

FINE TUNING IMAGE INPUT SETS FOR DEEP  
NEURAL NETWORKS USING  
HALLUCINATIONS

By

KAVYA JALLA

Bachelor of Engineering in Information Technology

Osmania University

Hyderabad, Telangana, India

2012

Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December 2016

FINE TUNING IMAGE INPUT SETS FOR DEEP  
NEURAL NETWORKS USING  
HALLUCINATIONS

Thesis Approved:

Dr. Christopher Crick

---

Thesis Adviser

Dr. David Cline

---

Dr. Johnson Thomas

---

## ACKNOWLEDGEMENTS

I would first like to thank my advisor Dr. Christopher Crick of the Computer Science department at Oklahoma State University. The professor has guided me during the research with his thoughtful insights and his careful supervision. I would like to thank my committee members Dr. David Cline and Dr. Johnson P Thomas for their involvement in the research.

I would like to thank my parents Jalla Venkateswarlu, Jalla Shylaja and my sister Jalla Reshma for giving me with unflinching moral support to complete the research.

I would also like to thank my friends Praveen Kumar Daga, Divya Chalam Dadi for helping in preparation of data sets and Rakshit Dayal Allamraju for helping me with the equipment and experimental setup. I would like to thank my friends and roommates for all their support and encouragement.

Name: JALLA KAVYA

Date of Degree: DECEMBER, 2016

Title of Study: FINE TUNING IMAGE INPUT SETS FOR DEEP NEURAL NETWORKS USING HALLUCINATIONS

Major Field: COMPUTER SCIENCE

Abstract:

An artificial neural network is a system of software made up of neurons which work based on the neural structure of brain. While training any neural network, it is difficult to understand the features it has learnt at each layer. Deepdream is an algorithm which inverts the neural network using gradient ascent. It finds and enhances patterns of convolutional neural networks in input image based on the training images given to the network. Using this algorithm, we can find the pattern which each layer of a neural network recognizes from the image and enhance it. Often during training, the neural network learns the features which may or may not be the desired features. Having a knowledge of the features recognized in each layer will help us produce better results and increase the efficiency of the neural network.

Deepdream has been used for the purpose of entertainment for the hallucinogenic pattern it produced in the input images. To our knowledge this is the first implementation of usage of deepdream in learning the features from each layer of network and thus helping us fine tune the training image set so that the neural network gains better understanding of features from the tuned input set.

## TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	1
1.1 Training a neural network	2
1.2 Deep neural networks	2
1.3 Training deep architectures	3
1.4 Training deep neural networks on image data	3
MOTIVATION	5
II. REVIEW OF LITERATURE	6
III. ALGORITHM AND METHODOLOGY	10
3.1 Working of deepdream algorithm in simple terms	10
3.2 Technical overview of Deepdream Algorithm	11
3.3 Algorithm	12
Step 1: Choose a layer and filter of our interest	12
Step 2: Compute input image activations up to that layer	15
Step 3: Back propagate the activations of filter back to the input image	15
Step 4: Multiply gradients with the learning rate & add them to the input image	15
IV. EXPERIMENTAL SETUP	17
4.1 Installation and working with Caffe	17
4.2 Training the neural network & Preparation of Dataset	18
4.3 Dataset Preparation	18
4.4 Running Deep Dream	19
4.5 Classification accuracy calculation	20
V. ANALYSIS AND RESULTS	21
5.1 Observation 1	21
5.2 Observation 2	23
5.3 Observation 3	25
Chapter	Page

5.4 Observation 4	27
5.5 Observation 5	29
5.6 Trimming and Fine Tuning	31
5.7 Observation 7	31
5.8 Observation 8	34
5.9 Observation 9	36
5.10 Observation 10	37
5.11 Guidelines to prepare and fine tune an image data set for training a deep convolutional network	38
REFERENCES	40

## LIST OF TABLES

Table	Page
5.9 Number of training images vs Class values	37
5.10 Percentage of validation images vs Class values	38

## LIST OF FIGURES

Figure	Page
1.1 Artificial neuron with 3 inputs and one output	1
1.2 Deep neural network with 3 hidden layers	3
2.1. Convolution by Gabor filters	6
4.1. Successful installation of Caffe	18
5.1 Mug images training set	21
5.1.1 Output of mug training data on different layers of Alexnet	22
5.2 Rose images training set	23
5.2.1 Output of roses training data on pool 5 layer of Alexnet	24
5.3. Plain background bicycle training set	25
5.3.1. Output of bicycle training data on Inception_5b/output layer of GoogleNet	26
5.4. Bicycle training set with non-plain background	27
5.4.1 Output of bicycle training data on Inception_5b/output layer of GoogleNet	28
5.5. Bicycle training set from ImageNet	29
5.5.1 Output of bicycle training data on Inception_3b/output layer of GoogleNet	30
5.7. Bicycle training set with mixed and tuned input	32
5.7.1 Output of bicycle training data on Inception_5b/output layer of GoogleNet	32
5.7.2 Output of roses training data for on AlexNet Pool5 layer	33
5.8. Bicycle data flipped and rotated	34
5.8.1. Output of rotated bicycles on Pool 5 layer of AlexNet	35
5.9. Rose images training set	36
5.9.1 Output of roses training data on Pool 3 layer of Alexnet	36



## CHAPTER I

### INTRODUCTION

Human brain is made up of neurons which are the basic functioning units of information processing. Each neuron processes some information which helps us make sense from the things we see and perform. An artificial neural network is also similar to human brain, which comprises a set of neurons that when trained perform certain task. Every neuron in the neural network is associated with an activation function which may be as simple as a unit step function or as complicated as a hyperbolic tangent function.

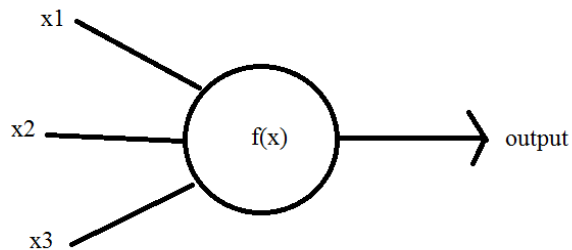


Fig 1.1 Artificial neuron with 3 inputs and one output

A neural network is activated by a set of input neurons which are activated by an input to the network. The input may be pixels of an image which activate the activation function. The result of this function is transmitted to the neurons in next layer along with the weight associated with the neuron. This goes on until it reaches the output layer. Every neural network has  $n$  neurons in its output layer where  $n$  represents the  $n$  number of classes of outputs the network produces.

### 1.1. Training a neural network

The initial weights associated with every neuron are 0 as the network is not trained to identify anything. The input records are given one at a time and the output is recorded. Then we calculate the error percentage based on the values of expected output and the recorded actual output. The error is propagated through back propagation[10] from the network. Hence each neuron in the network is associated with some weights and activation function that can classify the output.

### 1.2 Deep neural networks

A deep neural network is an artificial neural network with multiple hidden layers. Recent research is more oriented towards deep neural networks as they can represent large number of functions compactly. For example, consider a neural network that performs the function of Boolean OR gate. We require only one hidden layer and this can be performed using a simple neural network. Consider Boolean OR operation for 10000 inputs, then it requires 10000 neurons for the input layer to hold each value. But, using a deep neural network, we can add hidden layers in the network that can reduce the complexity of the problem to some polynomial  $k$ [19].

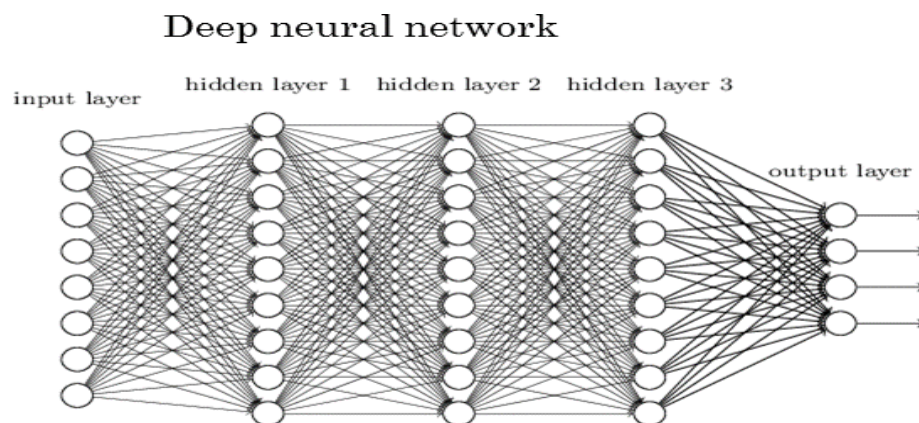


Fig 1.2 Deep neural network with 3 hidden layers

### 1.3 Training deep architectures

Training a deep architecture is difficult as the researchers train the network by initializing the neurons to random weights and use a labelled training set to classify them using a supervised learning objective. There are many problems in this techniques few of which are

#### I, Local optima:

In single hidden layer neural network the results are more converged during learning method. But in deep neural networks as the learning involves highly complex polynomial function that may not converge. Thus training with gradient descent or any other methods cannot be performed due to bad local optima.

