Symptotic Analysis of Autoencoder Architectures for Image Colorization and Noise Reduction

Project Report for DS 303, Introduction to Machine Learning, Spring 2021

Nikhilesh Rajput
Dept. of Electrical Engineering
IIT Bombay
200070067@iitb.ac.in

Rohan Kalbag
Dept. of Electrical Engineering
IIT Bombay
20d170033@iitb.ac.in

Shivam Patel
Dept. of Electrical Engineering
IIT Bombay
200070077@iitb.ac.in

***Abstract*—The goal of this study is to examine the various methods, algorithms and paradigms that we can derive from neural networks, specifically shallow Artificial Neural Networks (ANNs), to colorise images and create noise free image reproductions. This paper focuses on the Autoencoder architecture, which is usually a shallow hourglass shaped neural network implementation, with limited number of trainable parameters. This architecture can prove vital for the above applications in cases of computational hardware limitations. We try to examine the effect of higher number of epochs during training, and at what point is the 'elbow point' attained. We also try to examine and contrast the performance of Autoencoders and PCA for noise reduction.**

***Index Terms*—Neural Networks, Autoencoders, PCA, Image Colorification, Noise Reduction**

I. Introduction

Neural Networks where introduced in 1943 as a logical abstraction to the functioning of logical and memory based tasks of our brain. This idea of electrically and mathematically modeling the brain was first set in motion by mathematician Walter Pitts and neurophysiologist Warren McCulloch. When the Von Neumann Architectures took over the computer industry, neural network research was neglected and left behind. Some controversial papers suggesting that neural networks cannot be extended to non-linear decision abilities even further led to decline in research inquisitiveness and funding.

But with the advent of faster and more powerful computers in the industry, research on this computationally intense field took pace. Soon came the arguments based on the complexity of decision making abilities based on the number of layers of the neural networks and the nodes of the network. Since then, there has been no looking back. Remarkable contributions from researchers have led to the discovery of applications in stock market prognosis, facial recognition, self driving cars, medical diagnosis, defense, weather forecasting etc.

Neural Networks are divided into three categories, depending on the semantic data type that they are utilised for, and the type of architectures implemented -

- **Artificial Neural Networks** - ANNs are capable of learning any non-linear decision boundaries and functions. These are excellent for predicting decision boundaries and functions in tabular and numerical data. But due to limitations posed by exploding and vanishing gradients, non accountability for serial and temporal data like text and time scale stock market data respectively.
- **Convolutional Neural Networks** - Training ANNs on image data can prove to give respectable results, but the computational overhead involved due to the large number of parameters attributed to multiple layers and large pixel number. This is overcome by using CNNs, which allows for a translationally invariant 'filter' which can be parsed over the image to retrieve/ emphasize different desired features from the image.
- **Recurrent Neural Networks** - The shortcomings of unaccountability of the above networks to serial and temporal data led to the development of RNNs, which have essentially a 'feedback' element to the neural net. This enables us to parameterize the distance between different data points in a sentence, stock market data, audio files etc.

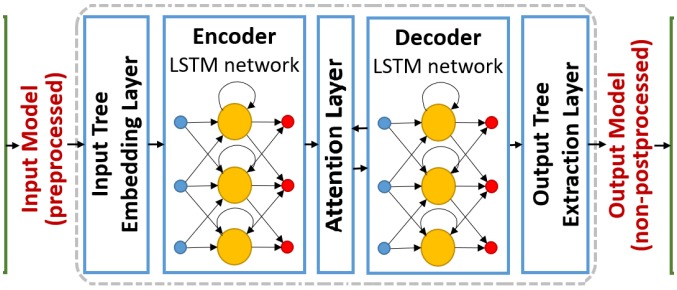

Fig. 1. RNN Architecture

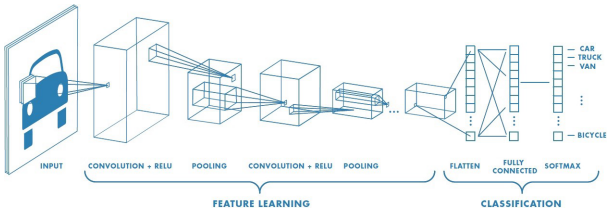

Fig. 2. Convolutional NN

Image Colorization

Historically, CNNs have been used to predict color palettes of different shapes and objects in an image, by somewhat rote-learning of models. This was achieved using a very large dataset, and training it to judge the relationship between colours of objects and object shapes in the image. For example, trees have a distinctive shape, and green color is strongly associated with trees. Similar mappings can be generated from different object shape contours to their colours. A sufficiently large trained model can be used to colorize arbitrary images, although to different and varying outcomes. But conventional CNN approach has a large number of trainable parameters, which is computationally intensive. Also, this method is sensitive to outliers/noise in the image, which might give incorrect color mappings.

