Twin Cities

Guided by

Prof. Paul Schrater

Table of Contents

CHAPTER 1

INTRODUCTION

1.1 Objective:

With the development of new technologies every day, we are witness to various revolutionary products like personal assistant in phones, smarter home appliances, etc. These products make our lives simpler and better. However, despite decades of research, visual recognition is still considered as an unsolved problem. This part of computer vision has many diverse real world applications, ranging from video games to self-driving cars. However, it has also been traditionally very difficult to predict successfully, due to different factors (lighting, resolution, camera angles, color balance, etc.) that go into creating an image, variations among the images of same classes, same class images with different backgrounds etc.

This paper proposes a technique for the implementation of automatic image classification system which is robust to the variations like image artifacts, orientation of the object in the image, contrast, color saturation etc which can efficiently capture these variations and classify the images with improved accuracy.

1.2 Motivation

The human visual system is one of the inspiration behind this attempt of solving visual recognition problem, although we do not directly try to emulate it directly in approach but merely in the outcome. Besides this, many practical applications like Robotics, Medical, Security in the form of automated surveillance system, visual aid for visibly disabled people etc are inspirations behind this project.

1.3 Approach

The difficulty in image classification and object recognition arises due to humongous number of possible images. We need models to capture the variability of scenes and the variations in the objects given the limited memory resources. Also, even images with resolution as low as say 360x480, has 0.17M first degree features and adding 2nd, 3rd and other higher degree features would only exponentially increase the feature size, memory requirement, complexity of training and testing. Hence, simple algorithms like logistic regressions cannot be used for image classification system. Hence we zeroed on to use neural networks more specifically convolutional neural networks to model the system which can better capture these variabilities.

Literature Survey through various publications gave us an insight of different techniques that are currently being used to solve the problem of image classification. Most of the neural network including convolutional neural network based systems showed promising results for image classification problem.

1.4 Outline

This paper is divided into 4 chapters, the first of which is introduction, second chapter explains the training and testing the model, third chapter explains the results achieved and the last chapter explains the future scope, limitations and conclusion.

CHAPTER 2

TRAINING AND TESTING

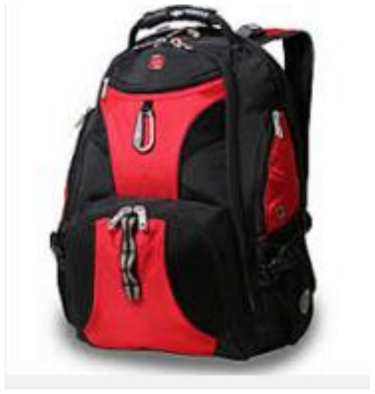
2.1 Image Dataset

We have used subset of openly available image dataset called “Caltect 256” which is a collection of 30607 images of various resolutions. This image dataset consist of 256 different classes of images like animals, objects, nature based images etc. Relatively smaller database with fewer classes was intentionally used considering the computation complexity involved in training, testing and experimenting with different algorithm in the given time. Subset of the dataset was used for training and subset was used for testing. Though the dataset of original images is smaller, we have also used the different version of processed original image to make the system robust to artifacts and variations among the images. This is described in more detailed in next section.

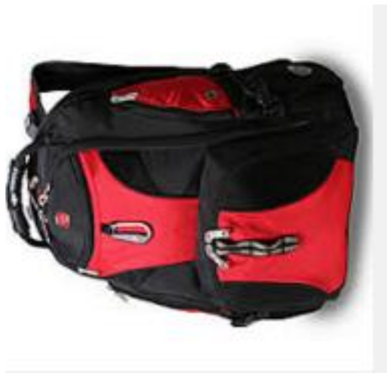
2.2 Pre Processing

Keeping in mind that our model needs to capture the variabilities in the images, background, color contrast etc, the model built is a multi-layered convolution neural network consisting of alternating convolutions and non-linearities. Additionally, the model has been trained with the following set of images to make it robust to variabilities in the test images:

a) Flipping the image: Few training images are randomly flipped from left to right, right to left, upside down in order to make the system robust to the orientation of the images.

