# Convolutional Autoencoders

Zhongzheng Shu, Bhawna Shiwani, and Mostafa El Gamal

## Introduction

The main goal of our project is to use unlabeled images to improve the classification accuracy of a convolutional neural network. In practical applications, the amount of available unlabeled data is larger in size than that of the labeled data. The natural question that follows is: how unlabeled data can be used to pretrain a neural network in order to improve its classification\regression accuracy. To achieve our goal, we experiment the possibility of mixing convolutional networks with autoencoders. We experiment on a dataset (STL-10) that has been studied in the literature since 2011. The data consists of 5,000 labeled images for training, 8,000 labeled images for testing, and 100,000 unlabeled images. Each image has a size of 96x96 pixels in color and represents one of 10 classes (airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck).

The small amount of labeled data makes it difficult to achieve good classification accuracy. Therefore, it is desired to exploit the huge amount of unlabeled data to improve that accuracy. The current attempts in the literature managed to achieve classification accuracies that range from around 50% to around 74%. The report is divided into two main sections. 1) Supervised learning: in this section we try different network architectures to achieve the best classification accuracy using the labeled data solely; 2) Unsupervised learning: in this section we apply the concept of autoencoders to pre-train a convolutional network using the unlabeled data. At the end, we compare the results obtained in the two sections to examine the performance change after applying the unlabeled data to the network.

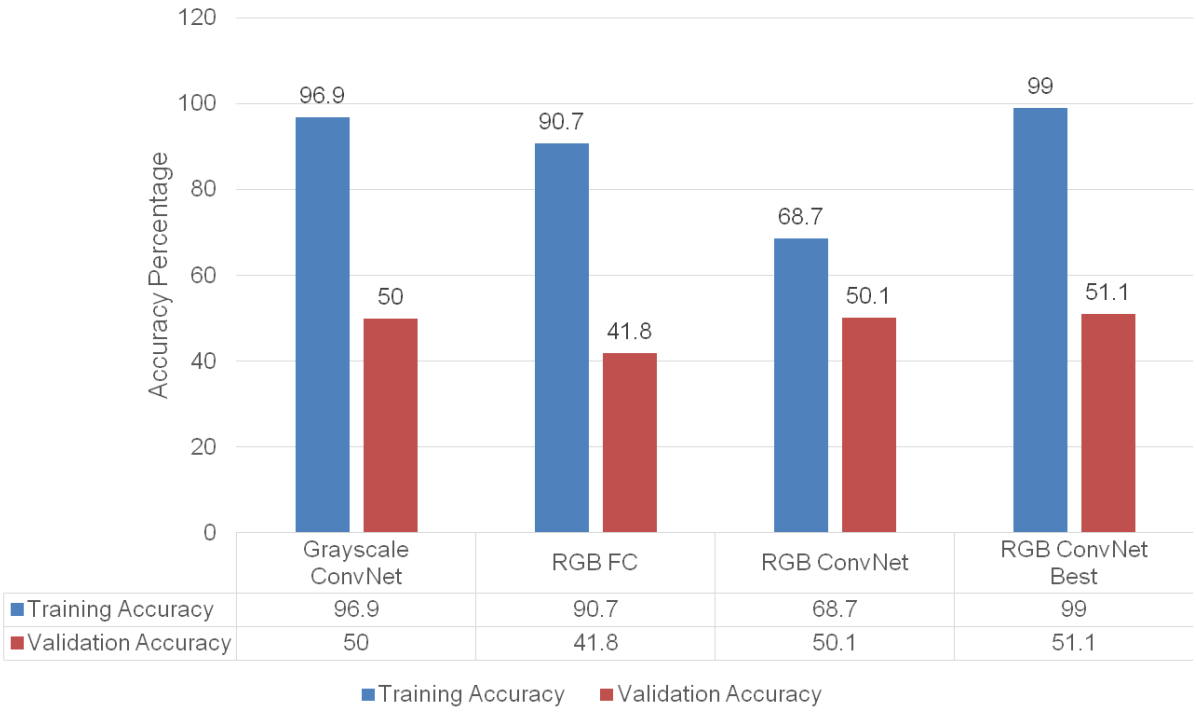
## Section 1: Supervised Learning

In this section, we performed different experiments to achieve the best classification performance using the 5,000 labeled images. Our experiments included:

1. Fully connected neural network.
2. Convolutional network.

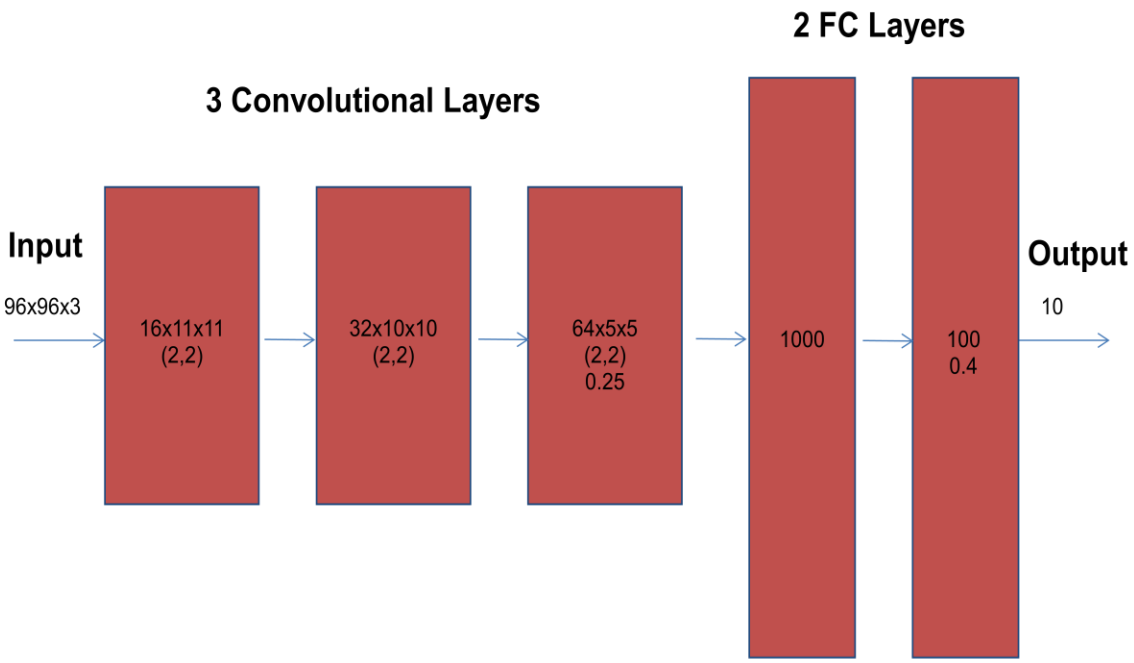
We also optimized different parameters such as; dropout rate, momentum, rectification, ...

All our experiments were done on both colored images and gray scale ones. The figure below summarizes the results of our experiments.




| | Grayscale ConvNet | RGB FC | RGB ConvNet | RGB ConvNet Best |
|---|---|---|---|---|
| Training Accuracy | 96.9 | 90.7 | 68.7 | 99 |
| Validation Accuracy | 50 | 41.8 | 50.1 | 51.1 |

As shown in the figure, the best validation accuracy was achieved using a convolutional network with colored images as its input. Below is the architecture of that network.



The network consists of three convolutional layers and two fully connected layers. Dropout has been applied to the third convolutional layer and the second fully connected layer with rates of 0.25 and 0.4, respectively. A momentum of 0.8 and max pooling of size (2,2) were used. The training was done over 100 epochs and a 5 fold cross validation was used to get the validation accuracy.

The corresponding test accuracy of that network is around **52%**, and the confusion matrix is shown below.

