

An Autoencoded Transductive Support Vector Machine for Noisy Image Classification

*Report submitted in fulfillment of the requirements
for the B.Tech Project of*

Fourth Year B.Tech.

by

Khushi Agrawal (17075060) and Janhavi Gupta (17075061)

Under the guidance of
Prof. K.K. Shukla



Department of Computer Science and Engineering
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India
November 2020

Dedicated to

*My parents, teachers, family and
friends...*

Declaration

We certify that

1. The work contained in this report is original and has been done by ourselves and the general supervision of our supervisor.
2. The work has not been submitted for any project.
3. Whenever we have used materials (data, theoretical analysis, results) from other sources, we have given due credit to them by citing them in the text of the thesis and giving their details in the references.
4. Whenever we have quoted written materials from other sources, we have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi
Date: 20/11/2020

Khushi Agrawal(17075060), Janhavi Gupta(17075061)
B.Tech. Students
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled “**An Autoencoded Transductive Support Vector Machine for Noisy Image Classification**” being submitted by **Khushi Agrawal (17075060)** and **Janhavi Gupta (17075061)**, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT (BHU) Varanasi
Date: 20/11/2020

Prof. K.K. Shukla
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Acknowledgments

We would like to express our sincere gratitude to our supervisor Prof. K.K. Shukla for providing all the necessary support and guidance for the completion of this project.

Place: IIT (BHU) Varanasi

Date: 20/11/2020

Khushi Agrawal, Janhavi Gupta

Abstract

Almost all of the real-world datasets contain noise to some extent. These distortions in the images may lead to inaccurate and unexpected results and hence developing algorithms robust to these noises helps greatly in improving the overall performance of the model. Feature selection and extraction is one of the common ways to improve the accuracy of the model, and autoencoders are one of the most common mechanisms for extracting the useful features. However, these autoencoders run the risk of learning the identity function, providing no useful extraction and making its purpose useless. Denoising Autoencoders (DAE) were then introduced as an extension of the autoencoders, which purposely added noise to the images for effective feature extraction followed by reconstruction of the decoded images.

This project focuses on developing a robust algorithm for classification of noisy images by first feeding the images through a denoising autoencoder. The reconstructed images obtained are then fed to a conventional Transductive Support Vector Machine, which is a semi-supervised machine learning model for classification. The results obtained were then analyzed with those obtained from conventional SVM, TSVM (without DAE) and robust TSVM (developed in the previous semester), which provided a base for establishing the effectiveness of DAE along with semi-supervised TSVM Model.

Contents

List of Figures	x
List of Tables	xi
1 Introduction	1
1.1 Overview	1
1.2 Motivation of the Research Work	2
1.3 Recapitulation of previous semester	2
1.3.1 Transductive support vector machines (TSVM)	2
1.3.2 Loss Functions	3
1.3.3 Optimisation methods	3
1.3.4 Performance of the model	4
1.4 Organisation of the Report	5
2 Literature Study	6
2.1 Denoising Autoencoders	6
2.2 Different models using TSVM	7
3 Autoencoded TSVM: A hybrid of DAE and TSVM	9
3.1 Preprocessing of input data	9
3.2 Feature extraction from preprocessed data	10
3.3 Multi-class classification using TSVM	11

CONTENTS

4	Experiments and Results	13
5	Conclusions and Discussion	17
	References	18
	Appendix	20

List of Figures

2.1	A Denoising Autoencoder	6
2.2	Hinge Loss (left) and Ramp Loss (right)	8
3.1	[1] Input samples for DAE from MNIST, [2] MNIST samples after addition of noise, [3] Reconstructed images from DAE	10
3.2	VGG19 Architecture	11
3.3	Plot for Principal Component Analysis	12
4.1	Flowchart of Autoencoded TSVM	14
4.2	Plot of Loss vs Epochs for DAE	15

List of Tables

1.1	Results obtained on the Australian dataset for SVM and TSVM using Dual Form and SGD	4
4.1	Reconstruction of image using DAE on noisy data	15
4.2	Results obtained on Wine dataset using multi-class TSVM	16