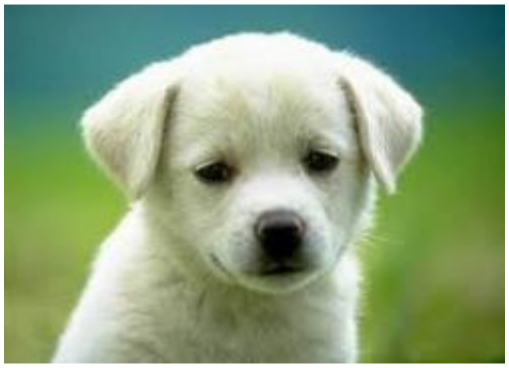


(Figure 2.2-1: Original Bagpack Image)

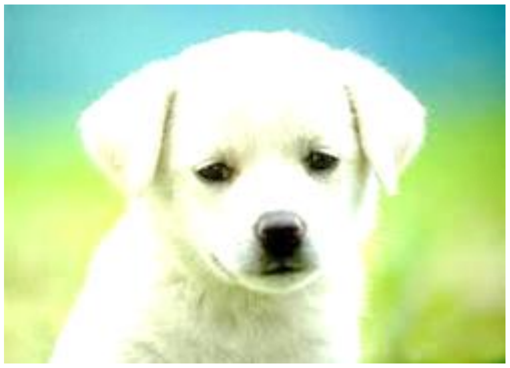


(Figure 2.2-2: Left rotated Bagpack Image)

b) Distort image brightness: Brightness of the image has been randomly distorted to make it robust to brightness factor.

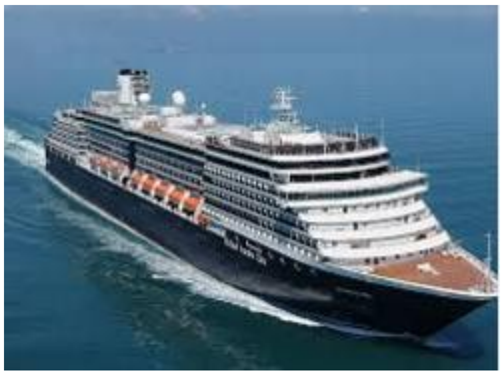


(Figure 2.2-3: Original Dog Image)

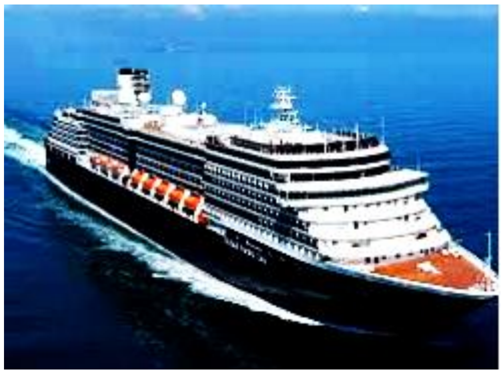


(Figure 2.2-4: Brighter Dog Image)

c) <u>Distort image contrast:</u> Contrast of the image has been randomly distorted to make it robust to brightness factor.

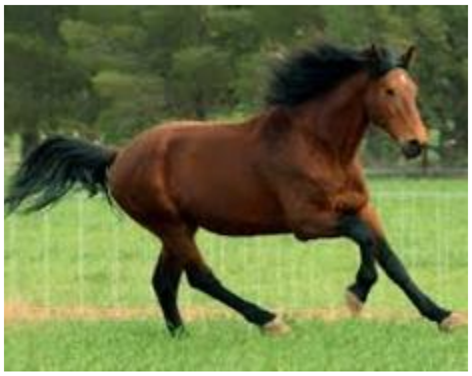


(Figure 2.2-5: Original Cruise Image)

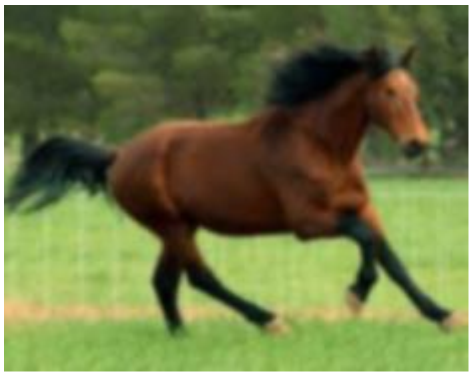


(Figure 2.2-6: Higher Contrast Cruise Image)

d) <u>Blur the image</u>: Few input images were randomly blurred to make it robust to artifacts in the images such as blurring artifacts.