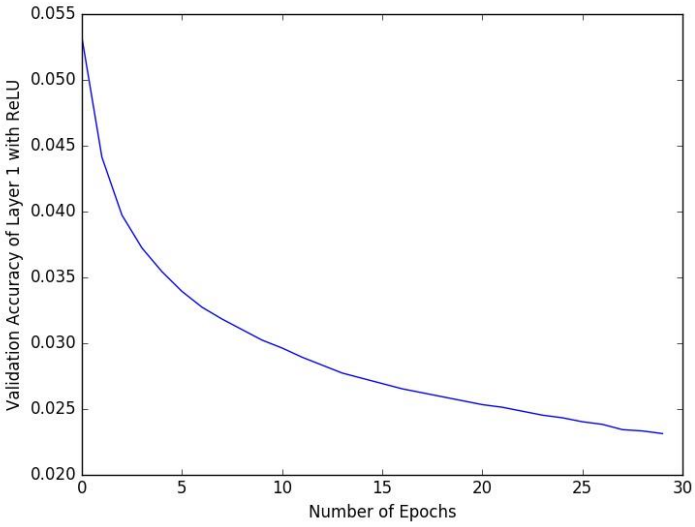
| | airplane | bird | car | cat | deer | dog | horse | monkey | ship | truck | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **airplane** | 601 | 46 | 23 | 7 | 10 | 9 | 3 | 13 | 51 | 37 | 800 |
| **bird** | 32 | 420 | 8 | 81 | 29 | 61 | 10 | 133 | 14 | 12 | 800 |
| **car** | 17 | 28 | 555 | 10 | 1 | 17 | 6 | 9 | 28 | 129 | 800 |
| **cat** | 4 | 130 | 8 | 273 | 42 | 88 | 35 | 194 | 13 | 13 | 800 |
| **deer** | 9 | 112 | 1 | 94 | 315 | 62 | 66 | 124 | 9 | 8 | 800 |
| **dog** | 8 | 134 | 6 | 119 | 39 | 199 | 94 | 194 | 2 | 5 | 800 |
| **horse** | 5 | 53 | 4 | 33 | 36 | 90 | 426 | 140 | 4 | 9 | 800 |
| **monkey** | 4 | 120 | 3 | 86 | 19 | 108 | 49 | 399 | 3 | 9 | 800 |
| **ship** | 72 | 20 | 29 | 18 | 8 | 6 | 2 | 5 | 563 | 77 | 800 |
| **truck** | 39 | 25 | 121 | 25 | 8 | 11 | 11 | 27 | 61 | 472 | 800 |
| **Total** | 791 | 1088 | 758 | 746 | 507 | 651 | 702 | 1238 | 748 | 771 | |

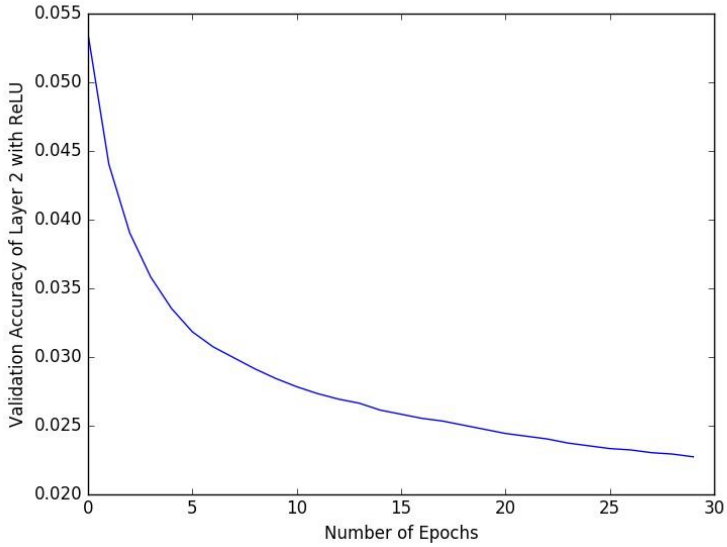
It is clear that the network tends to predict (monkey and bird) when the image shows another animal. Also trucks and cars are mixed up in a number of occasions. In the following section, we try to use the unlabeled images to achieve better classification accuracy.