Chapter 1

Introduction

1.1 Overview

The focus is to use semi-supervised learning as it acts as an intermediate between supervised learning and unsupervised learning. In supervised learning, all the data samples have to be labelled and this is generally not the case when the real world data is dealt with. On the other hand, in case of unsupervised learning, each data sample is unlabelled and detecting accuracy of such models becomes difficult. Semi-supervised learning makes use of labelled samples as well as unlabelled samples for training the model and predicts the labels for unlabelled data. Most of the data available is not pure and contains some or the other form of noise. So, the algorithms need to be trained in such a way that they perform well on the datasets on which they are trained and are also robust to noise. Our main work is to study Transductive support vector machines for semi-supervised learning and denoising auto-encoders for making TSVMs robust to noise.

1.2 Motivation of the Research Work

TSVMs [3] make use of both labelled as well as unlabelled data for classifying samples. Several loss functions such as Hinge Loss [4], Ramp loss [9], Rescaled Hinge Loss [12], etc. can be used in the optimization problem. Labelled and unlabelled samples make use of different loss functions for better results as is clear from the results in [3]. Since it is difficult to label all the training samples particularly in case of large datasets [2], whereas not labelling any of the samples leads to lower accuracy, TSVM provided an effective trade-off between the two and motivated us to explore more in the area.

Autoencoders [14] have been used for efficient data codings since long in the field of Deep Learning. Using the extended form of it as a denoising autoencoder (DAE) [8] for feature extraction provided a fruitful method for noise removal using the network of encoder-decoder pipeline. PCA [10] has been proven to be highly effective for feature extraction and hence it was considered to apply PCA to the decoded images for further refinement of extracted features. As TSVM handles the binary classification problem, the one-against-rest heuristic method is found suitable to divide multi-class classification problem into several binary classification problems and thus apply TSVM on it.

1.3 Recapitulation of previous semester

1.3.1 Transductive support vector machines (TSVM)

TSVM defines a hyperplane to divide the labelled samples with largest margin possible and also classifies unlabelled samples. It makes sure that the unlabelled samples should also be at largest margin possible from the hyperplane. The equation for the same is given in (1.1).

1.3. Recapitulation of previous semester

$$\begin{aligned} \arg \min \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^L \epsilon_i + C^* \sum_{i=L+1}^{L+U} \epsilon_i \\ \text{s.t.} \quad & y_i(w^t x_i + b) \geq 1 - \epsilon_i, \quad i = 1, \dots, L, \\ & |w^t x_i + b| \geq 1 - \epsilon_i, \quad i = L + 1, \dots, L + U. \end{aligned} \tag{1.1}$$

1.3.2 Loss Functions

In the previous semester, we studied different loss functions like hinge loss, ramp loss, rescaled hinge loss functions. Out of these functions, the main focus was on hinge loss function which is used in conventional TSVM and ramp loss function which added robustness to TSVM. Hinge loss function was replaced with ramp loss function as it is not robust to outliers. Hinge loss is an unbounded loss function and the outliers which lie far from the margin start contributing prominently in the loss function, whereas Ramp loss is a bounded, non-convex function which is more robust to outliers.

Ramp Loss function is defined by a linear combination of Hinge Loss given in (1.2).

$$R_s(t) = H_1(t) - H_s(t) \tag{1.2}$$

1.3.3 Optimisation methods

We dealt with two methods for solving the optimisation problem stated in (1.1), dual method and stochastic gradient method. The optimization problem under consideration was non-convex and hence was solved using the Convex-Concave Procedure (CCCP) [13]. In this procedure, the entire cost function is decomposed into two parts and is written as a sum of a convex and a concave part.

Consider a cost function $J(\theta)$ which can be written as given in (1.3).

$$J(\theta) = J_{\text{concave}}(\theta) + J_{\text{convex}}(\theta) \tag{1.3}$$

The minimisation of $J(\theta)$ can be achieved by employing CCCP procedure (see (1.4)).

$$\min arg (J_{convex}(\theta) + J'_{concave}(\theta) * (\theta)) \quad (1.4)$$

Dual method made use of Lagrangian's multiplier for solving the optimisation problem and it's implemented using the **cvxopt** [6], python's library function. The drawback of this approach is that it's execution time is $O(n^2)$, where n is number of samples, thereby making it unsuitable for large-scale datasets. In case of stochastic gradient method, the weights are updated in every iteration on the basis of a random chosen sample and it scales better with increasing number of samples so it is better suited for large datasets. We implemented both these methods in the previous semester, and analyzed the performance of both methods with conventional SVM, on several datasets, with varying percentage of noises to check for the robustness of the proposed method.