#### II. Data availability:

Availability of labelled data is another problem. The above described technique uses labelled data to train the networks which is difficult for complex problems to get required number of examples.

### 1.4 Training deep neural networks on image data

In deep learning the features are learnt in hierarchies of the deep net architecture [1]. The higher levels are formed by the compositions of features [1] of the lower level. To gain a better understanding of how to train a network, we have to understand what features are learnt at each layer. As network layers are built hierarchically training each level extracts higher and deeper features of image where the final output layer takes the decision of which class the image belongs to [2]. For example, the first layers extracts the basic features like edges, corners of a face. The intermediate layers extract features like eyes, nose, ears etc. The final layer takes the decision of which face does that image belong to, like a cat, a dog.

## MOTIVATION

Detecting an object or a feature from a digital image is called image recognition. Recently deep convolutional neural networks are being widely used for object detection and classification. It is often observed that the neural network does not learn the features which we want it to learn. There are instances where researchers have found that a network trained to identify image of dumbbell identifies dumbbell along a hand holding it. Very less research has been performed on what a network is learning when compared to training a network on what we want it to learn. We can invert the network upside down and enhance the features learnt at each layer to find what the network has learnt using algorithm called Deepdream.

It is important to understand the features learnt by the network. Considering the complex algorithms involved, it is difficult to train the deep neural structure. To overcome this problem, we present the experiment to fine-tune the input image set based on the important features a deep convolutional neural architecture extracts from its training image set. The above can be achieved by extracting features at each layer using Deepdream algorithm and analyzing the results to fine tune a generalized image data set. This experiment is intended to provide guidelines for preparing an image dataset to train any convolutional neural network.

## CHAPTER II

### REVIEW OF LITERATURE

In the recent times, deep neural networks are widely used in the field of feature extraction and pattern recognition. In general for image data we use convolutional neural networks. As images can be of size  $200 \times 200 \times 3$  which results in 120000 weights for neurons, it is difficult to use normal neural networks [8]. In mathematical terms convolution is the amount of overlap of one function as it shifted over another function. Hence the filters that are used in the layers reduce the dimensionality of the image. The initial layers can identify the edges of an image.

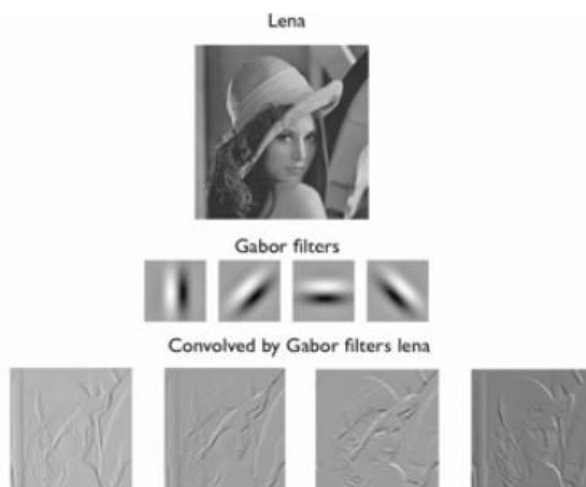


Fig 2.1. Convolution by Gabor filters

The above image shows Gabor filters which when convolved with the image of Lena produce images that identify the edges of the face [9]. After producing different activation maps for different filters, we can stack them to generate an output volume that can help us identify the class of the image at higher layers of the network.

A convolutional neural network has each layer of data in three dimensional array of size where  $w$  and  $h$  are spatial dimensions and  $d$  is the channel dimension generally 3 representing BGR (blue green red) channels [12]. Segmentation of an image is the process of partitioning digital image into multiple segments which helps representing the image in a more meaningful and easier manner. Extending fully convolutional neural networks of classification to segmentation and adding multiple layers improve learning and analyzing of image [12].

The deep networks are achieving almost human level accuracy in classifying images and data. There are few cases where the deep neural networks are not always successful in classifying few images which can be conflicting [3]. There have been only a handful of papers that address the understanding of deep neural networks and how the network understands the images. But recent studies reveal that networks always do not learn what we want them to learn. It is often easy to fool a network by giving a conflicting image which the network identifies to be something else according to its training but is actually a completely different image. Example consider a network that identifies digits from 0-9. Images that are classified to recognize 1 tend to have vertical lines in the images. So when an image with vertical lines is given to the network, the network results with 99.99% that the given is digit 1 [21]. This concludes that deep neural networks can be easily fooled with high confidence predictions for unrecognizable images [4].

Inverting image representations using up convolutional neural networks also throws light on analyzing feature representations [11]. An interesting approach is by training the network to predict expected pre-image of the given feature map of an image. The expected feature map is the weighted average of all natural images which could have produced the feature vector given. Up convolutional network refers to up-sampling by a factor along with convolution of the network. Applying the algorithm to different layers of the Alexnet network has helped in understanding the network closer. The higher layers of network preserve the color and positions of the object including the final layer [11]. Any information which when replaced by zero activation does not affect the image reconstruction can be an unimportant feature of the image. By applying up convolutional neural networks we can find that most of input image information is contained in form of non-zero activation functions in higher layers of the network.

Another interesting study on understanding deep image representations by inverting them has been done [7]. By optimizing the objective function using gradient descent, we can optimize the image reconstruction in deep and shallow neural networks. A generalized framework to invert representations such as HOG and SIFT can be found by finding an image whose representation matches the best with the given one[7]. HOG (Histogram of Oriented gradients) follows a sliding window mode in the process of object detection. SIFT (Scale invariant feature transform) follows matching local regions to find similarities between its trained images and image that requires reconstruction, alignment. A differentiable image representation can be inverted with gradient descent. For a given feature vector we approach an image that reduces loss function by calculating the Euclidean distance between the feature vectors. Thus reconstructing the image with more invariant and abstract notion of image content formed in the neural network. When optimization function is applied to the network, information in each layer of the network can be

visualized. Applying gradient descent to objective function helped in understanding features extracted from each layer and this has been first addressed in [7].

Transforming input images into different artistic styles has been achieved in [14]. They have separated and recombined the style and content of image using a network trained on artistic images. Inversion of deep neural networks by applying gradient descent to the loss functions for content reconstruction and style reconstruction. This helps in gaining understanding of human interpretation of artistry. Inversion of neural network can also be used to transform input images [15]. For a given image and some class label, research on how the network can transform the given image to be categorized as image of given class label is performed [15]. By applying inversion at the final layers that calculate the class score of the images we can extract features that can be updated in the input image to change its class label.

In the recent years, little amount of research has been performed on how the information at each layer of a neural network can be used. Few of the applications include image reconstruction, inverting an image. Using inversion of neural networks with image representations we can get some idea about what a network is learning at each layer. Deep dream is an algorithm that helps us identify features learnt from its training and can reproduce them in the output image. This algorithm has been used for entertainment purposes in the recent past by creating hallucinogenic effects on images in the internet. But information from this algorithm helps us fine tune the input set for training as this can eliminate the probability of network learning unwanted features. Our paper proposes an analysis of this algorithm on different data sets and provide guidelines to fine tune input data set for a neural network.



## CHAPTER III

### ALGORITHM AND METHODOLOGY

As discussed in the previous section deep neural networks have been used from many decades but they are being widely used recently due to their improved accuracy and results when compared to other networks. Large part of the research is dedicated on how to train the neural networks and what can be networks used for. Very little research is done on learning what a network has learnt and how this can be used to fine tune input data.

Deepdream is an algorithm that can be used to learn what features a network has learnt during its training.

#### 3.1 Working of deepdream algorithm in simple terms

Consider an image recognition software that is trained to recognize what an image is. The software compares the given input to the images with which it has been trained or known, if a match is found it recognizes the image. For example, if a neural network has been trained to identify cats and the given input is an image of cat, then the network recognizes it as a cat. If we present an image of an elephant to the same software, it informs that there is no cat in the image [5].

