SAT-DL: Programming Assignment: 3

D. Mohit Varsha 2019H1030026G

April 2020

TASK 1

Construct good Denoising Autoencoder using the MNIST Dataset

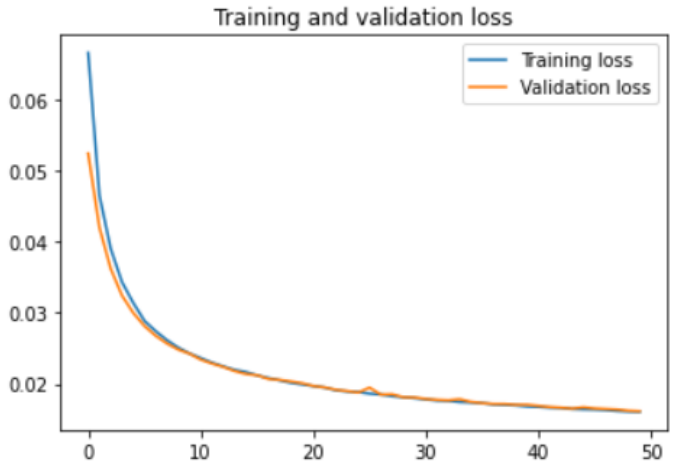
Working

Dataset: The data set is MNIST, it has samples for Hand Written Digits, in the form of grey scale images that occupy 28x28 pixels. There are a total of 60000 training samples and 10000 testing samples. For the model we will use 48000 training samples and 12000 validation samples.

Model: The model has 3 sections. They are the encoding part, the latent representation and the decoding part. For encoding part we use 3 convolution layers and 3 max pooling layers then we have the hidden or latent representation. For decoding part, since the output size is same as input we use 3 convolution layers and to bring back the size 3 up sampling layers. Finally we get the output of the model.

Training: Before we feed the training dataset to the model we introduce some noise to the images using random function and then we scale the values between a range 0 and 1. Then we train the model with a batch size of 1000 and epochs of 50.

Results: Given below is the training and validation loss of the model.



We also have the noisy images as input and de-noised images as the output.

Given below are few examples.

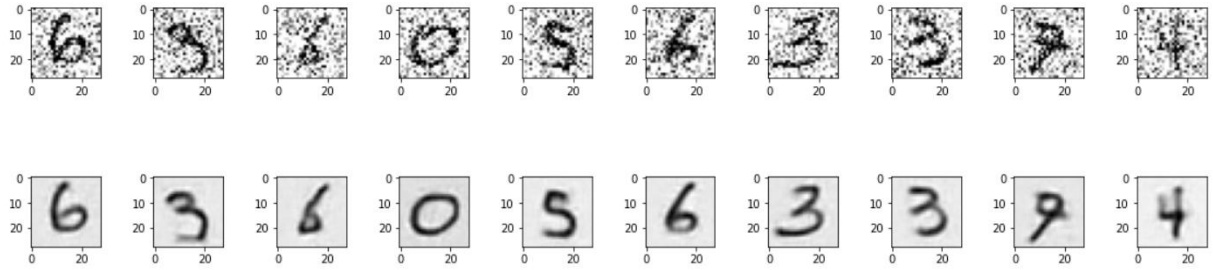


Fig 2(): Noisy Images as input on top and Noise Free Images as output on bottom

TASK 2

Construct a classification model using Fashion-MNIST Dataset

Working

Dataset: The data set is Fashion-MNIST, it has samples for fashion wear, in the form of grey scale images that occupy 28x28 pixels. There are a total of 60000 training samples and 10000 testing samples. For the model we will use 48000 training samples, 12000 validation samples and 10000 testing samples.

Model: The model is based on Convolutional Neural Network (CNN) as they are widely used for image classification. We have 2 sets of Conv2D and MaxPool2D layers followed by flatten layer then a dense layer and for output we have a dense layer of output size 10, because we have 10 different class labels. We use dropouts for regularization. Also care is taken that one hot encoding is done for the labels.

Training: Before we feed the training dataset to the model we scale the values between a range 0 and 1. Then we train the model with a batch size of 1000 and epochs of 50. We also use a validation set.

Testing: We then test the above model with a separate test dataset different from training and validation sets, which has 10000 samples.