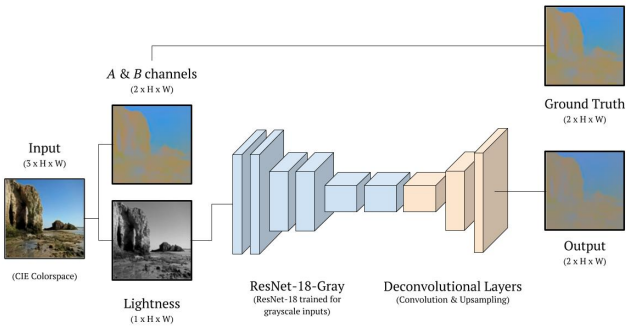

Fig. 3. Image Colorisation using ResNet based Encoding Architecture

Noise Reduction

Noise reduction in data, especially images and audio, is a vital and crucial part of various analytical procedures, where the applied algorithms are highly sensitive to outliers. Such applications include medicine and healthcare, defense, nano-research etc. Noisy data can lead to mis-predictions, which can be devastating in the above mentioned fields. Hence, there is a crunching need for reliable noise reduction techniques. Conventional ML gives us some time tested techniques for noise reduction, such as PCA, SVD etc. But these algorithms involve taking matrix inverses, which is very computationally intensive, especially with a large number of features as image pixels. The Big-O of these algorithms is $\mathcal{O}(n^3)$, which is the bottleneck when dealing with large number of datapoints like image pixels.

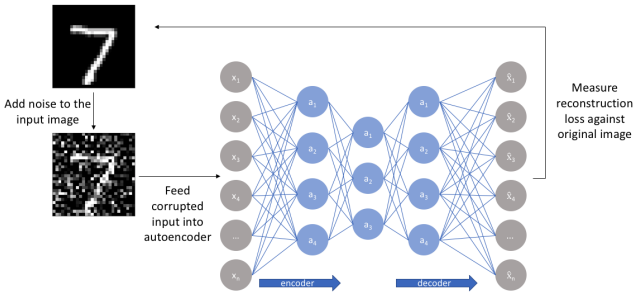

Fig. 4. Autoencoders as denoise-ing networks

II. Autoencoder Architecture

Autoencoders are essentially feedforward ANNs, which are used extensively for image feature mapping and representational/feature mapping. The structure of autoencoders is such that it allows for a 'bottleneck' in the feedforward network, which allows for the network to essentially summarize the features present across all training images. Thus, they often behave to encode the common feature maps in a lower dimensional subspace, hence the name 'Autoencoders'. There are three integral components of any autoencoder implementation -

1) **Encoder** - This maps from the input layer which equals the number of pixels, to a layer with lower number of nodes, which fundamentally means reducing the dimensions of the input to a lower dimensional vector, corresponding to the major features of the image.
2) **Coded vector** - This layer has the smallest number of hidden layers in the entire architecture, and it semantically represents the encoded information of the entire image.
3) **Decoder** - The decoder layers take the coded parameters obtained from the coded vector, and obtains an image through scaling up the dimensions over successive layer, trying to replicate the original input image.

Such a dimensionality reduction and subsequent upscaling after feature learning is possible due to correlation in the training data. Unless we have similar features in the training data, such an autoencoder scheme is not guaranteed to work. We may get inefficient and inaccurate outputs. Some properties of the autoencoder architectures are -

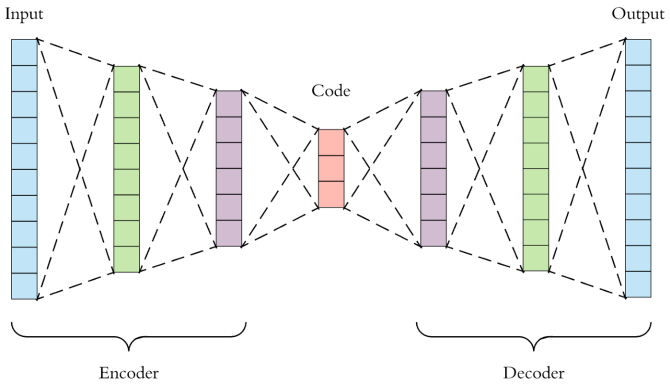

Fig. 5. Basic Autoencoder

- It is specific on the data that it is trained on. Do not expect it to reconstruct images of a car if it has been trained on a dataset of flowers or fruits.
- It is often called the PCA of the NN world, due to its dimensionality reduction abstraction
- Autoencoders are unsupervised learning algorithm, as the input and the output is more often than not the same image vector.