Consider a software that has been trained to identify cats in an image and we have given an image of an elephant and informed it that there is a cat in the image. The software tries to extract the feature nearest to the cat features from the elephant's image and returns saying it is a cat. Taking this concept, consider the image changing the input image to be more cat like. Deep dream is a similar concept where the network is trained on a set of input images and when an image is given, the algorithm extracts features at each layer of image that are similar to its training set[20]. The output layer has a feedback link which sends the image to the network again finally making the image a representation of what the network sees in the image rather than what we want it to see in the image.

### 3.2 Technical overview of Deepdream Algorithm

Deepdream is an algorithm that is written in caffe framework. Initially the input image is converted into an array. A *net\_fn* variable carries the neural network information on which the deep dream algorithm is being implemented. Another variable *param\_fn* carries the training information of the neural network. Network stores the information in data blobs. A basic gradient ascent step is performed on the input data blobs.

In deepdream we implement gradient ascent function through different scales. Initially we prepare base images for the octaves (arrays to store ascent). Then we allocate space for network produced image details. We continue the process and upscale details from previous octave. After this, we resize the image according to network's input image size. Then de-process the image using utility functions to see what the network has learnt.

In the deepdream function we can specify which layers activation function has to be maximized and hence see what each layer of the network is learning from the training.

### 3.3 Algorithm:

Step 1: Choose a layer and filter of our interest

Step 2: Compute input image activations up to that layer

Step 3: Back propagate the activations of filter back to the input image

Step 4: Multiply the gradients with the learning rate and add them to the input image

Step 5: Repeat steps 2 to 4 until satisfied with the result

Following are the detailed steps of the algorithm:

Step 1: Choose a layer and filter of our interest

To work with deep dream we have to use a convolutional neural network. For the experiment I have used AlexNet and GoogleNet which are classification convolutional neural networks. AlexNet is a network comprising of 650000 neurons. They are arranged in five convolutional layers, followed by some max pooling layers and three fully connected layers followed by a 1000 way soft max layer. We can select any of the convolutional layers, max pool layers. GoogleNet [16] is a 22 layer deep neural network. It has inception layers which are series of combinations of different convolution and filter concatenation layers. The network also includes max pooling layers which are then combined to get output of a single layer adding advantage of getting different features at same layer.

For image data we use convolutional neural networks. In mathematical terms convolution is the amount of overlap of one function as it shifted over another function. As images can be of size  $256 \times 256 \times 3$  which results in 196,608 weights for neurons, it is costly to use such number of neurons in each layer as it also increases the processing time, Convolutional networks process portions of the input image. They are small collections of neurons whose

outputs are tiled so that their input regions overlap, to obtain a better representation of the original image. Following is the operation performed in convolutional layer

Input Image	Filter	Convolved output
+1 +1 +1 0	+1 +1 +1	
+1 +1 +1 0	+1 -8 +1	0, __
+1 +1 +1 0	+1 +1 +1	

Input Image	Filter	Convolved output
+1 +1 +1 0	+1 +1 +1	
+1 +1 +1 0	+1 -8 +1	0, -5
+1 +1 +1 0	+1 +1 +1	

The dot product of input image and the filter is calculated where the size of matrices multiplied is as per size of filter which results in the convoluted output matrix. These convolutional layers are followed by pooling layers to simplify their output. Pooling layers subsample the image and hence reduce the number of parameters in the network. Here a 2x2 filter with stride of 2 is slides over the entire image. Stride 2 means the filter would move 2 pixels after performing operation. As we use 2x2 filter with stride 2, the overlaying does not happen. Following shows operation of maxpool on input image of size 4x4 with filter of size 2x2. The maximum of 2x2 cells of input image is considered as the output of that region of the image.

Input Image	Maxpool output	Input Image	Maxpool output
-------------	----------------	-------------	----------------

+3 +2 +5 +1	+9 ___	+3 +2 +5 +1	+9 +5
+9 +7 +2 +4	___ ___	+9 +7 +2 +4	___ ___
+8 +3 +1 +2		+8 +3 +1 +2	
+1 +5 +6 +2		+1 +5 +6 +2	
Input Image	Maxpool output	Input Image	Maxpool output
+3 +2 +5 +1	+9 +5	+3 +2 +5 +1	+9 +5
+9 +7 +2 +4	+8 ___	+9 +7 +2 +4	+8 +6
+8 +3 +1 +2		+8 +3 +1 +2	
+1 +5 +6 +2		+1 +5 +6 +2	

The fully connected layers are only present at the end of the network as it is costly to have a network full of these layers. They require many parameters when compared to convolutional and pooling layers. As the name suggests in a fully connected layer every neuron is connected to every neuron of the previous layer. The advantage of adding this layer at the end of network is that it we can easily learn the nonlinear combinations of outputs of convolutional layers.

Step 2: Compute input image activations up to that layer

Every neuron in the network is associated with weights and activation function. The pixel values from the input image are sent to neurons layer by layer where their activations are calculated. In recent networks the activation function in all layers is ReLU (Rectified Linear Unit) function which transforms the input non-linearly and is defined as

$$f(x) = \max(0, x)$$

Elementwise activations are applied to the input image on all neurons up to the layer selected.

Step 3: Back propagate the activations of filter back to the input image

Backpropagation aims to update the weights in the network so that the actual output is closer to the expected output. Backpropagation is performed during training of the network. The loss function defines the filter activations of the layer. In general the loss function is defined as

$$E = ((\text{target}) - (\text{actual}))^2.$$

Adding all the weights of neurons and applying activation function gives the output the filter activation at that layer. This is performed in the forward pass. In deepdream the gradient ascent of the loss function are calculated. By doing this we maximize the loss function which enables us to get the difference between input image network learned features.

Step 4: Multiply the gradients with the learning rate and add them to the input image

Learning rate is a constant value that is multiplied to the gradients of the loss function in order to manage the weights in the network. We calculated the gradients of the loss function and add them to the input image. Now we allocate another empty image to store the network details. We upscale the features in details matrix to improve the network learned features. Adding these to the input image adds features that the network has identified at the layer. Deepdream has a feedback link that takes the output of this step and sends the image to the network again, i.e. the gradient ascent function is called which maximizes the loss function. Repeating the above steps multiple times produces hallucinogenic effects added into the input image. These effects are nothing but the features that the network has learnt from its training at that layer.

This experiment intends to extract features in the network using the deep dream algorithm and use this information to optimize the input set. By running the experiment multiple times and training the network with different datasets we can gain a better understanding of the network and thus optimize the input image set so that it improves the feature extraction as well as image recognition of the network.

## CHAPTER IV

### EXPERIMENTAL SETUP

For the experiment we have used Caffe deep learning framework. Following are the steps involved in experimental setup and execution

#### 4.1 Installation and working with Caffe

Caffe is a deep learning framework made with speed, expression and modularity which is developed by Berkeley Vision Learning Center (BVLC). Deep dream is developed in caffe.

Installation of caffe involves 3 major steps [6].

- I. Installation of prerequisites
- II. Compilation
- III. Hardware requirements

Prerequisites:

Caffe has several dependencies. The following are few of which that have to be installed for caffe to run successfully. Few of the dependencies are CUDA, BLAS, boost, protobuf, Glog, gflag, hdf5, python.

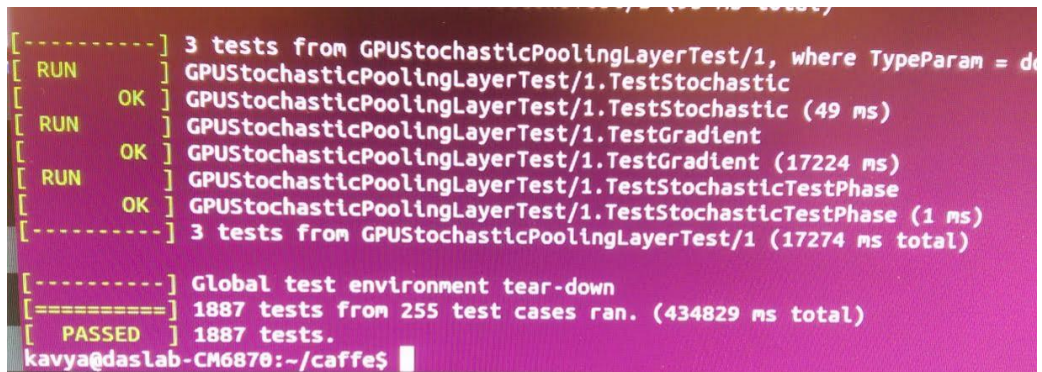
Compilation:



Caffe can be compiled using the Makefile.config provided in the package downloaded from github. A series of commands following the “make all”, “make test”, “make runtest” are issued.