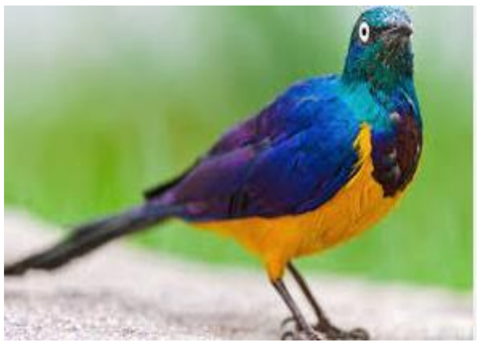


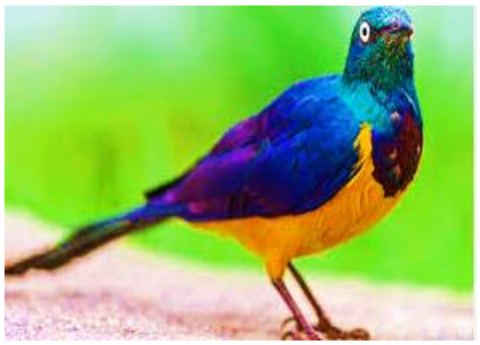
(Figure 2.2-7: Original Horse Image)



(Figure 2.2-8: Blurred Horse Image)

e) <u>Saturate the Chroma of the image</u>: Few input images were randomly saturated to make it robust to color artifacts present in the images.


(Figure 2.2-8: Original Bird Image)


(Figure 2.2-9: Saturated Bird Image)

Open source APIs were used to automatically generate the images with aforementioned variations and used to train the model.

2.3 Training the Convolution Neural Network

In the recent years, convolution neural networks have led to breakthrough results in variety of pattern recognition problems such as computer vision, speech recognition etc. Convolutional neural networks can be thought of a kind of neural network that uses many identical copies of same neurons. This allows the neural network to have lots of neurons and hence express computationally large models while keeping the number of actual parameters – the values describing how neurons behave – that need to be learned fairly small.

A CNN consists of a number of convolutional and subsampling layers optionally followed by fully connected layers. The input to a convolutional layer is an "m x m x r" image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has r=3. The convolutional layer will have k filters (or kernels) of size n x n x q where n is smaller than the dimension of the image and q can either be the same as the number of channels r or smaller and may vary for each kernel. The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size m−n+1. Each map is then subsampled typically with mean or max pooling over p x p contiguous regions where p ranges between 2 for small images (e.g. MNIST) and is usually not more than 5 for larger inputs. Either before or after the subsampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map.
A convolutional neural network consists of several layers [9]. These layers can be of three types:

a) Convolutional: Convolutional layers consist of a rectangular grid of neurons. It requires that the previous layer also be a rectangular grid of neurons. Each neuron takes inputs from a rectangular section of the previous layer; the weights for this rectangular section are the same for each neuron in the convolutional layer. Thus, the convolutional layer is just an image convolution of the previous layer, where the weights specify the convolution filter.

In addition, there may be several grids in each convolutional layer; each grid takes inputs from all the grids in the previous layer, using potentially different filters.

b) Max-Pooling: After each convolutional layer, there may be a pooling layer. The pooling layer takes small rectangular blocks from the convolutional layer and subsamples it to produce a single output from that block. There are several ways to do this pooling, such as taking the average or the maximum, or a learned linear combination of the neurons in the block. Our pooling layers will always be max-pooling layers; that is, they take the maximum of the block they are pooling.

c) Fully-Connected: Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. A fully connected layer takes all neurons in the previous layer (be it fully connected, pooling, or convolutional) and connects it to every single neuron it has. Fully connected layers are not spatially located anymore (you can visualize them as one-dimensional), so there can be no convolutional layers after a fully connected layer.

Suppose that we have some N×N square neuron layer which is followed by our convolutional layer. If we use an m × m filter ω, our convolutional layer output will be of size (N−m+1) × (N−m+1). In order to compute the pre-nonlinearity input to some unit x^{l}_{ij} in our layer, we need to sum up the contributions (weighted by the filter components) from the previous layer cells:

$$x_{ij}^{\ell} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab} y_{(i+a)(j+b)}^{\ell-1}.$$

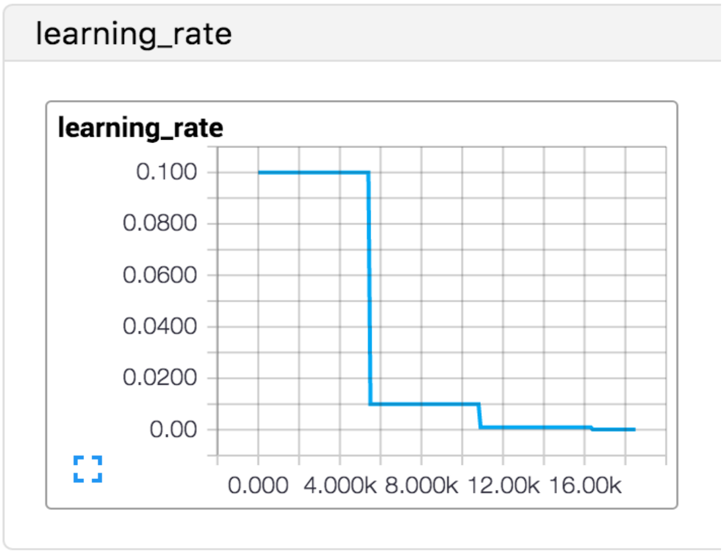
Then, the convolutional layer applies its nonlinearity:

$$y_{ij}^{\ell} = \sigma(x_{ij}^{\ell}).$$

Original images (as mentioned in section 2.1) and processed images (as mentioned in section 2.2) were used to train the system to make it robust to the artifacts and various variations possible in the images.
The commonly used method for training a network to perform N-way classification is multinomial logistic regression, aka. softmax regression. Softmax regression applies a softmax nonlinearity to the output of the network and calculates the cross-entropy between the normalized predictions and a 1-hot encoding of the label. We also apply the usual weight decay losses to all learned variables for regularization and hence address the overfitting problem. The objective function for the model is the sum of the cross entropy loss and all these weight decay terms.

Standard gradient descent algorithm has been used to train the model with adaptive learning rate that decays over time. Fig 2.3 describes the change in the learning rate for “Caltech 256” dataset with increasing iterations. As seen from the image the learning rate decays faster with iterations.

(Figure 2.3： Learning rate for “Caltect 256” using TensorFlow API)



2.4 Introduction to Tensorflow

TensorFlow is a Google open source for the implementation and deployment of large scale machine learning models. It was developed by Google brain development team. TensorFlow has the capability to run on different platforms such as distinct mobile devices and GPU platforms. The main approach adopted in this system can be described from the name Tensor that is an illustration of the N-dimensional array and the flow which is based on the data flow diagram form of calculation.

The first generation machine learning system that Google used is called DistBelief. It included the use of machine learning as the mode of recognition for the different objects within a photo based on the identification of the distinct features. DistBelief uses a positive reinforcement approach to collect that included the approval of the identified object and if wrong, there will be more features used in the process of identification. TensorFlow primarily uses a more advanced recognition system from DIstBelief. It uses more articulate parameters by the use of deep learning. This incorporates the artificial neural network and is more detailed since it uses many layers. The main approach used by TensorFlow is sorting through data layers known as nodes to determine the exact figure. There are several steps in determining the object's identity with the first step to look at the first layer for the broad aspects of the object. Then moves on to the other details of the object to give a more articulate definition.

Moreover, another useful feature in the Tensorflow is it provides some APIs in several languages for visualization of network activities during training, including input images, losses and distributions of activations. It can help us to understand, debug, and optimize our system.

2.5 TensorFlow Implementation

Model trained for our implementation is a multi-layer architecture consisting of alternating convolutions and non-linearities. These layers are followed by fully connected layers leading into a soft-max classifier. Implementation involves 3 stages

1. Model inputs
2. Model prediction
3. Model training

2.6.1 Model Inputs

The images are processed as follows:

- They are cropped to 24*24 pixels.
- They are approximately whitened to make the model insensitive to dynamic range.

For training, we additionally applied a series of random distortions to artificially increase the data set size:

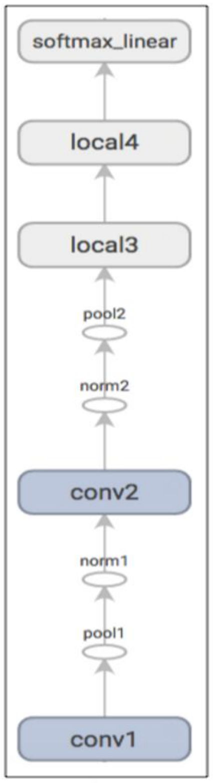
- Randomly flip the image from left to right.
- Randomly distort the image brightness.
- Randomly distort the image contrast

2.5.2 Model Prediction

The prediction part consists of the convolutional layer which is the core building block of a Convolutional Neural Network. The layer's parameters consist of a set of learnable filters, which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convoluted across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing 2-dimensional activation map of that filter. As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input.

Local Connectivity:

When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume. The extent of this connectivity



is a hyperparameter called the receptive field of the neuron. The connections are local in space, but always extend along the entire depth of the input volume.