1.3.4 Performance of the model

The model implemented in the previous semester showed significant improvement of accuracy for the robust stochastic gradient method, as compared to the dual form and conventional SVM, with increasing percentage of noises (see table 1.1). Noise was added incrementally by switching the labels of samples in the training data set. The model was run on several datasets, and the results obtained on one such dataset is shown as under:

Table 1.1 Results obtained on the Australian dataset for SVM and TSVM using Dual Form and SGD

Outliers	0%	10%	20%	30%	40%	50%
SVM	82.41	79.31	76.21	79.66	78.62	78.62
TSVM (dual form)	63.45	64.14	59.66	44.14	63.45	63.79
Robust TSVM (sgd)	84.14	85.17	80.35	84.48	81.72	85.52

1.4 Organisation of the Report

The report discusses the literature studied in the previous semester which comprised of different loss functions used for TSVMs. Further, it contains the proposed model for the classification of noisy data which is a combination of denoising auto-encoders and conventional TSVMs. It also contains the experimentation and results on different datasets proving the robustness of the model.

Chapter 2

Literature Study

2.1 Denoising Autoencoders

Autoencoders are used for effective compression followed by decompression of images which helps in data dimensionality reduction by ignoring the noise present in images. The term 'noise' here refers to any distortions than what is expected in any data sample. These noises affect the performance and hence many algorithms have been developed for efficacious pre-processing and noise removal. Conventional autoencoders developed suffered the risk of overfitting and were not robust to noisy images, and hence Denoising Autoencoders (DAE) were developed as an extension of the conventional autoencoders which provided more robust filters and significantly reduced the chances of overfitting by purposefully distorting the input images, and then training the model to recover the original, unperturbed images as close to the input as possible.

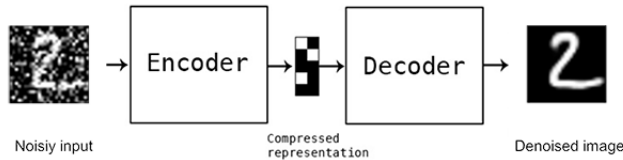


Figure 2.1 A Denoising Autoencoder

Denoising autoencoders are a stochastic version of conventional autoencoders which

2.2. Different models using TSVM

minimize the risk of the model learning the identity function, and helps extract useful features. It consists of an encoder pipeline followed by a decoder pipeline. The encoder pipeline compresses the image, which is then decompressed by the decoder pipeline, thereby resulting in the final reconstructed images as shown in Fig 2.1. Both of these pipelines consist of several hidden layers which help in effective feature extraction. Noise is added on purpose to provide robustness to the model and help improve its generic performance. This is specially useful in cases where the number of hidden layers are large, and the model runs a high risk of learning the identity function. The encoder part of the denoising autoencoder downsamples the input image, removing the noise present in the data. The decoder then upsamples this image to construct the decoded image, free of noise. The stated pipeline works on the simple mechanism of minimizing the difference between the obtained reconstructed image and the original image.

2.2 Different models using TSVM

As discussed earlier, TSVM is a semi-supervised machine learning model which makes use of labelled as well as unlabelled samples for training and classification. The conventional TSVM mentioned in [3] makes use of the Hinge Loss function (Fig 2.2a) and stochastic gradient method for solving the optimization problem. However, further observation showed that Ramp Loss being bounded (see Fig 2.2b), provides more robustness to the model as compared to the unbounded Hinge Loss function (see Fig 2.2a), and hence Ramp Loss was used in the robust TSVM approach as an improvement to the conventional TSVM which uses Hinge Loss.

In the proposed approach, the reconstructed images obtained from Denoising Autoencoder were then fed to the conventional TSVM for classification, marking only a few samples as labelled and the others as unlabelled.