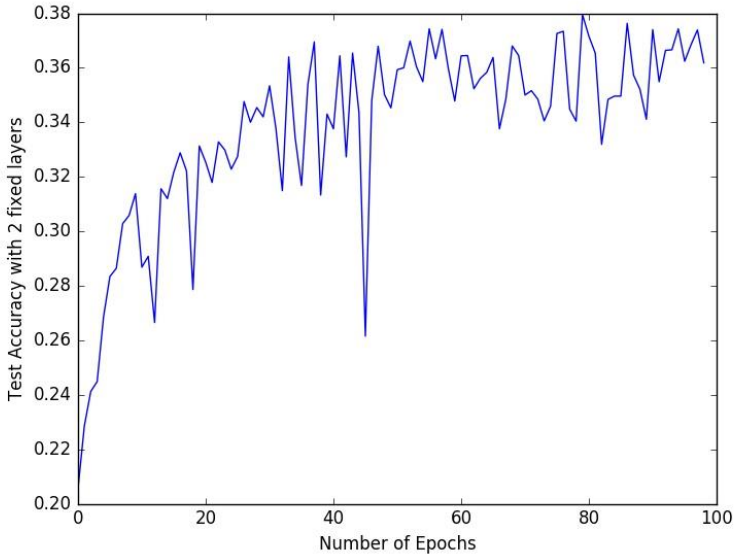
## Section 2: Unsupervised Learning

In this section we pretrain the convolutional network obtained in the previous section using the concept of autoencoders. That requires the reconstruction of the input images after a single convolutional layer, and the minimization of the reconstruction error. To achieve that, first we tried the simple autoencoder pretraining layers using the gray scale images. With one layer of autoencoders the best accuracy achieved was 30%. Then we pretrained second layer of autoencoders as well and the accuracy increased to 37%. As it is clear from the results below simple autoencoder layers did not prove that effective, as the validation accuracy stops

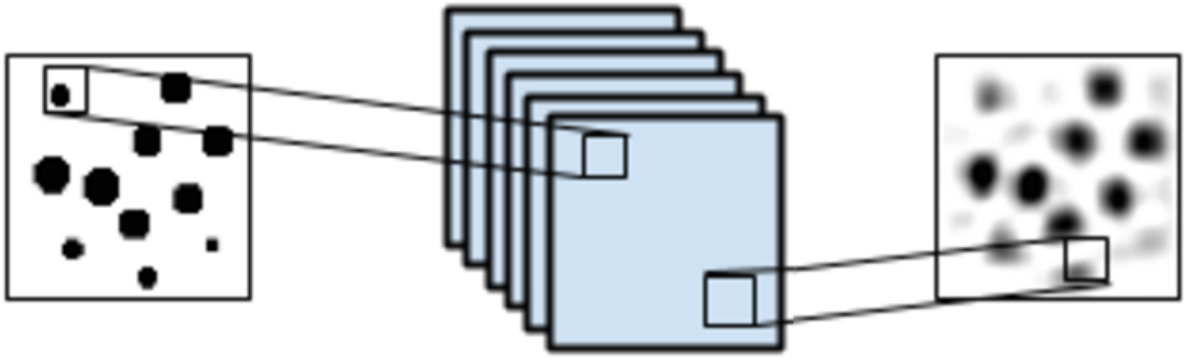
decreasing after certain epochs. The test accuracy also doesn't improve much after 37%. One possible reason of this could be the large image size and the slow feature learning as complete image is considered at a time.




To solve this problem, we built a deconvolutional operation that reconstructs the image from the convolutional patches. The diagram below illustrates the details of that process.



The image is divided into a number of patches, and then the feature maps are formed by mapping the patches into pixels. The deconvolutional operation is simply multiplying each pixel of the feature maps by a filter of the same size as the patch. The deconvolutional filter is trained and validated on the 100,000 unlabeled images in order to minimize the images reconstruction error. The sizes of the training dataset and the validation dataset are 90,000 images and 10,000 images, respectively.

**Pretraining one convolutional Layer**

Accuracy: 0.1117

The training MSE dropped from 0.03 to 0.02 over 5 epochs, while the validation MSE dropped from 0.021 to 0.020. Due to time constraints, we did the training of the convolutional autoencoder over the gray scale images. Each training epoch took around 50 minutes to be completed.

| □ | airplane | bird | car | cat | deer | dog | horse | monkey | ship | truck | TRUE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| airplane | 0 | 0 | 21 | 779 | 0 | 0 | 0 | 0 | 0 | 0 | 800 |
| bird | 0 | 0 | 41 | 758 | 0 | 0 | 1 | 0 | 0 | 0 | 800 |
| car | 0 | 0 | 110 | 690 | 0 | 0 | 0 | 0 | 0 | 0 | 800 |
| cat | 0 | 0 | 16 | 784 | 0 | 0 | 0 | 0 | 0 | 0 | 800 |
| deer | 0 | 0 | 17 | 783 | 0 | 0 | 0 | 0 | 0 | 0 | 800 |
| dog | 0 | 0 | 22 | 778 | 0 | 0 | 0 | 0 | 0 | 0 | 800 |
| horse | 0 | 0 | 67 | 733 | 0 | 0 | 0 | 0 | 0 | 0 | 800 |
| monkey | 0 | 0 | 28 | 772 | 0 | 0 | 0 | 0 | 0 | 0 | 800 |
| ship | 0 | 0 | 12 | 788 | 0 | 0 | 0 | 0 | 0 | 0 | 800 |
| truck | 0 | 0 | 75 | 725 | 0 | 0 | 0 | 0 | 0 | 0 | 800 |
| Pred | 0 | 0 | 409 | 7590 | 0 | 0 | 1 | 0 | 0 | 0 | |

In this case as we see the results are not very good and are highly biased towards cats. In the second experiment we tried training 2 layer convolutional neural network.

**Pretraining two convolutional Layers**

Accuracy: 0.1

| □ | airplane | bird | car | cat | deer | dog | horse | monkey | ship | truck | TRUE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| airplane | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| bird | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| car | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| cat | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| deer | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| dog | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| horse | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| monkey | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| ship | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| truck | 0 | 0 | 0 | 0 | 800 | 0 | 0 | 0 | 0 | 0 | 800 |
| Pred | 0 | 0 | 0 | 0 | 8000 | 0 | 0 | 0 | 0 | 0 | |

In this case also the results are all deer. This indicates some issues is network structure or the training methods. Just to verify we trained a 3 convolutional layer network.