Results: Following are various graphs that depict the output achieved by the above model

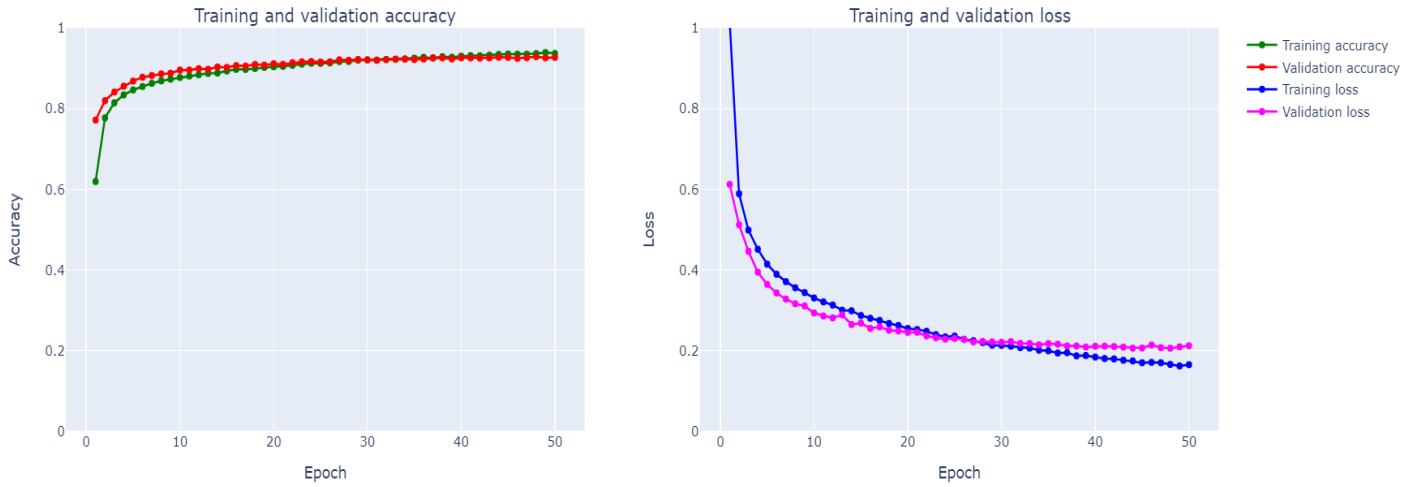


Fig 3(): Towards left we have training and validation accuracies. In the image right side we have training and validation loss.

We have Test loss and Test accuracy of the model as:

Test loss for model 2: 0.23717711246013642

Test accuracy for model 2: 0.9160000085830688

Fig 4(): The images in green towards left explain few samples that were classified correctly by our model. The images in red towards the right explain few samples that were classified incorrectly by our model.

TASK 3

Use Model in Task 1 as a pre trained model to learn the new classification problem given in the dataset of Task 2

Working

The idea is to use the autoencoder that is trained in Task 1 and use data set of Task 2 to retrain the model. This happens in the following sequence. First autoencoder is trained on MNIST dataset. Then the weights of the model are saved. Then we construct model for Task 3, as, taking the Encode part of autoencoder from Task 1 then to this add flatten layer followed by dense layer then the output layer which again is a dense layer with 10 outputs since we have 10 labels to classify. Then train the model again but with Fashion-MNIST dataset this time.

Training: Before we feed the training dataset to the model we scale the values between a range 0 and 1. The weights previously saved are loaded into the new model. Then we re-train the model with a batch size of 1000 and epochs of 50. Also care is taken that one hot encoding is done for the labels.

Testing: We then test the above model with a separate test dataset different from training and validation sets, which has 10000 samples. This Test Set is same as in Task 2.

Results: Following are various graphs that depict the output achieved by the above model.