The experiments in the previous semester were carried out on datasets with binary classifications. This was further extended this semester to multi-class classification

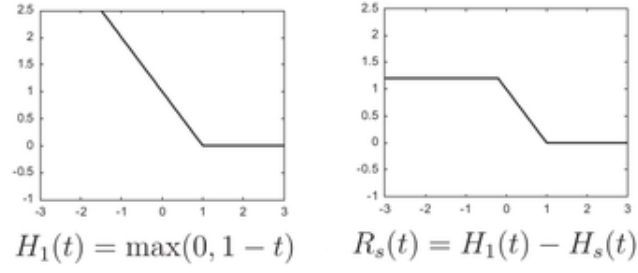


Figure 2.2 Hinge Loss (left) and Ramp Loss (right)

using TSVM with one against rest approach. For each class, the weights were calculated using TSVM by treating samples of remaining classes as negative. Finally, the sample is assigned the class for which it has maximum score that is the class having score closest to 1.

Chapter 3

Autoencoded TSVM: A hybrid of DAE and TSVM

The following sections show a methodological approach to our proposed model and its various components along with implementation details.

3.1 Preprocessing of input data

The **Denoising Autoencoder** used in the project included an encoder pipeline having two convolutional layers and two max pooling layers for downsampling. The compressed image obtained is then fed to the decoder pipeline which has two convolutional layers and two convolutional transpose layers which are used for upsampling the compressed images. The activation function used was leaky ReLU (Rectified Linear Unit) [11] instead of ReLU [1] in order to allow backpropagation of negative gradients while updating weights. Cross entropy loss function [15], being based on probabilities of prediction, is greatly effective in minimizing the difference between actual and predicted values, and hence is used in the proposed implementation. Also, Adam Optimizer [5] is used while training as it is a combination of AdaGrad (suitable for sparse gradients) and RMSProp (suitable for noisy datasets) and hence is suitable for optimization of

noisy datasets.

The images below clearly reflect the functioning of the implemented Denoising Autoencoder. First, the input samples are shown, followed by the samples after addition of noise (taking a noise factor of 0.5). Lastly, the reconstructed images obtained after feeding the noisy images to the Encoder-Decoder pipeline are shown in Fig 3.1.

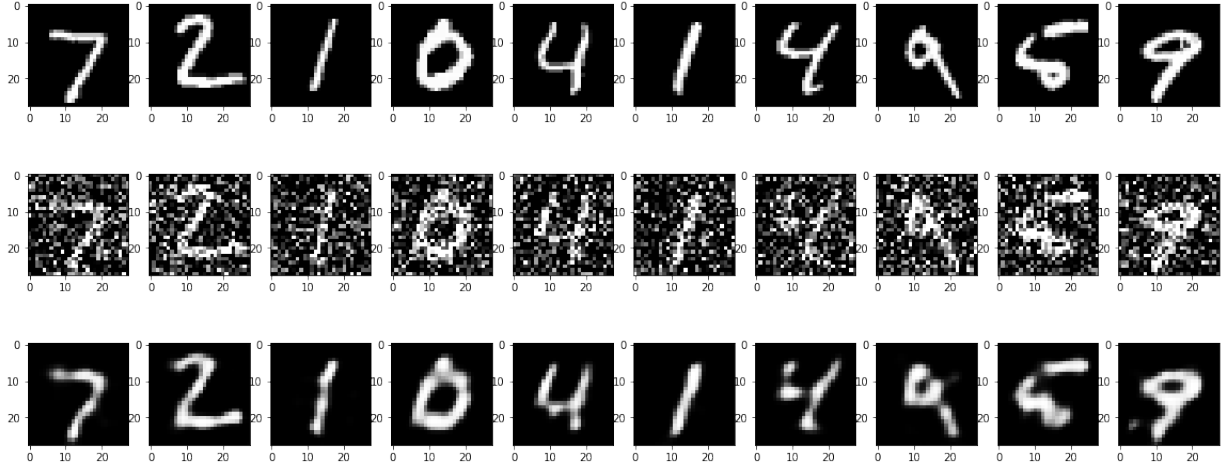


Figure 3.1 [1] Input samples for DAE from MNIST, [2] MNIST samples after addition of noise, [3] Reconstructed images from DAE