Hardware:

The computer should be compatible with CUDA. The computer should possess NVIDIA graphics card.

A terminal window with a dark background and light-colored text. The text shows the execution of tests for the GPUStochasticPoolingLayerTest. It lists three tests: TestStochastic (49 ms), TestGradient (17224 ms), and TestStochasticTestPhase (1 ms). The total time for these three tests is 17274 ms. Below this, it shows a global test environment tear-down and a summary: 1887 tests from 255 test cases ran in 434829 ms total, all of which passed.

```
[-----] 3 tests from GPUStochasticPoolingLayerTest/1, where TypeParam = de
[ RUN    ] GPUStochasticPoolingLayerTest/1.TestStochastic
[ OK     ] GPUStochasticPoolingLayerTest/1.TestStochastic (49 ms)
[ RUN    ] GPUStochasticPoolingLayerTest/1.TestGradient
[ OK     ] GPUStochasticPoolingLayerTest/1.TestGradient (17224 ms)
[ RUN    ] GPUStochasticPoolingLayerTest/1.TestStochasticTestPhase
[ OK     ] GPUStochasticPoolingLayerTest/1.TestStochasticTestPhase (1 ms)
[-----] 3 tests from GPUStochasticPoolingLayerTest/1 (17274 ms total)

[-----] Global test environment tear-down
[=====] 1887 tests from 255 test cases ran. (434829 ms total)
[ PASSED ] 1887 tests.
kavya@daslab-CM6870:~/caffe$
```

Fig 4.1. Successful installation of Caffe

#### 4.2 Training the neural network & Preparation of Dataset

Deepdream algorithm is implemented in Caffe framework using ipython notebook. The algorithm can be used on convolutional neural networks. For our research I have used AlexNet[13] and GoogleNet[16].

#### 4.3 Dataset Preparation

I have downloaded the images from google based on the category on which I want to train the network, e.g. cars, trains etc. I have prepared two disjoint folders with train and validation data images. I have created datasets from scratch as the experiment intends to provide the analysis of

network with respect to variations in the training data. Then we prepare file listing the labels along with the categories to which they belong to. We resize all the images to 256x256 as per the network architecture.

Now we create level dB (stores keys and values of the input data on disk) for training and validation data. Then we compute image mean for the data as the model requires us to subtract image mean from each image. The mean of the whole image dataset is calculated and subtracted from each image as it improves the accuracy in the framework.

Every model has a train\_val.prototxt file which specifies Train or Test phase. This creates a training network and a testing network in the same file. Input layer receives its data from train\_leveladb created and validation receives its data from val\_leveladb files.

We provide the parameters of our training like initial learning rate, policy of increase in learning rate, gamma (a constant value to multiply with learning rate through the training process), total number of iterations for validation and training, snapshot status in solver.prototxt file.

#### 4.4 Running Deep Dream

Deepdream is implemented using Caffe framework. The algorithm is written in python which can be executed using ipython an interactive python shell.

After preparation of dataset, we train our convolutional neural network. After successful completion of training of the network, we execute the deepdream code for our network and filters by selecting the layer from which we want to extract the features.

After each successful execution we can analyze the results based on the training set and the layers selected to draw conclusions and gain understanding of network in relation to the training image

data set.

#### 4.5 Classification accuracy calculation

I have used DeepDetect which is an application program interface of deep learning. I have trained my neural networks in caffe and used DeepDetect to set up an image classifier for the trained caffemodel [17]. After downloading and installing DeepDetect and its prerequisites from the git [18], I have started the DeepDetect server and created services for the caffemodel files of my trainings to calculate the class scores of test images given to each trained model.

## CHAPTER V

### ANALYSIS AND RESULTS

After training the network for the dataset prepared, we run the deepdream by selecting the layers from which we want the features to be extracted. Selection of layers and filters produces different patterns that the network finds similar to its trained images from the input image. By analyzing the output we can use this experiment as baseline detector for different kinds of data and how they have been learned by the network.

#### 5.1 Observation 1:

Training set:

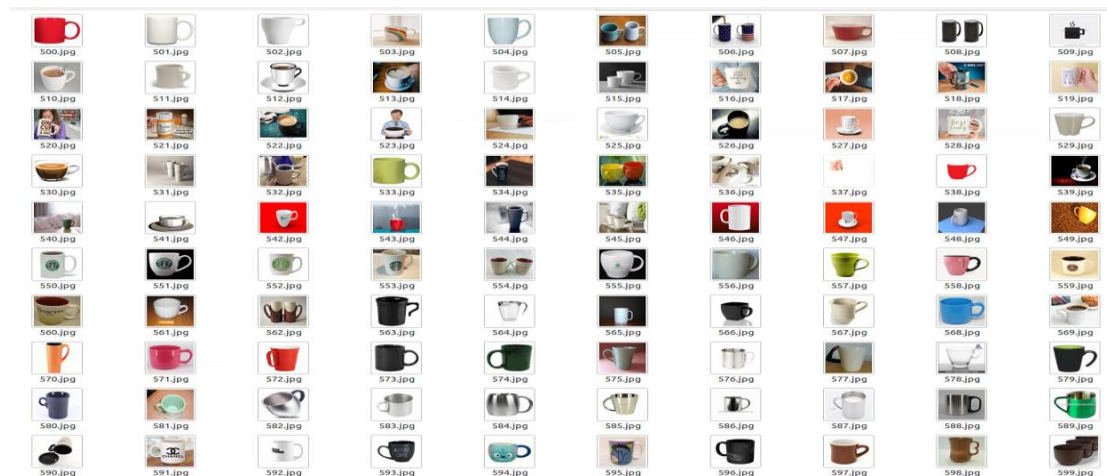


Fig 5.1 Mug images training set

I have prepared a dataset of images of mugs and trained Alexnet for the data.

By training Alexnet for 100000 iterations and selecting different layers of Alexnet I got the following results.

Results Observed:

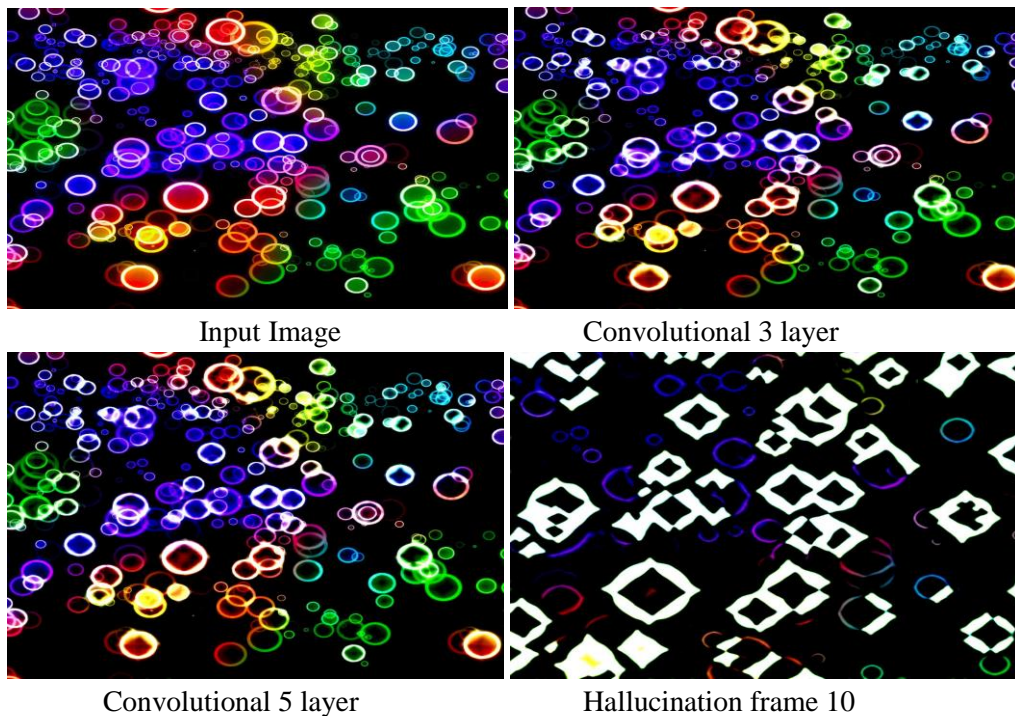


Fig 5.1.1 Output of mug training data on different layers of Alexnet

Analysis:

We can observe that the network has identified white edges on the image at different layers. As the training data has majority of data as white mugs, the network has identified white edges in the input image from different layers.

Conclusion:

From here we can observe that color is also an important feature in image recognition as color of an object is preserved in almost all the layers of the network.