Norm layer is useful to prevent neurons from saturating when inputs may have varying scale, and to aid generalization.

2.5.3 Model Training
We used multinomial logistic regression (softmax regression) to train our N-way classification network. For regularization, we also applied weight decay losses to all learned variables. We trained the model using standard gradient descent algorithm with a learning rate that exponentially decays over time.

2.5 Testing

Subset of the Caltech 256 dataset was used for testing. Additionally, processed versions of original images were also used for testing to test for the robustness of the system to the artifacts and the variations present in the images.

Open source APIs from TensorFlow have been used for the implementation.

CHAPTER 3
RESULTS

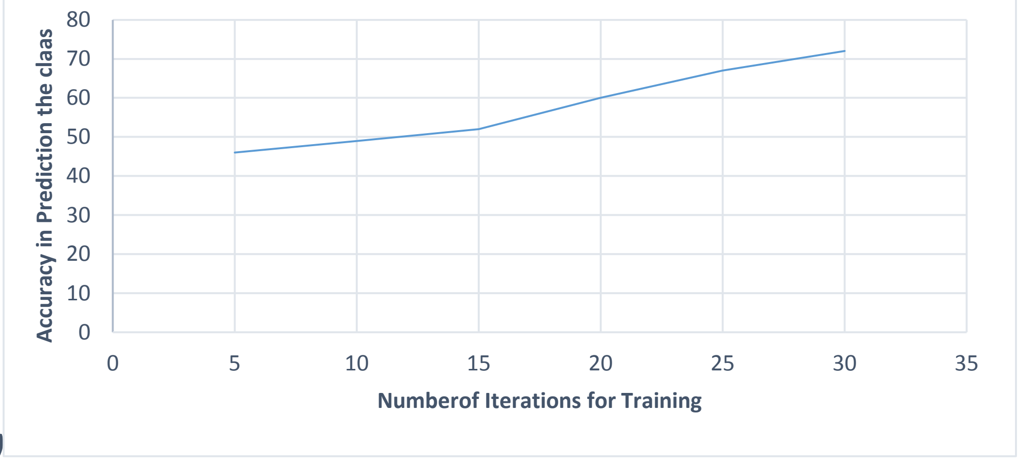
3.1 Prediction Accuracy

Convolution neural network was trained with subset of images of Caltech 256 dataset which consisted of 2000 images belonging to 10 different classes. Smaller dataset was intentionally chosen to reduce the time required for training and testing and hence allowing us to experiment with different variations.

Sufficient experiments were carried out by varying the parameters of the convolution neural network, different number of training iterations, varying the number of images in test dataset and train dataset and also with images of different resolutions.

Since training the system is an iterative process, the prediction accuracy was sensitive to the number of iterations in training. Following figure depicts the variation of accuracy with training iterations.

Figure 3.1-1: Accuracy vs Iterations for“Caltect 256”



()

Best prediction accuracy was achieved for 30K iterations and was around 72% accuracy with test images containing the following artifacts:

1) Images with artifacts
2) Rotated/flipped images
3) Blurred images
4) Images with similar background.
5) Images with similar classes
6) Images which did not belong to any of the classes that the system was trained for.

Testing the system with the subset of dataset and experimenting with different variations gave us the following results:

Total number of images = 1095

Train Images	**Test Images**	**Iterations**	**TensorFlow Batchsize**	**Accuracy**
895	200	20,000	32	65.0%
895	200	20,000	64	57.0%
895	200	20,000	128	49.5%
995	100	20,000	32	60.0%
995	100	20,000	64	66.0%
995	**100**	**20,000**	**128**	**72.0%**
995	100	20,000	256	49.2%
1045	50	20,000	64	60.1%

Figure 3.1 1 Table describing the prediction result accuracy for different experiments

3.2TensorFlow Results:

The following figures shows the variation of loss (i.e error) with different iterations. As expected, the loss decreases with increasing iterations. And once the difference between consecutive errors goes below a threshold, algorithm is said to be converged.