**Pretraining three convolutional Layers**

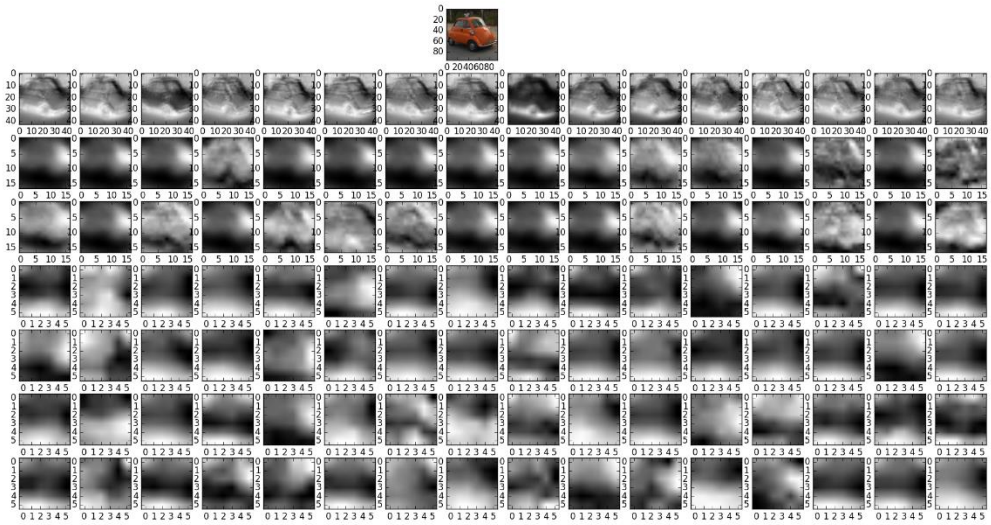
Accuracy: 0.1

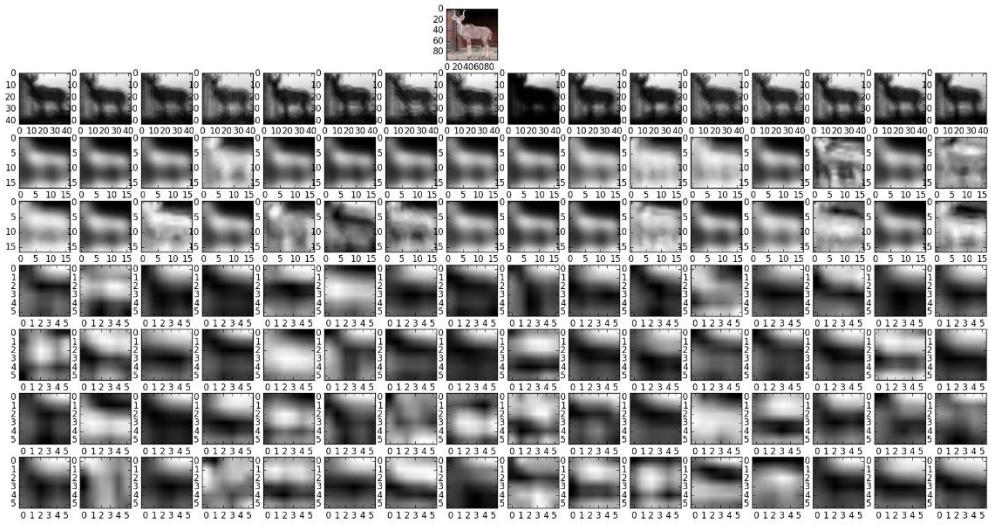
| □ | airplane | bird | car | cat | deer | dog | horse | monkey | ship | truck | TRUE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| airplane | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| bird | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| car | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| cat | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| deer | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| dog | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| horse | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| monkey | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| ship | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| truck | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 800 |
| Pred | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8000 | |

**Discussion:**

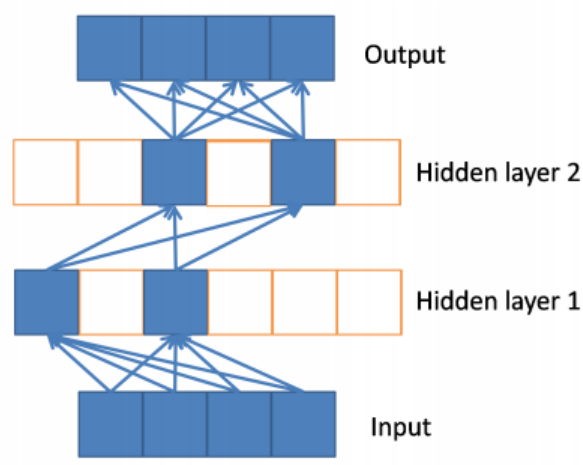
As the results are not improving in any case, we decided to look into the feature maps.

Below are samples of the feature maps obtained from training one layer of the convolutional autoencoder.