3.2 Feature extraction from preprocessed data

Since the number of features for real-world large scale datasets can be extremely large, feature extraction is essential in order to ensure that the model learns only the important characteristics. Keras provides a set of pre-trained deep learning models for extracting features. Our project uses the **VGG19 model** [7], pretrained on the ImageNet dataset. This model has two parts: from input layer to last max-pooling layer known as feature extraction part, and a classification part which includes the dense layers (see Fig 3.2). The variable *include-top* takes on a boolean value indicating whether the classification part is to be included, which is set as false for our use case. While extracting features, *preprocess-input* is imported for appropriate scaling of pixel values,

3.3. Multi-class classification using TSVM

followed by feature prediction using the *predict()* function of the VGG19 model loaded earlier.

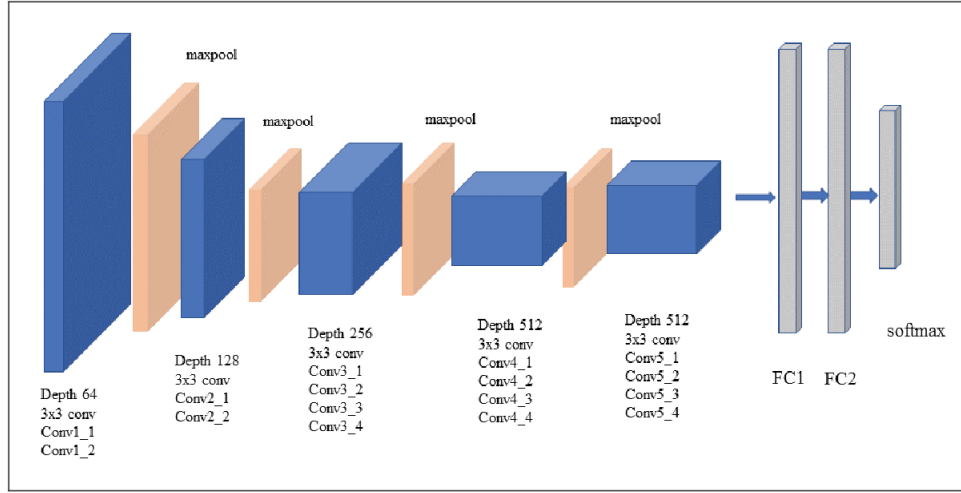


Fig. 3. VGG-19 network architecture

Figure 3.2 VGG19 Architecture

After extracting the features, further refinement is done by applying Principal Component Analysis (PCA) on the extracted features. PCA is an unsupervised, linear algorithm for dimensionality reduction which is based on the idea of correlation among the features. It attempts to combine the highly correlated features, thereby reducing the number of dimensions, while trying to preserve their originality as much as possible, i.e., minimizing the reconstruction error. The new components thus obtained are referred to as principal components and these are the ones fed to the semi-supervised TSVM model for classification. Our project uses the inbuilt PCA model provided by scikit-learn in python, and the plot of variance v/s number of components obtained is shown in the Fig 3.3.

3.3 Multi-class classification using TSVM

TSVMs were used in the previous semester for binary classification using labelled and unlabelled samples. For datasets involving multiple classes, the classification is performed using **one against rest approach**. For each class, the dataset is divided

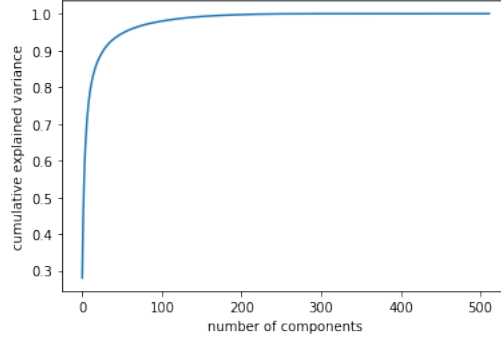


Figure 3.3 Plot for Principal Component Analysis

into two-parts: the positive samples for that class and the negative samples for all the remaining classes. Now, the weights are calculated using conventional TSVM for the same class and are stored. For each sample, prediction is performed based on the scores. Scores are calculated using weights for the corresponding class and sample is classified as belonging to the class having maximum score that is, the score closest to one. The final accuracy is then estimated based on the number of correctly predicted samples (correct predictions/total predictions).