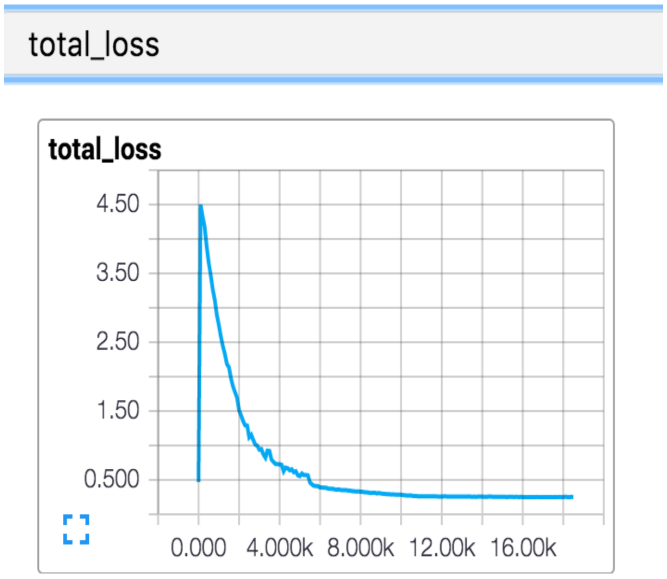


Figure 3.2 1 Moving average of loss vs iterations

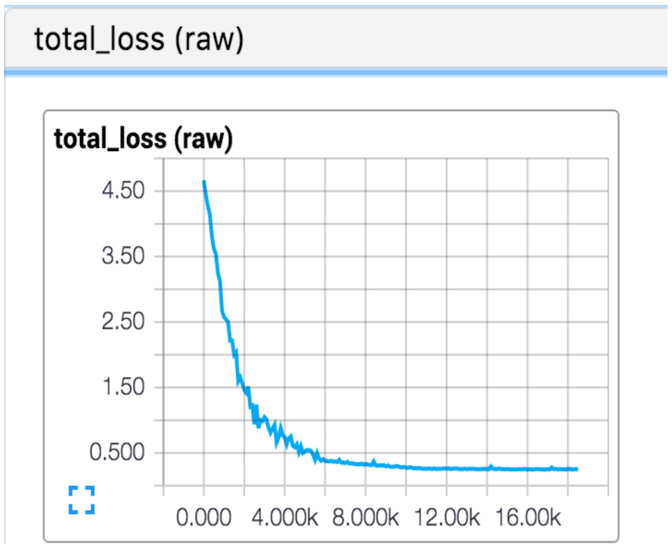
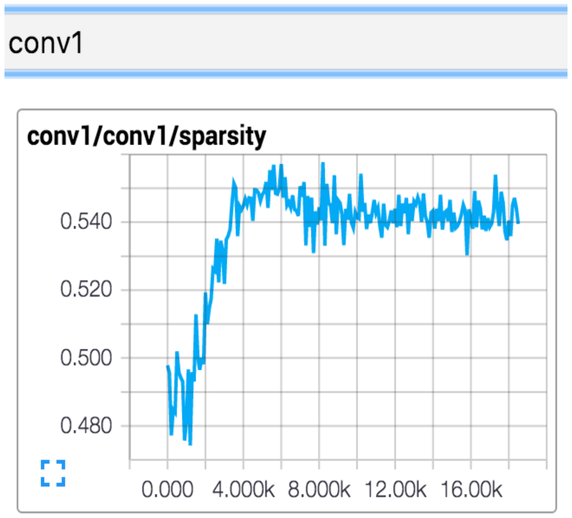
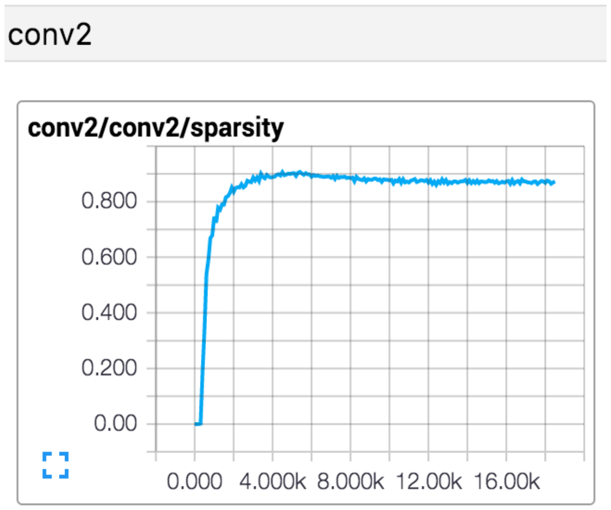
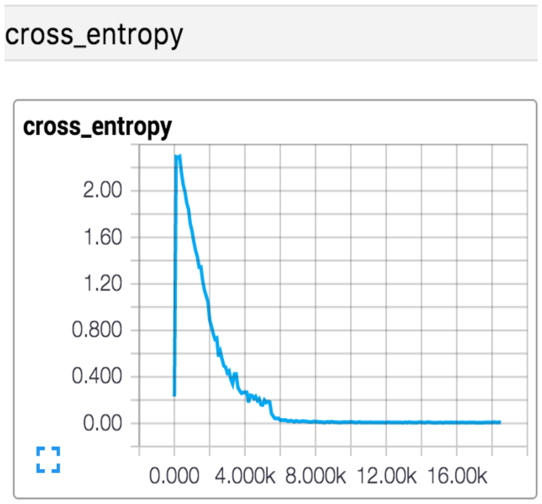


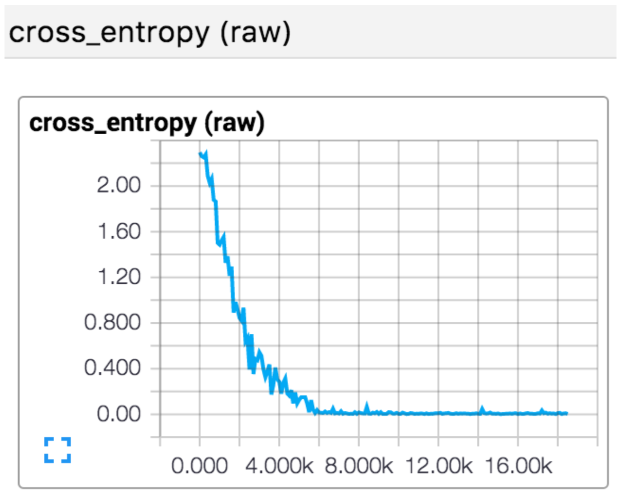
Figure 3.2 2 total loss for that iteration vs iterations

The following graphs show the variation of variation entropy, sparsity, weight loss etc. with training iterations.

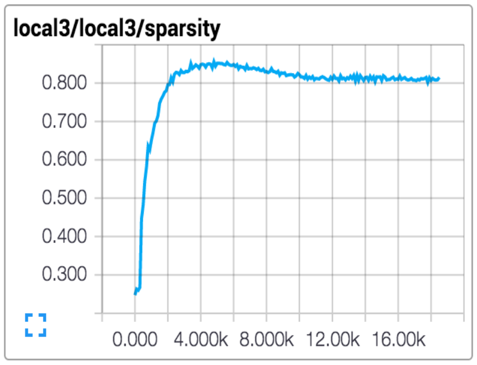


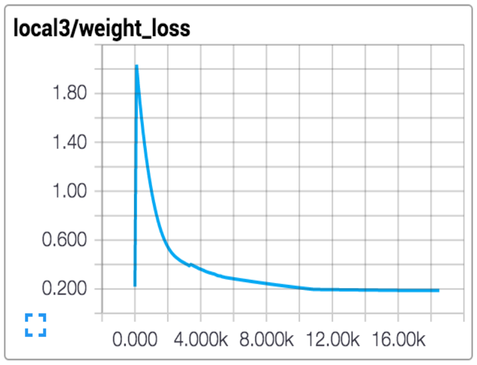


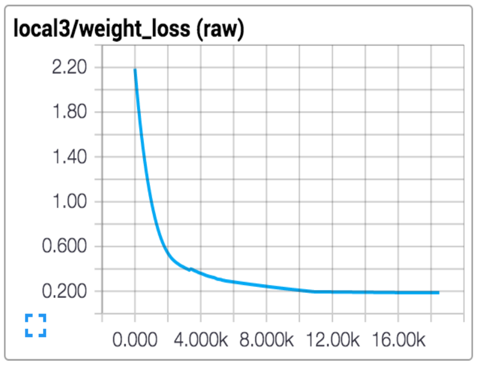




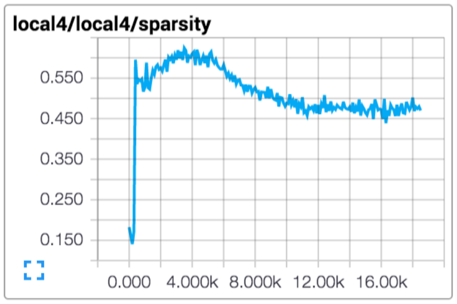
local3

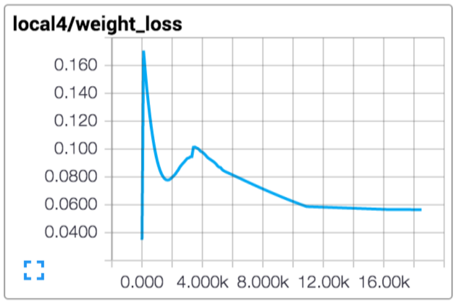


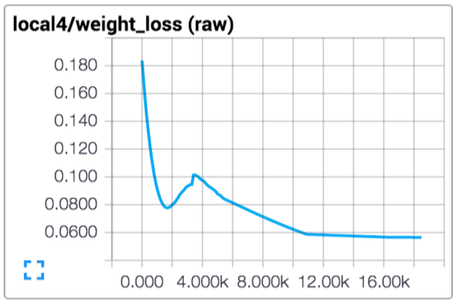




local4







CHAPTER 4

LIMITATIONS AND FUTURE SCOPE

4.1 Limitations

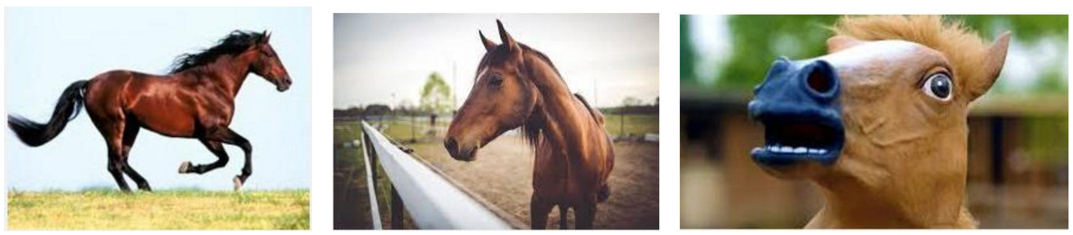
While the techniques presented in this paper have proved powerful with some impressive results, they are not able to cope with all the appearance variations (theoretically infinite) that any natural image present. In particular, the proposed classification algorithm only tackles objects viewed from one angle, and the range of scales that can be handled is somewhat limited, both by computational efficiency and by the representational power of the features. Also, sometimes the system is sensitive to the size of the object in the image. The current implementations of the work presented here are also too slow for real-time applications, hence accelerating the algorithms on GPU is necessary. Additionally, the system has been trained for relatively smaller dataset on our personal laptops which are comparatively slower, hence the system is yet to be tested for larger dataset (at least a million images).

4.2 Conclusion

From the results achieved, it is clear that convolution neural networks are suitable for modelling the myriad variations that are possible in visual recognition problem and hence can capture these variations with limited parameters to be learnt. Also, since the algorithm can be expressed in the form of matrix operations it is also suitable for implementation on GPUs and hence making it suitable for training the larger database in limited time. Thus, convolution neural networks, along with suitable additional algorithms like feature selection, feature generation etc can be used to solve the visual recognition problem.

4.3 Future Scope