III. Autoencoders for Image Colorization

For image colorization of CIFAR10 dataset, we have implemented a dense layer based autoencoder, where each layer is realized using fully connected layers, and the weights are trainable.

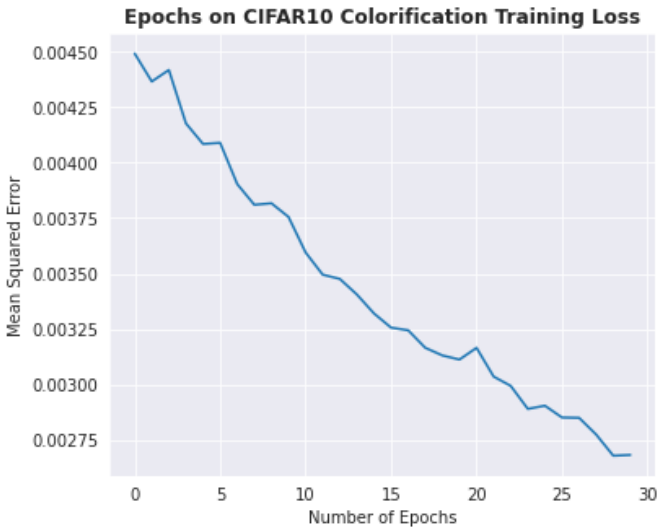

Fig. 6. Training Loss of the model with Epochs

This enables us to reduce the dimensions of the otherwise computational intensive neural network. Training the model for over 30 epochs gives a respectable training error which eventually stabilizes. The autoencoder is trained on 10k images of ships, which are 32x32 RGB channel images. The autoencoder learns the features and color mappings which are common to all images of ships, and tries to replicate the color mappings on grayscale images provided. Examining the output of the model, we find that there is strong correlation between the colors of the original image and the colorized image created by the autoencoder.

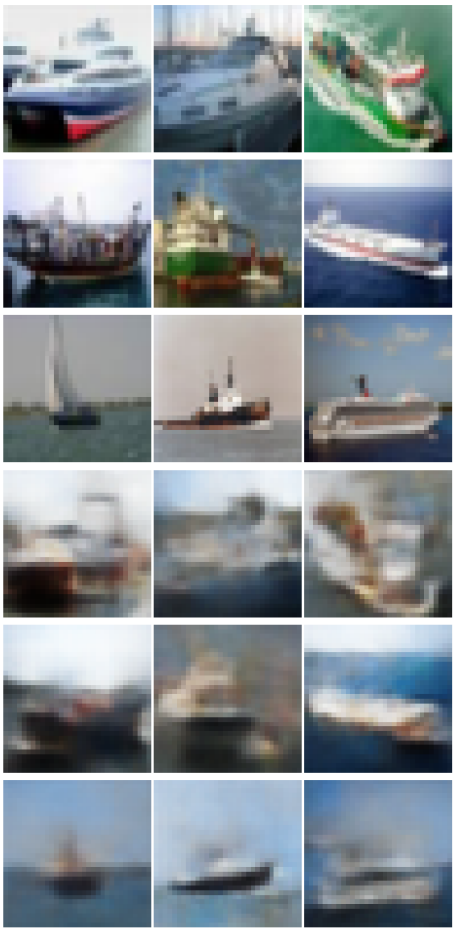

Fig. 7. Image Colorization on Ship Images (actual and reconstructed images in upper and lower halves respectively)

The data-specificity of autoencoders can be studied by predicting colors of images that are not from the ship class. For this analysis, we try to color images of airplanes, and plot the colorized images to compare and contrast the results. A first glance at the images reveals that the images of airplanes

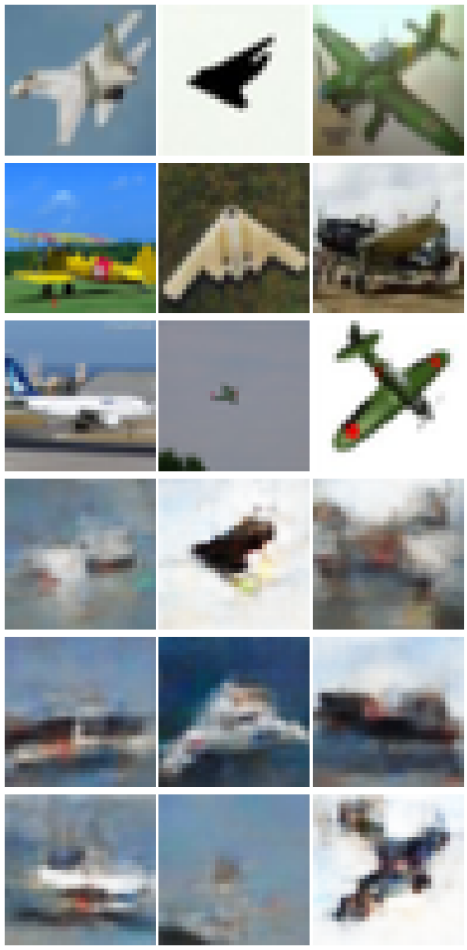

Fig. 8. Airplanes colorized using model trained on ship images

have forcefully been colored blue by the model, indicating that the feature of the 'water bodies' which are blue in color is learned by the model, and airplane images are also forcefully being colored blue.

IV. Autoencoders for Noise Reduction

For analyzing the noise reducing characteristics of specifically trained autoencoders, we use the digit dataset from MNIST, having 60k training images and 10k test images, each of size 28x28 pixels. We add gaussian noise to the dataset, and try to retrieve the noise free data from the autoencoder we trained. The model has been trained for 20 epochs with an Early Stopping regularizer with patience of three.

The model learns the digits accurately, even with a small number of hidden layers and latent parameters in the encoded vector. At first sight of the de-noised images of digits, we cannot directly see the

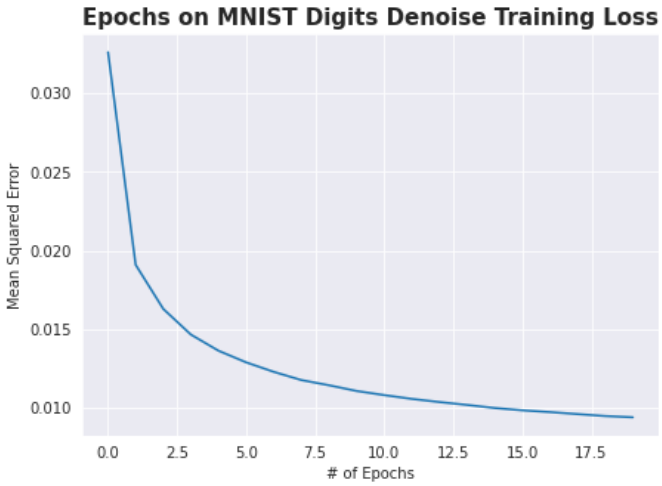

Fig. 9. Training Error of Denoising Autoencoder

difference between the original and reconstructed images. After close examination, we find that some digits in the train data are mutated to their more common/correct version, given that they had anomalies in terms of their visual appearance, as compared to the more general and correct handwritten form of the digit.

V. Autoencoders vs PCA for Dimensionality Reduction

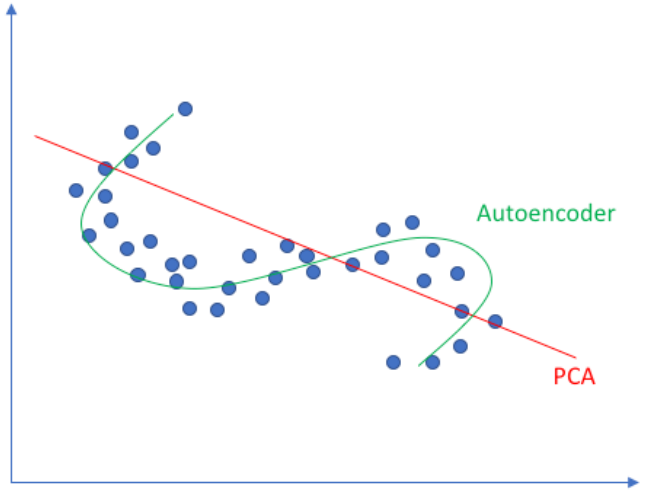

Fig. 10. Autoencoders vs PCA for non-linear dimensionality reduction

Principal Component Analysis is a conventional ML algorithm, which is widely used for representing the given data into a lower dimensional subspace, so that we can preserve the maximum explainable variance in the data. This is achieved by

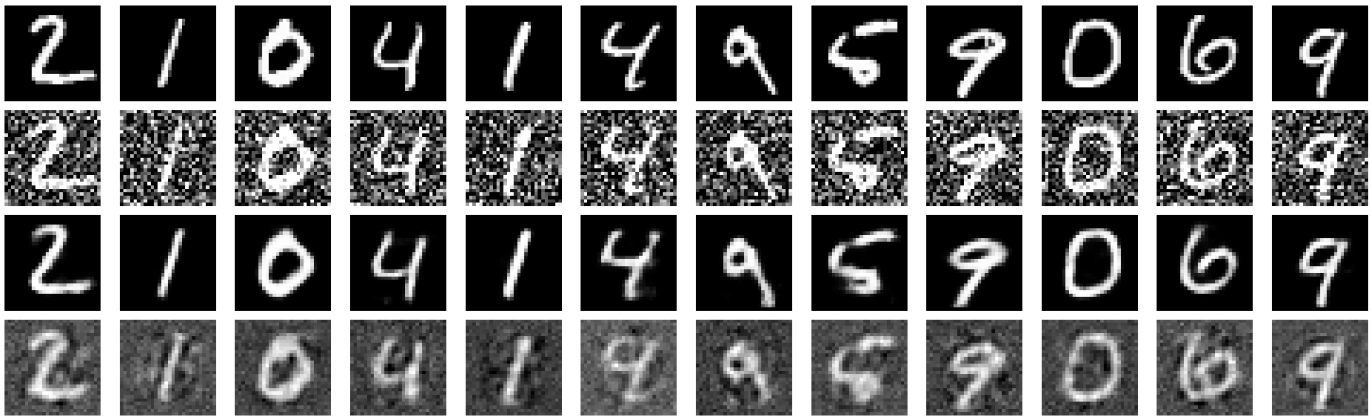

Fig. 11. Image Noise Reduction using Autoencoders, rows- (original, gaussian noise, autoencoder reconstruction, PCA reconstruction)

representing our data in the vector space represented by the 'k' eigenvectors of the covariance matrix with highest eigenvalues by magnitude. This is a mathematical procedure, and involves taking the inverse of matrices, which is a $\mathcal{O}(min(p^3, n^3))$ algorithm, where p is dimension and n is number of datapoints. This becomes very slow, especially on data sets with very large dimensional space.

This difficulty is overcome by using autoencoders, which are faster to train and work with large datasets. Learnable features and varying architectures allow for different types of dimensionality reduction, like simple or manifold etc.

Metric	PCA	Autoencoders
Linearity	Linear	Non Linear
Small Dataset	Faster	Slower
Large Dataset	Slower	Faster
Hyperparameters	N-dim	Architecture

Comparision of PCA and Autoencoders

For experimentally comparing and contrasting the noise reducing abilities of PCA and autoencoders, we perform the denoising of MNIST digits dataset, with gaussian noise added to it.

- **PCA** gives us options of choosing various different dimensions of reduced components/eigenvectors, and hence varying extents of reconstruction accuracy. Having a very small number of n_components in PCA, we get meaningless reconstruction, where all images look the same for different digit classes. This is because the lower number of PCA dimensions is not enough to capture the variability in data for making predictions on the digit classes. A moderate choice of n_components yields an optimal reconstruction. Even larger n_component value makes the reconstruction noisier, indicating that the PCA reconstruction has enough dimensions to even model the noise. Hence, the right choice of the hyperparameter n_components is essential for denoising of images, which is very computationally intensive to calculate. This problem is overcome by using autoencoders.
- For **autoencoders**, we observe that a densely connected model performs well and the reconstructed image is almost exactly identical to the original noise free image. This gives us a strong confidence in our prediction, and we can easily infer important features of our original image from the reconstructed image.

Model Name	Epochs Trained	Final Loss (MSE)	Training Time(s)
CIFAR Denoise	50	0.0138	575
CIFAR Colorize	30	0.0027	255
MNIST Denoise	20	0.0093	540

Performance Metrics for the Trained NNs

VI. Inferences and Conclusions

Thus Autoencoders are a versatile tool that can be used for applications such as Image Colorization and Noise Reduction.

- Through the results obtained for the colorization of the ships subset of the CIFAR-10 dataset we see that the Autoencoder learnt features specific to ships such as ships are surrounded with water

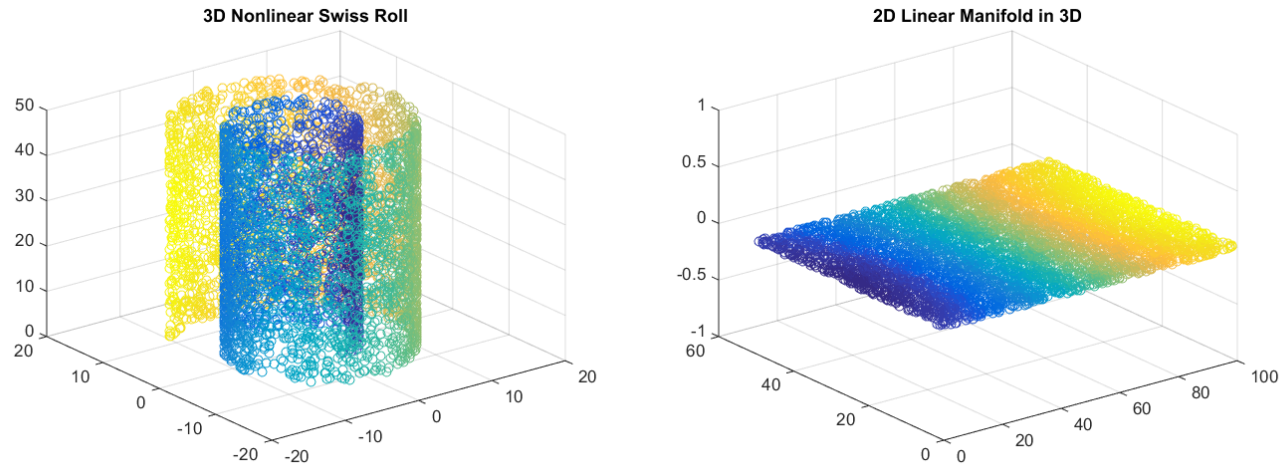

Fig. 12. Manifold Learning using Autoencoders

around the hull and surrounded with air above it, and predicted the colors correctly which can be seen in Fig. 7.

- We notice that the predictions are gibberish if we feed the Airplane train data to the model trained on the Ship dataset subset which can be seen in Fig. 8. Thus Autoencoders are class specific when it comes to colorization. For optimized performance in colorization we require the Autoencoder to be trained specific to the application. i.e if the application is to color images of ships, then training over the entire CIFAR-10 dataset can lead to erroneous results.
- For the MNIST dataset from the results in Fig. 11, we see that the Autoencoder's nearly eliminates the entire noise. Meanwhile, the PCA algorithm just reduces the noise upto a considerable extent. PCA doesn't learn features, it re-iterates an algorithm to reach the conclusion. Hence, we need large number of calculations to be done for every iteration. On the contrary, Autoencoder will learn and calculations are much more simple. Hence, we must prefer the Autoencoder Architecture.
- On training the Autoencoder Architecture on the Ship Subset of CIFAR-10. We notice that the images produced by this model have learnt features such as color of water, air. But we notice that the images do not have gaussian noise which is what we are looking for. But there is a lot of distortion in the output. This is because we have used a very small dataset of only 5000 images from the CIFAR-10. This can be fixed by using a larger training dataset of ships alone. Using the entire CIFAR 10 would lead to erroneous results as Autoencoders learn class specific as seen before.

ACKNOWLEDGEMENTS

We would like to thank Prof. Biplab Banerjee for teaching us this course with enthusiasm and detail, without which we would not have been able to complete this project. We would also like to thank all the TA's involved with this course, who helped us in doing the various assignments and were there to help us wherever we faced difficulties.

REFERENCES

[1] "Anomaly Detection Using Autoencoders" : [Online]. Available
[2] "Convolutional Autoencoders for Image Noise Reduction" : [Online]. Available
[3] "Autoencoders in Pytorch": [Online]. Available
[4] "Introduction to Autoencoders" : [Online]. Available
[5] "Autoencoders and its Variants, Zhang, et. al." [Online]. Available. IEEE
[6] "Autoencoders in Tensorflow": [Online]. Available
[7] "Colorization Autoencoders using Keras": [Online]. Available

HONOR CODE

The code and python files used in this project are completely generated by us. Use of experimentally obtained architectures is mentioned above with due mentioning of the owners. The image sources also have been mentioned as links in the respective figures.