Uses:

Most of the image recognition algorithms work on grayscale images and using this analysis we can say that color is also an important factor in training the network as well as image classification.

## 5.2 Observation 2:

Training set:

I have prepared dataset of images of roses. By training Alexnet for 100000 iterations and selecting different layers of Alexnet I got the following results.

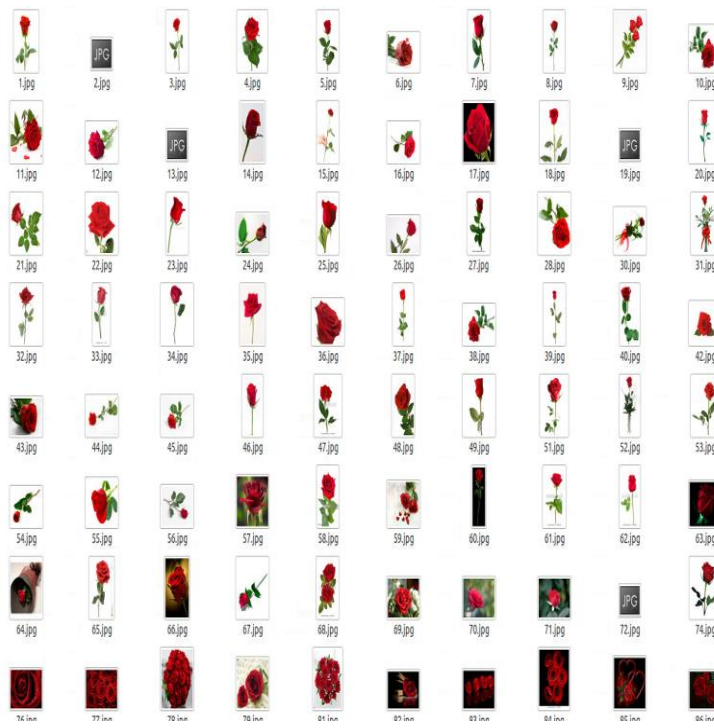


Fig 5.2 Rose images training set

Results Observed:

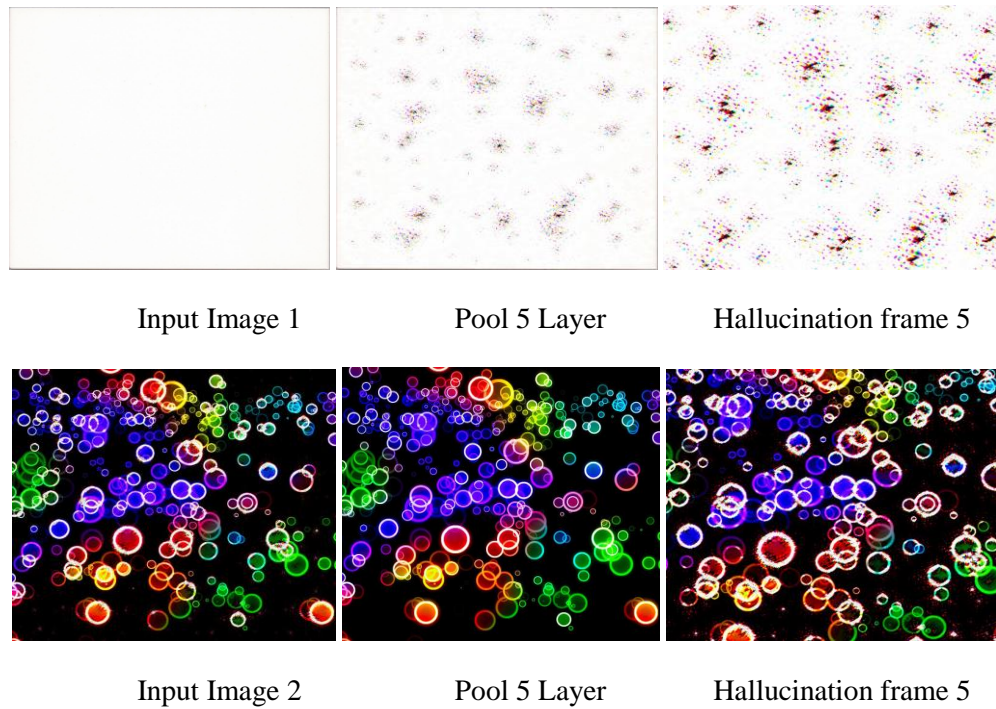


Fig 5.2.1 Output of roses training data on pool 5 layer of Alexnet

Analysis:

We can observe that the network has identified round structures in the input image at the layers of Alexnet. In input image 2 it has identified the red bubbles as roses and started to hallucinate on rose data. Roses training set had many images of roses oriented at different angles and the pictures of roses are present in different ways.

Conclusion:

From here we can observe that the network has understood the structure of rose in a better manner as the training data had roses in many ways.

Uses:

The class scores of test rose images for the caffemodel are 0.886, 0.799. This indicate a very high probability of classification accuracy. Thus, helping the network understand the structure of object used for training improves the network's understanding of the data. This in turn helps in

producing better results in image classification and image recognition.

### 5.3 Observation 3:

Training set:

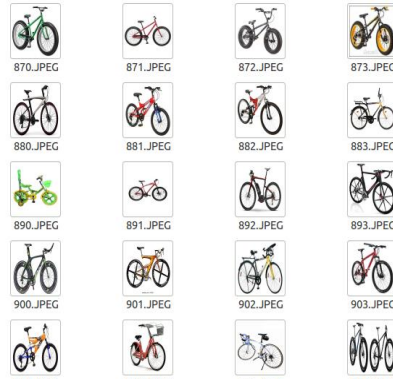
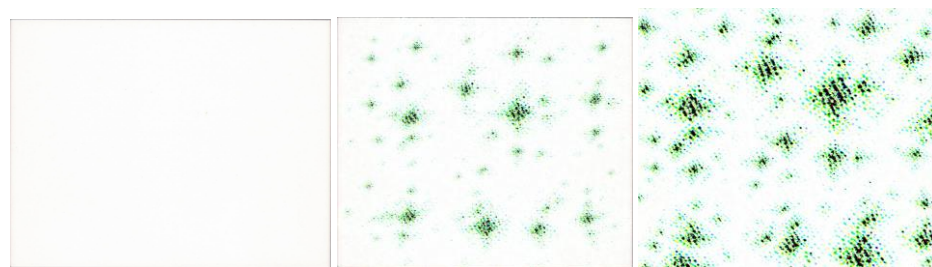


Fig 5.3. Plain background bicycle training set

I have prepared dataset of bicycles which have plain background. The data set comprises of images of different bicycles. I have trained the bicycle data on GoogleNet for 100000 iterations and the following results were observed.

Results Observed:



Input Image1

Inception 5b output

Hallucination frame 9



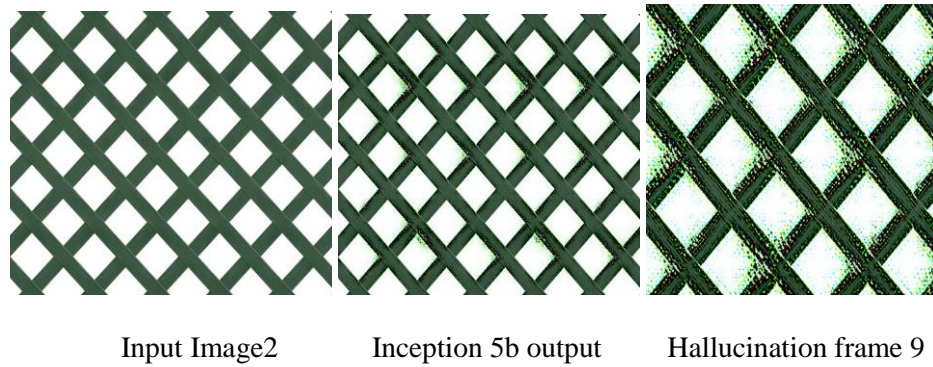


Fig 5.3.1. Output of bicycle training data on Inception\_5b/output layer of GoogleNet

Analysis:

From the hallucinations of input image 1 we can observe that the network has not learnt the structure of bicycle well. From hallucinations of input image 2 we can observe that the network has learnt the geometric corners of bicycle instead of entire bicycle. So the network has hallucinated at the corners of the lattice.

Conclusion:

If the majority of data then the network need not learn what we want it to learn .As there is no image with background data it may not be able to compare the clean and dirty images. So it will not be able to learn the entire features of the object correctly.