Combining the features: The implementation can be extended to combine the related feature and represent it in the model. For example: In our current implementation, it has 3 different representation for images of horses in different angles as shown in the figure below. Such different models representing the variations of different images from the same class can be combined and represented as a single model. This will not only bring down the complexity of the system but also improves the prediction accuracy. Additionally, we can always try to improve the prediction accuracy by using more advanced algorithms and make it more robust to variations in the images.



(Figure 4.2: Advanced Image Processing)

Making the system robust to many more variations: Currently we have made the system robust to few variations like blurring artifact, orientation of the image, color saturation, contrast etc. It can still be improved by making it robust to many more artifacts.

References:

[1] Alex Krizhevsky., Ilya Sutskever, Geoffrey E. Hinton: “ImageNet Classification with Deep Convolutional Neural Networks.”

[2] Peter Gehler., Sebastian Nowozin: “On Feature Combination for Multiclass Object Classification.” 2009 IEEE 12th International Conference on Computer Vision (ICCV).

[3] Christian Szegedy., Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich: “Going deeper with convolutions.” 2014 GoogleNet

[4] Olga Russakovsky., Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei: “ ImageNet Large Scale Visual Recognition Challenge.” Advances in Neural Information Processing Systems 25 (NIPS 2012)

[5] Dumitru Erhan., Christian Szegedy, Alexander Toshev, Dragomir Anguelov: “Scalable Object Detection using Deep Neural Networks.” Google 2013

[6] Omar Javed., Mubarak Shah: “Tracking and Object Classification for Automated Surveillance.” Springer-Verlag Berlin Heidelberg 2002

[7] Huitao Luo: “Algorithms for Video Object Detection and Segmentation with Application to Content-Based Multimedia Systems.” 2000

system better.

[8] *LeCun, Yann.* "LeNet-5, convolutional neural networks". Retrieved 16 November 2013.

[9] “Convolution neural network: Andrew Gibiansky”