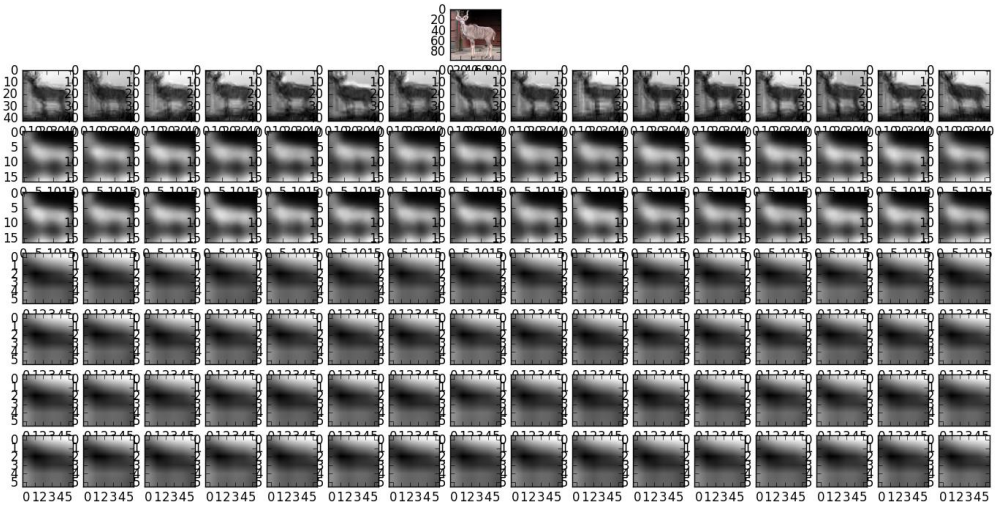
In this case the first row contains the feature maps from one-layer convolutional network. The second and third row contains feature maps from two-layer convolutional network and the last four rows contains the feature maps from three-layer convolutional network. As we see in this case the third layer is kind of abstract. On possible reason of this is the activation function that is being used. We are using ReLU in this case which sets all negative weights to zero and applies linear rectification on positive weights. ReLU is a good way of sparse representation, which works very well for supervised learning. But in our case, applying ReLU to both encoder and decoder phases is not a good idea, since we are losing information twice and trying to reconstruct input afterwards. The figure below shows the network structure with 2 hidden layers with ReLu as activation function.




**Results with Sigmoid Activation:** After this we tried to pretrain convolutional layers using sigmoid activation function. And the test accuracy of respective layers is documented in the table as below.

| # pretraining convolutional layers | Test accuracy |
|---|---|
| 0 | 0.5010 |
| 1 | 0.4035 |
| 2 | 0.2855 |
| 3 | 0.2148 |

In this case the results are certainly better than using ReLU. The feature maps of convolutional layers in this case are:



However as the number of convolutional layers increases the accuracy drops instead of increasing. One possible reason of this can be the deconvolution process. As during the deconvolutional process the step of upsampling is involved and it is not possible to regain the information lost while subsampling. This as the number of layers in convolutional autoencoder increases the images after reconstruction becomes less and less accurate.

This problem can possibly be solved by using the approach mentioned in stacked what-where autoencoders[3]. Which is included in the future scope of the project. In addition, in order to keep the advantage of ReLu in AutoEncoder, we can apply ReLu in encoder phase but a linear function in decoder phase, which is proved to be better in [5].

**References**


[1] Adam Coates, Honglak Lee, Andrew Y. Ng “An Analysis of Single Layer Networks in Unsupervised Feature Learning”, AISTATS, 2011.

[2] Tsung-Han Lin, H. T. Kung “Stable and Efficient Representation Learning with Nonnegativity Constraints”, ICML, 2014.

[3] Junbo Zhao, Michael Mathieu, Ross Goroshin, Yann LeCun “Stacked What-Where Auto-encoders”, ArXiv, 2016.

[4] Mike Swarbrick Jones “Convolutional autoencoders in python/theano/lasagne”, Blog, 2015.

[5] Xavier Glorot, Antoine Bordes, Yoshua Bengio “Deep Sparse Rectifier Neural Networks”, AISTATS, 2011