Uses:

The class scores of test bicycle images without background data for the caffemodel are 0.796, 0.724. This indicate a good probability of classification accuracy for images. The class scores of test bicycle images background data for the caffemodel are 0.125, 0.395 and 0 or the correct class has not been predicted. This indicate an average probability of classification accuracy for noise images. But real time data has lot of background data and the classification network should perform well for images with background data also. Thus, plain images are not very useful

training set for classification algorithms in real time data.

#### 5.4 Observation 4:

Training set:

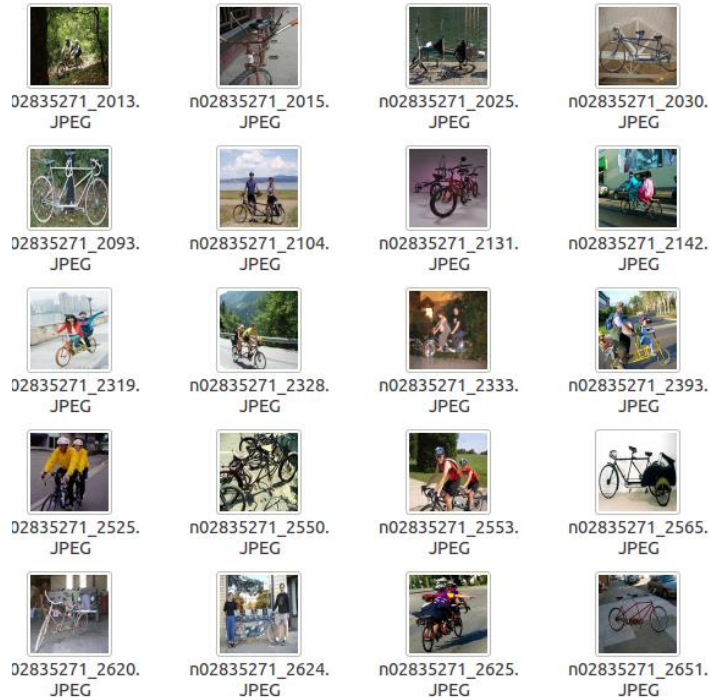


Fig 5.4. Bicycle training set with non-plain background

I have prepared dataset of bicycles which have lot of background noise. The data set comprises of images of different bicycles with human, animal and other background noise images. I have trained the bicycle data on GoogleNet for 100000 iterations and the following results were observed.

Results Observed:

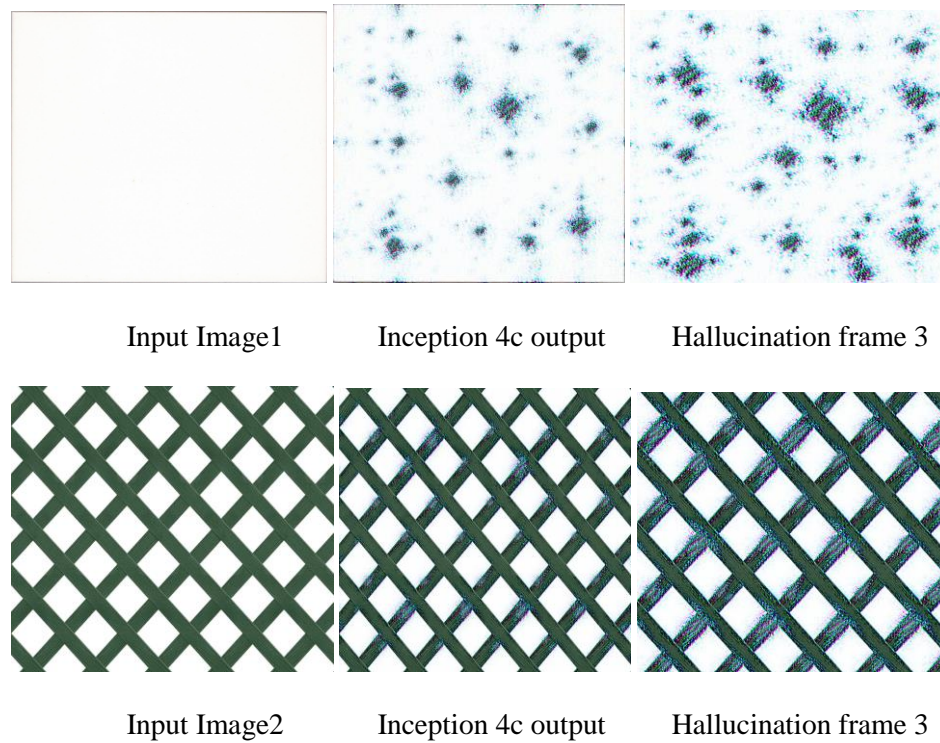


Fig 5.4.1 Output of bicycle training data on Inception\_5b/output layer of GoogleNet

Analysis:

From hallucinations of input images 1 and 2 it can be observed that the network did not learn the structure of bicycle well. It may be caused due to the problem of overfitting which refers to the phenomena of recognizing random error or noise instead of actual data. To overcome this problem we have to change the training set of the network so that the noise is eliminated in the hallucinations.

Conclusion:

Having a lot of background data in the training images may cause overfitting which leads to random object recognition instead of the desired results.

Uses:

The class scores of test bicycle images without background data for the caffemodel are 0, 0.00254

and 0.0321. This indicate a poor or almost no probability of classification accuracy for images. The class scores of test bicycle images with background data for the caffemodel are 0.253, 0.168 and 0 or the correct class has not been predicted. This indicate a low probability of classification accuracy for noisy images. But real time data has lot of noise and the classification network should perform well for noisy images also. Thus, noiseless images are not very useful training set for classification algorithms in real time data.

### 5.5 Observation 5:

#### Training set:

I have downloaded dataset of bicycles which has been provided in the ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2011. The data set comprises of 1300 images of different bicycles including plain and few images with background data. I have trained the bicycle data on GoogleNet for 100000 iterations and the following results were observed.



Fig 5.5 Bicycle training set from ImageNet

Results Observed:

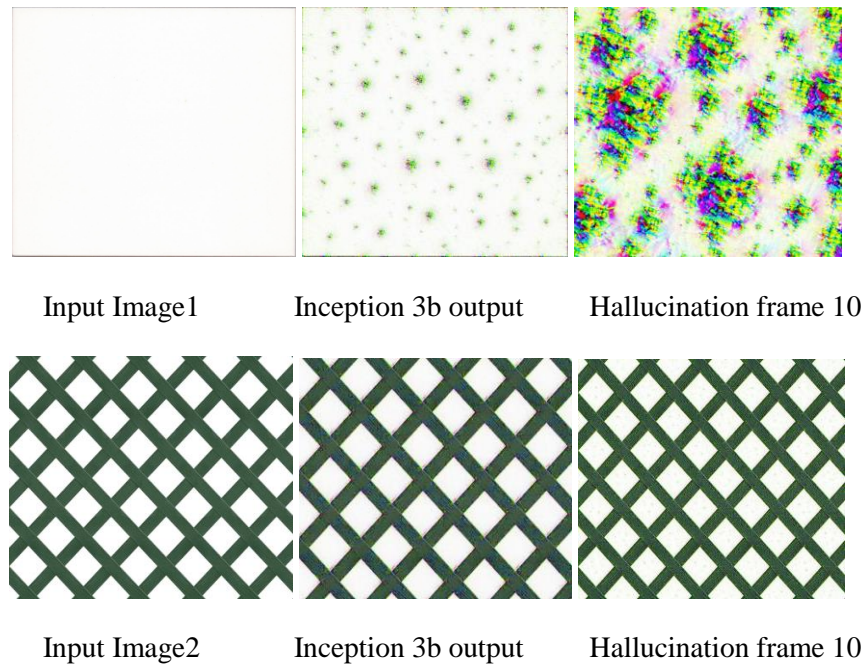


Fig 5.5.1 Output of bicycle training data on Inception\_3b/output layer of GoogleNet

**Analysis:**

From hallucinations of input images 1 and 2 it can be observed that the network did not learn the structure of bicycle well. As the ImageNet data has many images of humans riding bicycle in parks the network has learnt park as one of the features of the input image and hence in hallucination of image 1 it has produced colored pattern with a green pattern background. In hallucinations of input image 2 it can be observed that the network identified the green parts of the image as the network is trained with lot of images in parks.

**Conclusion:**

Having a lot of similar background data in the training images may result in network understanding the noise as a part of the training data.