Chapter 4

Experiments and Results

Initially, the noisy images were passed through the encoder pipeline for downsampling (compression) followed by the decoder pipeline of DAE which resulted in decoded images with noise removed. This was followed by feature extraction using the pre-trained Keras Model VGG19 and applying Principal Component Analysis (PCA) for further refinement.

Then, the data was passed through TSVM Model for classification. Even though DAE has the dense layers which can be used for classification, it cannot be used for our purposes because the classification through DAE is supervised and requires training data with marked labels, whereas TSVM only requires a small number of labelled samples and makes use of the best of both labelled and unlabelled samples for classification.

For training the model for TSVM, the datasets are divided into a part of labelled samples and unlabelled samples randomly by making sure that some of the samples of each class are present in such division. Then, TSVM for multi-class classification is applied to the prepared dataset. This process is repeated for some iterations and the final result is reported by calculating the average of accuracies received in all iterations.

The flowchart in Fig 4.1 shows the complete sequence of operations for classifying the noisy image samples.

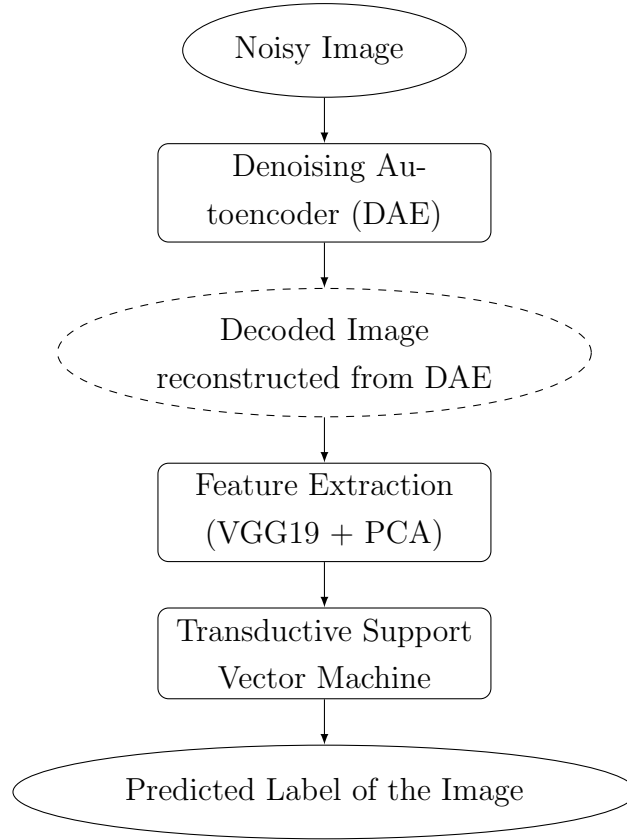


Figure 4.1 Flowchart of Autoencoded TSVM

The DAE model was trained using 60k training samples and tested on 10K test samples of MNIST dataset, using 25 epochs to minimize the loss and make it fit for reconstruction of images.

The losses were then plotted to showcase the training of DAE as losses were observed to be decreasing with each epoch. The loss vs epoch curve obtained on running 25 epochs on MNIST dataset can be seen in Fig 4.3.