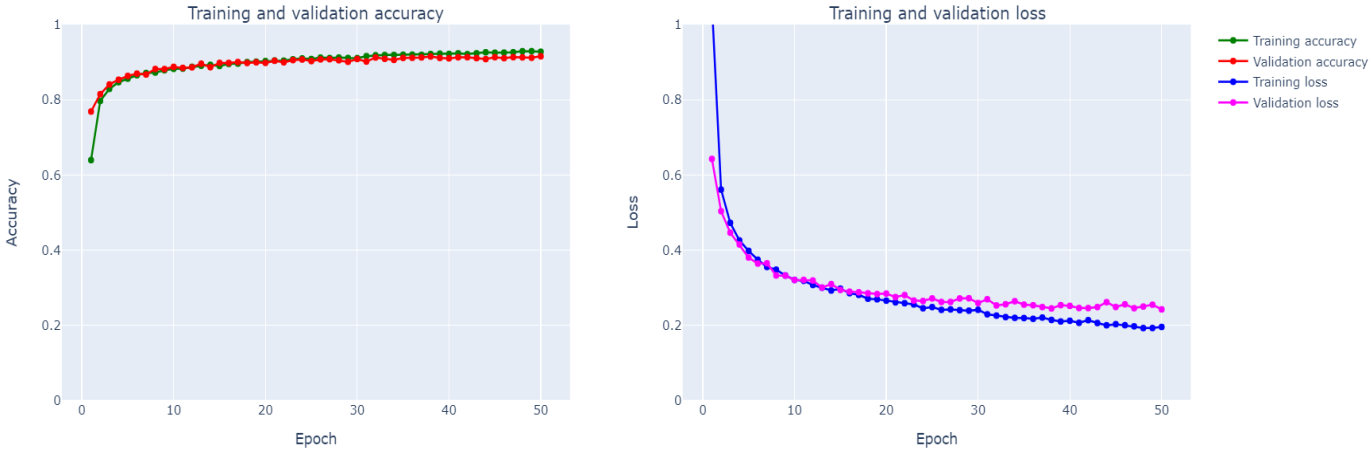


Fig 5(): The model is a fine tuned model based on autoencoder of Task 1. Towards left we have training and validation accuracies. In the image right side we have training and validation loss.

We also have Test loss and Test accuracy of the model as:

Test loss for model 3: 0.26341479016542435
Test accuracy for model 3: 0.9085000157356262

Fig 6(): The images in green towards left explain few samples that were classified correctly by our model. The images in red towards the right explain few samples that were classified incorrectly by our model

TASK 4

Compare Models in Task 2 and Task 3

First we compare the accuracies and losses in case of both models.

	2. Classification Model	3. Fine tuned Autoencoder
Test Accuracy	0.9160000085830688	0.9085000157356262
Test Loss	0.23717711246013642	0.26341479016542435
Train Accuracy last epoch	0.9400	0.9336
Train Loss last epoch	0.1604	0.1808
Validation Accuracy last epoch	0.9247	0.9150
Validation Loss last epoch	0.2092	0.2484

Next we compare precision, recall and F-Scores of both the models.

PRECISION, RECALL AND F-SCOREs

	Classification Model				Vs.	Fine Tuned Autoencoder Model			
label	precision	recall	fscore	support		precision	recall	fscore	support
0	0.8581	0.8830	0.8704	1000		0.8111	0.8930	0.8501	1000
1	0.9939	0.9790	0.9864	1000		0.9929	0.9810	0.9869	1000
2	0.8432	0.8980	0.8697	1000		0.8846	0.8200	0.8511	1000
3	0.9038	0.9400	0.9216	1000		0.9328	0.9020	0.9171	1000
4	0.8777	0.8470	0.8621	1000		0.8140	0.8840	0.8476	1000
5	0.9860	0.9830	0.9845	1000		0.9840	0.9830	0.9835	1000
6	0.7813	0.7180	0.7483	1000		0.7655	0.7180	0.7410	1000
7	0.9613	0.9700	0.9657	1000		0.9630	0.9630	0.9630	1000
8	0.9809	0.9740	0.9774	1000		0.9848	0.9700	0.9773	1000
9	0.9699	0.9680	0.9690	1000		0.9623	0.9710	0.9667	1000
Avg	0.9156	0.9160	0.9155			0.9095	0.9085	0.9084	

- Other variations were tried for fine tuning the auto encoder that include,
  1. Using full auto encoder with encode and decode then attaching the classifier built in task 2 to the end of it. Transferring weights from auto encoder to this new model, freezing those layers and re-training the other layers.

     This model gave the results of

     loss: 0.47711253628730776 and accuracy: 0.8266000151634216
  2. Same model as above but instead of freezing the layers, we don't freeze any layers and re-train all the layers.

     This model gave the results of

     loss: 0.683859825152321 and accuracy: 0.8195999765396118
  3. Another model like in 1 but instead we add noise to data before sending it to the model. This model gave the results of

     loss: 0.47080628552436826 and accuracy: 0.833299994468689