Uses:

The class scores of test bicycle images without background data for the caffemodel are 0.497, 0.524. This indicate an average probability of classification accuracy for images. The class scores of test bicycle images noise for the caffemodel are 0.561, 0.425. This also indicate an average probability of classification accuracy for images with background data. The class scores of test bicycle images with parks for the caffemodel are 0.774, 0.621. This indicate good probability of classification accuracy for noisy images. But real time data has lot of different background data and the classification network should perform well for all images.

#### 5.6. Trimming and Fine Tuning:

From the observations so far, we can trim our data sets to produce better results by following steps:

- If the network leads to abstract hallucinations which are not related to the training set then it may be caused due to very high noise in the training set. Sometimes this may be caused due to very clean images in training set.
- Trimming the training set by decreasing the number of images with background data or adding more clean images to training set improves network efficiency.
- If we observe persistent colors or patterns in the hallucinations which are not a part of the desired object then this is caused due to large amount of similar background data in training set.
- Adding images with new backgrounds and trimming away data of similar backgrounds will improve the network performance.

#### 5.7 Observation 7:

Training set:

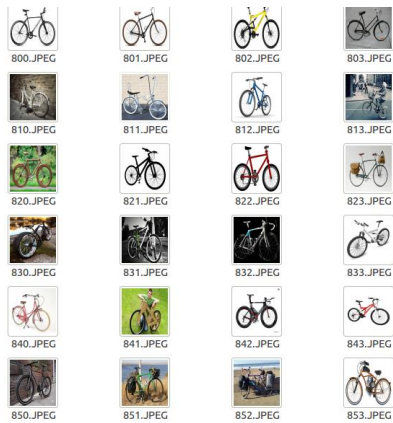


Fig 5.7. Bicycle training set with mixed and tuned input

I have modified the existing ImageNet dataset by trimming the similar background images and adding new clean images. I have trained this data on GoogleNet for 100000 iterations and the following results are observed.

Results observed:

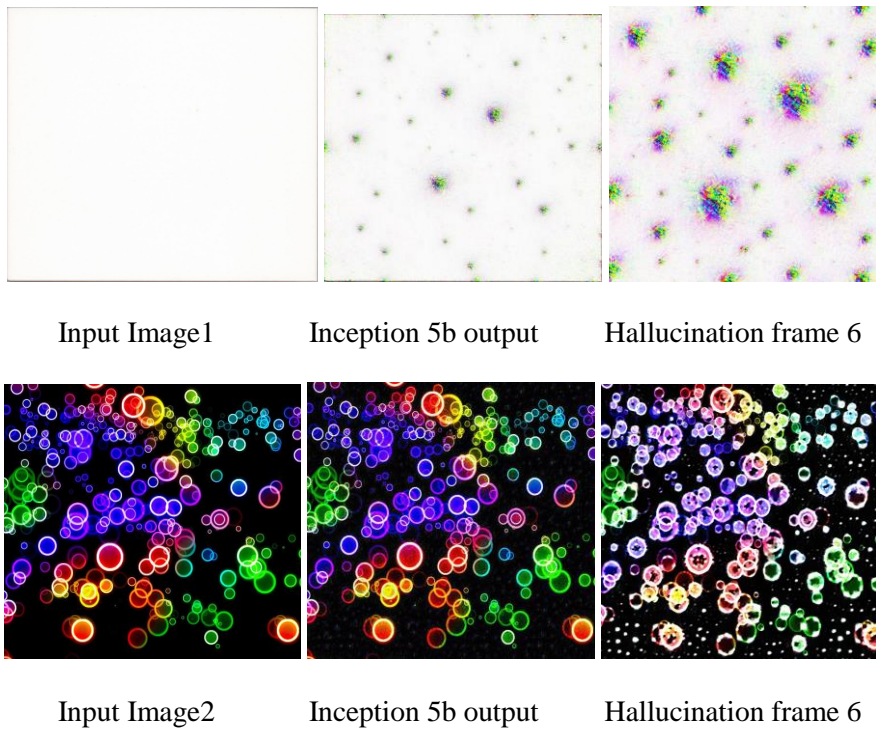
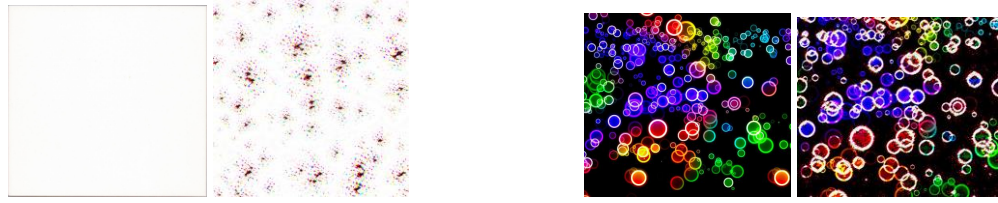


Fig 5.7.1 Output of bicycle training data on Inception\_5b/output layer of GoogleNet

Similar trimming was performed on rose data and following are results observed



Input image1 Hallucination frame 5      Input image 2 Hallucination frame 5

Fig 5.7.2 Output of roses training data for on AlexNet Pool5 layer

**Analysis:**

For bicycle data: From hallucinations of input images 1 and 2 it can be observed that the network has learnt the structure of bicycle well. As the training data has trimmed and tuned bicycle images the hallucination of image 2 identify the bubbles of image as bicycle tires irrespective of bubble colors.

For rose data: From images 1 and 2 it can be observed that the network has understood the structure of roses better. A good data set identifies the object irrespective of the color of the object

**Conclusion:**

By trimming away unwanted images and adding more images without background data, the network has improved the recognition capacity of bicycles.

**Uses:**

For bicycles: The class scores of test bicycle images without background data for the caffemodel are 0.820, 0.897. This indicate a very high probability of classification accuracy for images. The class scores of test bicycle images with background data for the caffemodel are 0.715, 0.703. This also indicate a good probability of classification accuracy for images with background data.

For roses: The class scores of test bicycle images without background data for the caffemodel are



0.928, 0.886. This indicate a very high probability of classification accuracy for images. The class scores of test bicycle images with background data for the caffemodel are 0.754, 0.796. This also indicate a good probability of classification accuracy for images with background data.

Thus using these techniques we can provide a good data set for real time classification models.

### 5.8 Observation 8:

Training set:

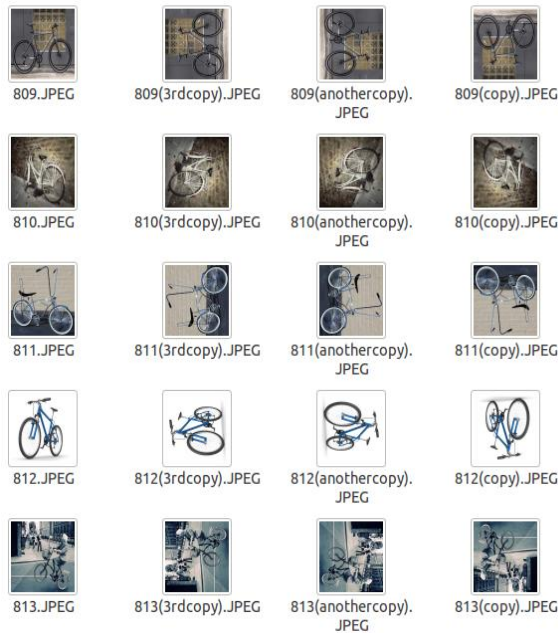
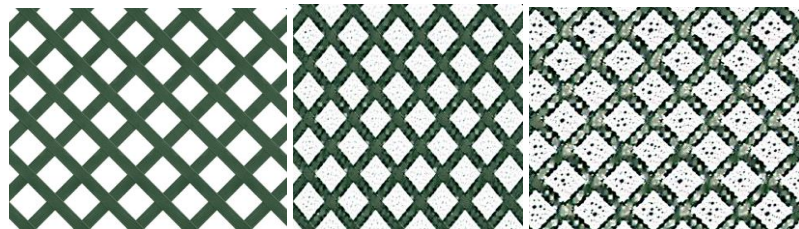
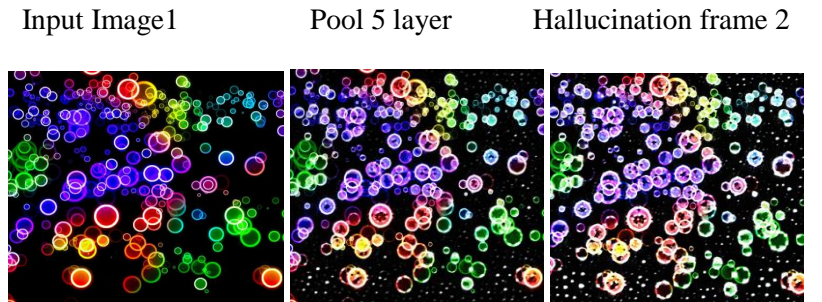


Fig 5.8. Bicycle data flipped and rotated

I have prepared bicycle data by flipping and rotating the data in different directions to observe the variations in the network efficiency.