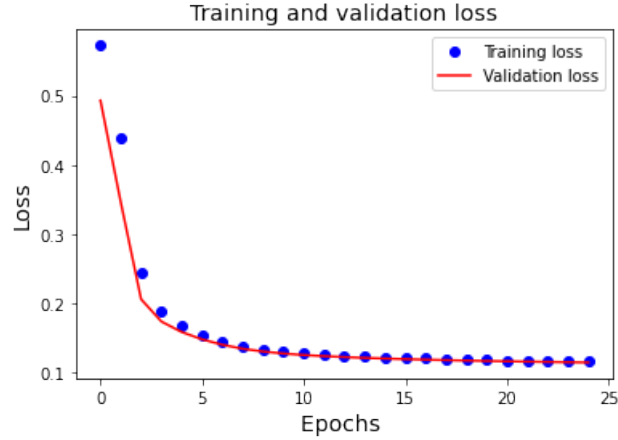
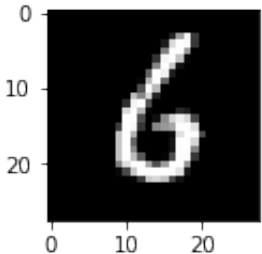
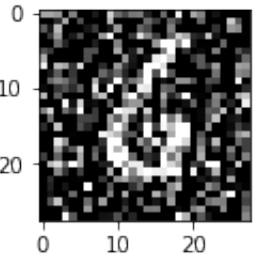
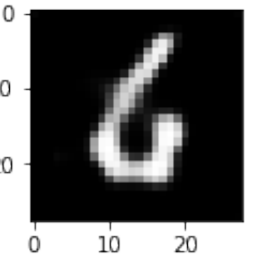


Figure 4.2 Plot of Loss vs Epochs for DAE

A comparative analysis of the images obtained after reconstruction is shown as below:

Table 4.1 Reconstruction of image using DAE on noisy data

Original Images	Noisy Image	Reconstruction of Noisy Image
		

First, Gaussian noise is added to the image to get the noisy image as shown in column 2, Table 4.1 above. The noise factor is a hyperparameter which can be changed to test the effectiveness of the model on varying levels of noises. The figure shown above was distorted with a noise factor of 0.5. These noisy images are then passed through

the encoder-decoder pipeline for training the DAE to obtain a reconstructed image as close to the original as possible, as shown in the third column of Table 4.1.

The multi-class classification using TSVM was made to run on Wine dataset, which has around 200 samples with 3 classes. For training, the dataset was divided into 10 parts, each part having some randomly taken labelled samples from each class and rest were marked as unlabelled. The final accuracy of the model is taken as the average of accuracies obtained in each part by using multi-class TSVM. The accuracy obtained on obtained on Wine using conventional and Ramp TSVM is given in Table 4.2.

Table 4.2 Results obtained on Wine dataset using multi-class TSVM

Conventional TSVM	68.25
Ramp TSVM	68.75

Hence, DAE and multi-class TSVM performed well individually. So, we tried to combine these two approaches to introduce a robust model which can classify noisy images by reconstructing a decoded image with noise removed, followed by its classification using conventional TSVM.

Chapter 5

Conclusions and Discussion

The approach proposed in this project work provided a robust multi-class classification through semi-supervised learning. Achieving robustness is of prime importance for developing algorithms which can have real-world applications as almost every real-world dataset is prone to noise and distortions. We proposed to achieve this robustness through Denoising Autoencoders which removes noise from the input samples and reconstructs the decoded image after noise removal. Since DAE follows the approach of supervised learning for classification, we replaced it with TSVM, a semi-supervised model which was explored in the previous semester. The combination of the two, resulted in a model which was robust to noise, and also followed the semi-supervised approach, thus achieving our objective.

In the previous semester, robustness was achieved by modifying the conventional TSVM with a more robust loss function (ramp loss). However, using DAE for achieving robustness coupled with conventional TSVM performed better, thus showing its effectiveness for real-world datasets.

In the future endeavours, we aim to further robustify the model, by exploring other loss functions which may be employed for achieving better results.

References

- [1] Abien Fred Agarap. “Deep Learning using Rectified Linear Units (ReLU)”. In: (2018). *Link*.
- [2] Andri Ashfahani et al. “DEV DAN: Deep evolving denoising autoencoder”. In: (2019). *Link*.
- [3] Hakan Cevikalp and Vojtech Franc. “Large-scale robust transductive support vector machines”. In: (2017). *Link*.
- [4] Claudio Gentile and Manfred K. Warmuth. “Linear Hinge Loss and Average Margin”. In: (1998). *Link*.
- [5] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: (2014). *Link*.
- [6] *Quadratic Programming with Python and CVXOPT*. *Link*.
- [7] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for large-scale Image Recognition”. In: (2014). *Link*.
- [8] Pascal Vincent et al. “Extracting and composing robust features with denoising autoencoders”. In: (2008). *Link*.
- [9] Lei Wang, Huading Jia, and Jie Li. “Training robust support vector machine with smooth Ramp loss in the primal space”. In: (2007). *Link*.
- [10] Binjie Xiao. “Principal component analysis for feature extraction of image sequence”. In: (2010). *Link*.

References

- [11] Bing Xu et al. “Empirical Evaluation of Rectified Activations in Convolution Network”. In: (2015). *Link*.
- [12] Guibiao Xu et al. “Robust support vector machines based on the rescaled hinge loss function”. In: (2017). *Link*.
- [13] Alan Yuille and Anand Rangarajan. “The Concave-Convex Procedure (CCCP)”. In: (2003). *Link*.
- [14] Junhai Zhai et al. “Autoencoder and Its Various Variants”. In: (2018). *Link*.
- [15] Zhilu Zhang and Mert R. Sabuncu. “Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels”. In: (2018). *Link*.

Appendix

The following code snippet shows the encoder pipeline having 2 convolutional layers and 2 max pooling layers for upsampling the input data sample.

```
encoder = Sequential([
    Conv2D(
        filters=32,
        kernel_size=(3,3),
        strides=(1,1),
        padding='SAME',
        use_bias=True,
        activation=lrelu,
        name='conv1'
    ),
    MaxPooling2D(
        pool_size=(2,2),
        strides=(2,2),
        name='pool1'
    ),
    Conv2D(
        filters=32,
        kernel_size=(3,3),
        strides=(1,1),
        padding='SAME',
        use_bias=True,
        activation=lrelu,
        name='conv2'
    ),
    MaxPooling2D(
        pool_size=(2,2),
        strides=(2,2),
        name='encoding'
    )
])
```

The following code snippet shows the decoder pipeline for downsampling the encoded image obtained from the encoder pipeline. This has two convolutional and two

convolutional transpose layers for downsampling.

```
decoder = Sequential([
    Conv2D(
        filters=32,
        kernel_size=(3,3),
        strides=(1,1),
        name='conv3',
        padding='SAME',
        use_bias=True,
        activation=lrelu
    ),
    Conv2DTranspose(
        filters=32,
        kernel_size=3,
        padding='same',
        strides=2,
        name='upsample1'
    ),
    Conv2DTranspose(
        filters=32,
        kernel_size=3,
        padding='same',
        strides=2,
        name='upsample2'
    ),
    Conv2D(
        filters=1,
        kernel_size=(3,3),
        strides=(1,1),
        name='logits',
        padding='SAME',
        use_bias=True
    )
])
```

Once decoded images are obtained after noise removal from DAE, feature extraction is applied using VGG19 and PCA, as shown in the following code snippet:

```
# Pre-trained VGG19 Keras Model
base_model = VGG19(weights='imagenet', include_top=False, input_shape
    =(32,32,3))
x = preprocess_input(x)
# feature extraction using VGG19 as base model
fea = base_model.predict(x)
# Extracting components through Principle component analysis
pca = PCA().fit(fea)
pca = PCA(n_components=22)
```

The classification on decoded images is done using conventional TSVM which makes use of Hinge Loss function. This is described in the following code snippet:


```

# Non-convex optimisation using CCCP procedure
if ii <= L and score < 1:
    gw = (-C1*yi*xi/L)
    gb = (-C1*yi/L)
elif ii > L and score < 1:
    gw = ((-C2*yi*xi)+(beta[ii-L]*yi*xi))/(2*U)
    gb = ((-C2*yi)+(beta[ii-L]*yi))/(2*U)
elif ii > L and score >= 1:
    gw = (beta[ii-L]*yi*xi)/(2*U)
    gb = (beta[ii-L]*yi)/(2*U)
else:
    gw = np.zeros(XX.shape[0])
    gb = 0

```

Multi-class classification in TSVM is done using the one-against-rest approach, treating one class as positive, and all other negative, followed by label prediction. This is shown as described below:

```

# treating one class positive and rest of the classes negative
yone = (y==i)+(y!=i)*-1
hyperplane = train_linear_transductive_svm_sg(X,yone,C1,C2,w0,b0,
    alpha)
# prediction of sample with class having score closest to 1.
y_pred = np.argmax(scores_all,axis=0)

```