Results Observed:





Input Image1                  Pool 5 layer                  Hallucination frame 2

Fig 5.8.1. Output of rotated bicycles on Pool 5 layer of AlexNet

Analysis:

From hallucinations of input images 1 and 2 it can be observed that the network has learnt the structure of bicycle well. As the data has many images of bicycles oriented in different directions the hallucination of image 1 and 2 have produced circular structures similar to the tires of bicycle along with some other features.

Conclusion:

Flipping and rotating images helps in gaining a better understanding of structure as per real time data.

Uses:

The class scores of test bicycle images without background data for the caffemodel are 0.621, 0.568. This indicate a good probability of classification accuracy for images. The class scores of test bicycle images with background data for the caffemodel are 0.657, 0.694. This also indicate a good probability of classification accuracy for images with background data. Real time data has lot of different noise and the object of interest may be oriented in different directions and hence classification network should perform well for all images. By creating this dataset we can also increase the data set size which in turn increases network efficiency.

### 5.9 Observation 9:

Training set:

I have prepared dataset of images of roses. By training Alexnet for 100000 iterations and selecting different layers of Alexnet I got the following results.

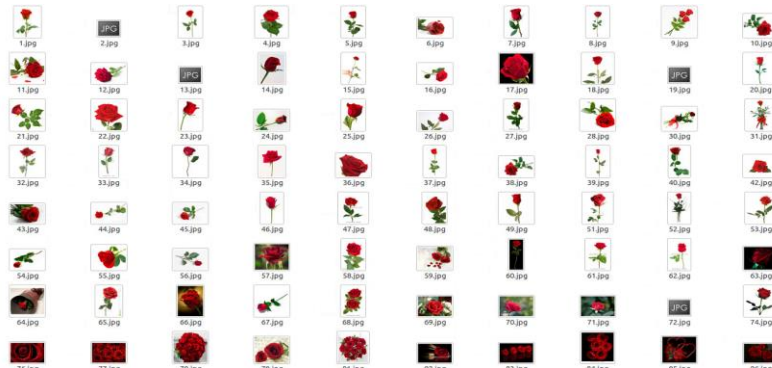
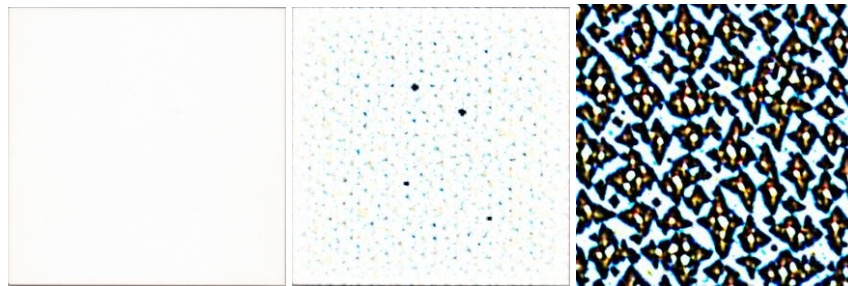


Fig 5.9. Rose images training set

Results Observed:



Input Image

Pool 3 layer

Hallucination frame 13

Fig 5.9.1 Output of roses training data on Pool 3 layer of Alexnet

Analysis:

We can observe that the network has not identified structure in the input image at the layers of Alexnet. The network has identified the structure of rose better in the previous observation. We have reduced the training set size for this analysis.

We have also observed the following results for different training set sizes

<b>Number of Images in Training set</b>	<b>Classification values</b>
<100	0 or no class score for the class
300	0.2 - 0.4, other classes are also predicted in results with high values
600	0.3 - 0.7
>800	0.5 - 0.99

Table 5.9 Number of training images vs Class values

Conclusion:

From here we can observe that the network it is important for the network to have enough training data of a category for it to understand the structure of an object better.

Uses:

If there training set size is less than 100 then the category is not recognized properly. But we can see that training set above 800 images produce good class scores for images. So it is better to have at least 800 objects of each category to train the network well.

5.10 Observation 10:

Also the validation data plays an important role in training and tuning the network well. The validation set and training set should be disjoint as the validation set helps in tuning the weights of the network after the training as per the data present. The following are the results observed for different validation set sizes. The values specified are sa percentages of number of images in training set.

<b>% of Validation images</b>	<b>Classification values</b>
< 5%	0 or no class score of the actual class
10%	0.2 - 0.7, sometimes no class score of actual class
>20%	0.5 - 0.99

Table 5.10 Percentage of validation images vs Class values

Conclusion:

From here we can observe that the network it is important for the network to have enough validation data of a category for it to tune the network better.

Uses:

It is better to have at least 20% objects of each category to validate the network well. Tuning the network helps in producing better results for real time data.

5.11 Guidelines to prepare and fine tune an image data set for training a deep convolutional network

- Color is an important aspect of training data so it is ideal to provide your object of interest in multiple colors so that the network can classify data independent of the color of the input image.
- Having training set with your object of interest in different angles helps the network to gain a better understanding of the network.
- Background data has a huge impact in the training if the network.
- Avoiding various images with similar backgrounds and including multiple backgrounds

helps the network to not identify background data as object of our interest.

- Trimming images with lot of background data and adding clean images to training set helps network to compare these images to the plain ones so that they can understand the object of interest in better manner.
- Having random clips of object of interest help the classification network in identifying the object of interest better in real time data.
- If you have a very small data set then flipping the images or rotating them will also help in improving network efficiency as well as increasing the size of training set
- Having at least 800 images in training set for each category will help in producing better results
- Having validation data set the size of 20% of training set will help in tuning the network well.
- Having a combination of 30% of clean images, 50% of noisy images and 20% random clips helps in producing better results.

## REFERENCES

1. Xavier Glorot, Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks” Artificial Intelligence Statistics Conference 2010.
2. <https://web.archive.org/web/20150703064823/http://googleresearch.blogspot.co.uk/2015/06/inceptionism-going-deeper-into-neural.html>
3. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. “Intriguing properties of neural networks”. arXiv preprints 2013
4. Anh Nguyen, Jason Yosinski, Jeff Clune. “Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
5. Understanding deep dream in simpler terms  
[https://www.reddit.com/r/explainlikeimfive/comments/3cbelv/eli5\\_can\\_anyone\\_explain\\_googles\\_deep\\_dream](https://www.reddit.com/r/explainlikeimfive/comments/3cbelv/eli5_can_anyone_explain_googles_deep_dream)
6. Caffe Installation. <http://caffe.berkeleyvision.org/installation.html>
7. Aravindh Mahendran, Andrea Vedaldi. “Understanding Deep Image Representations by Inverting Them” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015
8. Understanding convolutional networks. <http://cs231n.github.io/convolutional-networks/>
9. Convolutional networks <https://www.youtube.com/watch?v=n6hpQwq7Inw>
10. R. J. Williams “Inverting a connectionist network mapping by backpropagation of error.” Proceedings of Eighth annual conference of IEEE 1986

11. Alexey Dosovitskiy, Thomas Brox “Inverting Visual Representations with Convolutional Networks” arXiv preprint 2015
12. Jonathan Long, Evan Shelhamer, Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015
13. Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton. “Imagenet classification with Deep Convolutional Neural Networks” Advances in Neural Information Processing Systems (NIPS 2012)
14. Leon A. Gatys, Alexander S. Ecker, Matthias Bethge. “A neural algorithm of artistic style” In ArXiv e-prints 2015
15. Asha Anooosheh, Rishi Kapadia, Jared Rulison “Image transformation via Neural Network Inversion” 2015
16. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Ann Arbor. “Going Deeper with Convolutions” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015
17. Setup of an image classifier based on imagenet  
<https://deepdetect.com/tutorials/imagenet-classifier/>
18. Downloading and installing DeepDetect <https://deepdetect.com/overview/installing/>
19. Deep neural networks <http://neuralnetworksanddeeplearning.com/chap1.html>
20. Inception: Going deeper into neural networks  
<https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>
21. Using neural networks to recognize hand written digits  
<http://neuralnetworksanddeeplearning.com/chap1.html>



VITA

Kavya Jalla

Candidate for the Degree of

Master of Science

Thesis: FINE TUNING IMAGE INPUT SETS FOR DEEP NEURAL NETWORKS USING HALLUCINATIONS

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December, 2016..

Experience:

Professional Memberships: