The Impact of The Architecture Details on The Performance of a Siamese Neural Network

Zahir Bilal

July 2021

Abstract

One-shot or few-shots learning is an approach that aims to classify objects based on few amount of data. In this project we explore the approach of few-shots learning by utilizing a **Siamese Neural Network (SNN)**. In order to evaluate the impact of different architectures of SNN on its performance, we implemented two basic models with several variations: **Siamese Convolutional Neural Network (SCNN)** and **Siamese multi-layer perceptron (SMLP)**. Finally we compared the performance of the different structures of the SNN based on a ranking metric.

1 Introduction

The **Siamese Neural Network (SNN)** consists of two identical neural networks with identical structures (number and size of layers, number of features, etc.). The network's weights are joined by an energy function, which computes a pre-defined metric between the highest level feature representation (embeddings) at each side of the network. The similarity between two inputs is represented by the distance between the embeddings of the two inputs' features.

A SNN can be modelled using any of the conventional neural networks architectures. The determination of the most suitable neural networks architectures with their optimal hyperparameters in a SNN is usually dependant on the desired application and the the dataset. In overall, researches about the different architectures of the Siamese neural network have not attracted the interest of the machine learning community. This research is an attempt to pave the way to propose approaches to deal with the question: How would changes at the architectural level affect the performance of the SNN? To fulfill this purpose, we investigated the performance of an IRIS recognition Siamese Neural Network. The architecture details of the network were: The structure of the SNN, number of layers, size of the kernel of the CNN layers, size of the final embedding layer. Finally, we analyzed the performance of the different implemented networks based on their learning behavior and accuracy.

2 Related work

There are few significant research concerns the one-shot learning as in Koch et al (2014) [1], where they designed a SCNN to perform one-shot classification task for the different alphabets in the Omniglot dataset[1]. The mentioned approach has nearly outperformed other state-of-art learning models by scoring 92% in 20-way one-shot learning. In 2021, Angelovska et. al [2] has utilized two different SNN to design a complementary products recommender system. The two models were: A **Convolutional Nerual Network (CNN)** and a Long Short-Term Memory (LSTM). The performance of a SNN on a small dataset were also investigated by Figueroa-Mata et. al [3], where they

[1]http://iris.di.ubi.pt/ubiris2.html

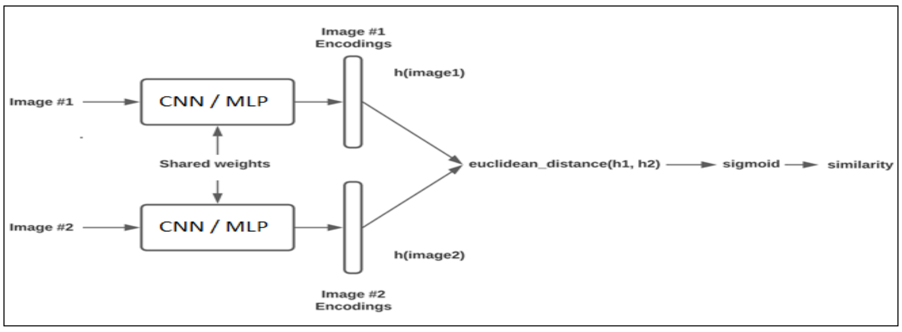

Figure 1: Structure of a Siamese Neural Network.

attempted to identify different Image-Based Plant Species.in their research they show how a CSNN has achieved better results than a CNN at almost each tested dataset size. Several variant of architectures of the Siamese recurrent neural network were studies by Ranasinghe et. al [4], where they evaluated the performance of each network on semantic similarity between texts.

3 Methodology

In this research, we implemented a SNN for an IRIS recogonition system. a CNN and a multi-layer perceptron structure were implemented.several variants of these structures were introduced to evaluate the effect of the architictural changes on the performance of the SNN.

Dataset pre-processing: The dataset MMUII[2] was used for training and multiple-ways testing of the models. The dataset contains images for 100 different subjects, each subject has in total ten eye images. For training the model we used 10000 pair of images which were picked randomly from 70 different subjects. The training set was splitted into training and validation sets by 80:20 ration randomly. We tested the model then on 1000 images picked randomly from the other 30 subjects, the model has not seen these example before. The images itself poses a challenge for our models, as it contains parts of the eye other than the IRIS, such as glasses, eyebrows and eyelashes. These factors were considered noise factors and resulted in making the classification task more challenging. For verification, the different SNNs implemented in this experiment were tested even on the Ubiris-II dataset[3] in which the images were actually captured on non-constrained conditions (at-a-distance, on-the-move and on the visible wavelength), with more realistic noise factors.

Training: To train the SNN, we first generate the inputs and define the ground truth label of the model. We randomly select a pair of images as inputs into the network, if the pair belongs to the same subject, then we have a similarity of 1, otherwise 0. We ensure that the training set is balanced for both types. image size were selected to be 105x105. Models were trained for 50 epochs, with a learning rate of 0.0005 and Adam optimizer. The Binary cross entropy loss function was chosen and the similarity between images embeddings were measured using euclidean distance. The training and experiment were executed through Google Colab, with a Tesla K80 GPU. We used libraries including Numpy, Matplotlib, and PyTorch.

Testing: The evaluation of a network on its performance in one-shot learning can be done via an n-way one shot learning evaluation metrics, where we

[2]http://andyzeng.github.io/irisrecognition
[3]http://iris.di.ubi.pt/ubiris2.html

find random n images representing n categories and one main image that belongs to one of the n categories. For our SNN, we computed the similarity of the main images against all n images, and the pair with the highest similarity means the main image belongs to the class. for more information about the experiment design and implementation please refer to[4]. The implementation of the algorithm was inspired by the approaches in the following sources [5] [6].

Networks Structure: To analyze the impact of the architecture on the performance of the SNN, we implemented the network presented in [1] and denoted it with CNN_1, in [1] this network scored 92% when evaluated by 20-ways one-shot learning on the Omniglot dataset. Having a more challenging dataset and more complicated task, we expect the accuracy of CNN_1 to decrease significantly when evaluated by 8-ways and 16-ways on our dataset. Afterwards several variations were introduced to its structure which included changing the number of the convolutional layers, the size and numbers of the kernel as well as the size of the final embeddings. The customized networks were denoted by CNN_2 - CNN_6 in Table 1. **conv** indicates the convolutional layers of the networks. The details of each layers were indexed as following: (number of input channels, number of output channels, filter size). **fcl** refers to the fully connected layers and the sizes of the corresponding layers and were also indexed as following (size of fcl no. 1, size of fcl no. 2, etc.). In this research we implemented three custom Multi-layers perceptron neural networks *MLP_1 - MLP_3*. The strides and the padding value used for all the Convolutional layers were fixed to 1. A ReLu activation function after each convolutional layer were considered as well as a Kaiming initialization for the weights and biased of all the layers.

Table 1: The structure of the implemented networks

Network	Network Structure	Parameters No.
CNN_1	conv: (1,64,10), (64,128,7), (128,128,4), (128,256,4), fcl: (4096,1)	38,951,745
CNN_2	conv: (1,64,10), (64,128,7), (128,128,4), (128,256,4), fcl: (4096,512,1)	41,045,825
CNN_3	conv: (1,64,10), (64,128,7), (128,128,4), (128,256,4), fcl: (4096,2048,512,1)	48,387,905
CNN_4	conv: (1,64,10), (64,128,7), (128,128,4), (128,256,4), (256,256,3), fcl: (4096,512,1)	20,664,385
CNN_5	conv:(1,64,12), (64,128,9), (128,128,7), (128,256,4), fcl: (1028,512,1)	4,897,349
CNN_6	conv: (1,64,10), (64,128,7), (128,256,4), fcl: (4096,512,1)	87,970,369
MLP_1	fcl: (4096, 2048, 1028)	55,701,553
MLP_2	fcl: (4096, 2048, 1028, 512)	56,227,885
MLP_3	fcl: (4096, 2048, 1028, 512, 128)	56,252,165

4 Results and analysis

As expected the accuracy of (CNN_1) has decreased in the 16-ways test to 60% in comparison with the 92% when evaluated on the Omniglot dataset. However, the other introduced variations on (CNN_1) have failed to reach a higher score. In (CNN_2) and (CNN_3) the number of the fully connected layers were increased to three and four respectively, which increased the number of parameters in the networks. In comparison with (CNN_1), The

[4]https://github.com/ZahirBilal/IrisRecognition-SiameseNeuralNetwork
[5]https://github.com/akshaysharma096/Siamese-Networks/blob/master/load$_{d}$*ata.py*
[6]https://sorenbouma.github.io/blog/oneshot/

accuracy of (CNN_2) has decreased insignificantly, whereas the accuracy of (CNN_3) has decreased drastically (almost 33% less accurate). We believe that having bigger network with 10 millions more parameters, has caused The network to display a steep over-fitting behavior (see A.2) and consequently to perform poorly.

Increasing the number of the convolutional layers in CNN_4 to 5 instead of 4 as in CNN_2 has decreased the accuracy insignificantly. This change has destabilized the learning process of CNN_4 (see A.2). This may indicate the deeper CNNs need less number of neurons in FC layers irrespective of type of the dataset[5]. Utilizing a network with a small number of parameters (CNN_5) has provided almost similar outputs in comparison with bigger networks (CNN_2). However the time needed to train the network has was almost half of the time needed for the first four networks. This makes such a small network an attractive option, as it is computationally inexpensive and its performance could be further enhanced by applying regularization techniques (see A.2).

The shallower network CNN_6 was a time consuming (double time of training CNN_2 were needed) and computationally expensive option, as it didn't enhanced the performance. As expected the SCNN has outperformed SNN with MLP architecture. The changing of the structure of the MLP networks didn't affect the accuracy significantly, which we couldn't explain its reasons. However, they were more prone to over-fitting (see A.2). Networks performances has shown improved results in general when batch normalization layers and dropout layers were introduced (see A.2). The results were verified by testing the all the networks on the Ubiris II dataset, the final results were relatively similar to MMUII (see A.1).

Table 2: The performance of the implemented networks on MMUII dataset

	Accuracy			
Network name	2-ways	4-ways	8-ways	16-ways
CNN_1	**0.92**	**0.80**	**0.70**	**0.60**
CNN_2	0.87	0.75	0.63	0.53
CNN_3	0.67	0.49	0.31	0.23
CNN_4	0.84	0.70	0.50	0.39
CNN_5	0.85	0.73	0.57	0.43
CNN_6	0.82	0.64	0.50	0.41
MLP_1	0.73	0.51	0.35	0.22
MLP_2	0.75	0.55	0.38	0.31
MLP_3	0.74	0.53	0.43	0.29

5 Conclusion and Future Work

Several factors should be taken in consideration when investigating the performance of SNN. These factors are usually interconnected and coupled such as the complexity of the task and the nature of the dataset. In this experiment we intended to evaluate the effect of the changes in the structure of the SNN on its performance. Therefore, we implemented two main architectures of SNN (CNN and MLP) with several variations. Increasing the number of embeddings layers for SCNN, increased the computational complexity and the training time as well as tends to destabilize the similarity learning process of the SNN, where smaller networks with few number of neurons in the embeddings layers has proved to be efficient at all previously mentioned aspects. In contrast, utilizing a bigger network with huge number of parameters (approx. 80 million parameter) were a time consuming and a computationally expensive process, with no significant improvement of the its accuracy. Deeper and shallower networks has failed to improve the accuracy of the network proposed in [1]. Deeper networks were prone to over-fitting. As expected, the Convolutional nueral networks as an architecture of the Siamese networks have outperformed the Multi-layers perceptron architecture. Expanded research in this fields is still required to generalize the results and to derive incisive conclusions. Different hyperparameters along with different network structure should be also experimented, in order to explain some of the unexpected results.

References

[1] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese Neural Networks for One-shot Image Recognition, 2015.

[2] Marina Angelovska, Sina Sheikholeslami, Bas Dunn and Amir H. Payberah. Siamese Neural Networks for Detecting Complementary Products. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop, Pages 65–70, 2021

[3] Figueroa-Mata G, Mata-Montero E. Using a Convolutional Siamese Network for Image-Based Plant Species Identification with Small Datasets. Biomimetics (Basel). 2020;5(1):8. 2020

[4] Tharindu Ranasinghe, Constantin Orasan and Ruslan Mitkov. Semantic Textual Similarity with Siamese Neural Networks. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), Pages 1004–1011, 2019.

[5] S.H.S. Basha, S.R. Dubey and V. Pulabaigari et al., Impact of fully connected layers on performance of convolutional neural networks for image classification, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.10.008

[6] S. Maitra, R. K. Ojha and K. Ghosh, "Impact of Convolutional Neural Network Input Parameters on Classification Performance," 2018 4th International Conference for Convergence in Technology (I2CT), 2018, pp. 1-5, doi: 10.1109/I2CT42659.2018.9058213.

A Appendix

A.1 N-Ways evaluation using UBIRISII dataset

Table 3: The performance of the implemented networks on UBIRISII dataset

	Accuracy			
Network name	2-ways	4-ways	8-ways	16-ways
CNN_1	**0.92**	**0.83**	**0.74**	**0.59**
CNN_2	87	77	64	52
CNN_3	0.65	0.50	0.35	0.21
CNN_4	0.85	0.74	0.55	0.39
CNN_5	0.88	0.74	0.57	0.43
CNN_6	0.80	0.66	0.49	0.40
MLP_1	0.78	0.60	0.47	0.37
MLP_2	0.77	0.63	0.48	0.36
MLP_3	0.79	0.68	0.53	0.42

A.2 Training and validation loss

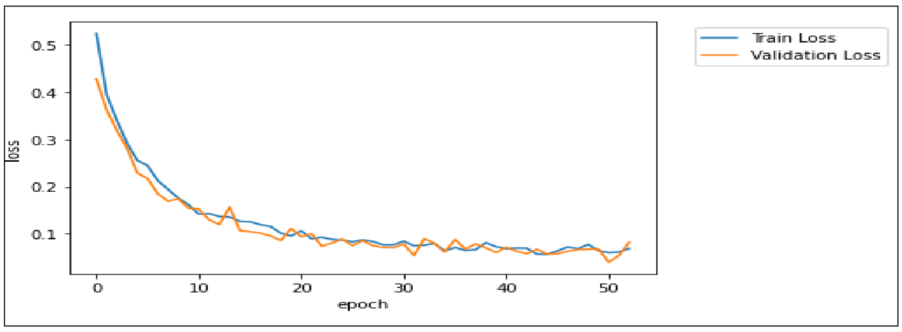

Figure 2: Training and Validation Loss over training epochs - CNN_1.

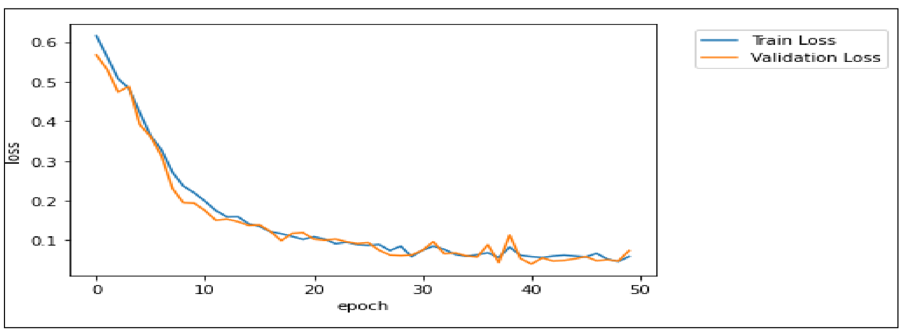

Figure 3: Training and Validation Loss over training epochs - CNN_2.

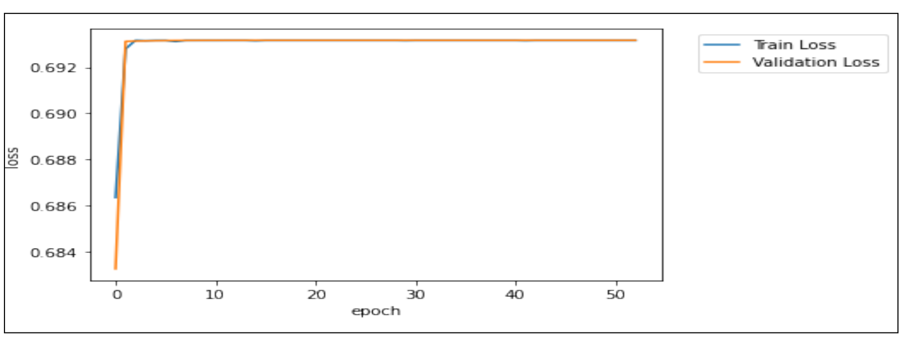

Figure 4: Training and Validation Loss over training epochs - CNN_3.

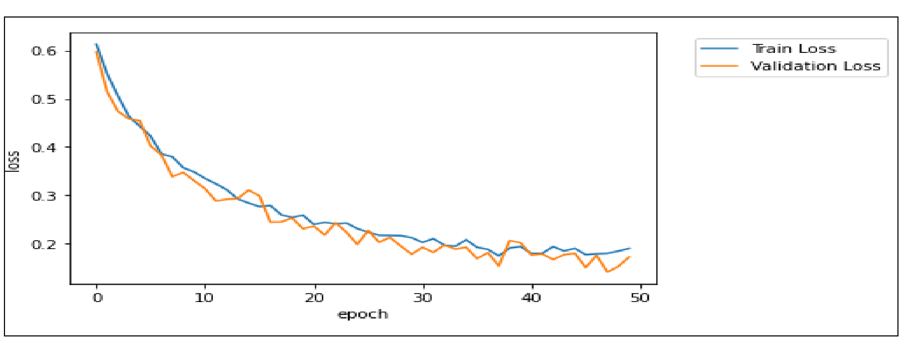

Figure 5: Training and Validation Loss over training epochs - CNN_3 (with 4 batch normalization's layers .

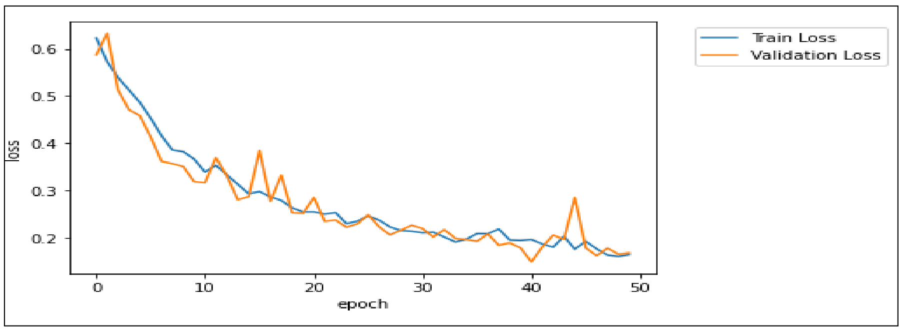

Figure 6: Training and Validation Loss over training epochs - CNN_4.

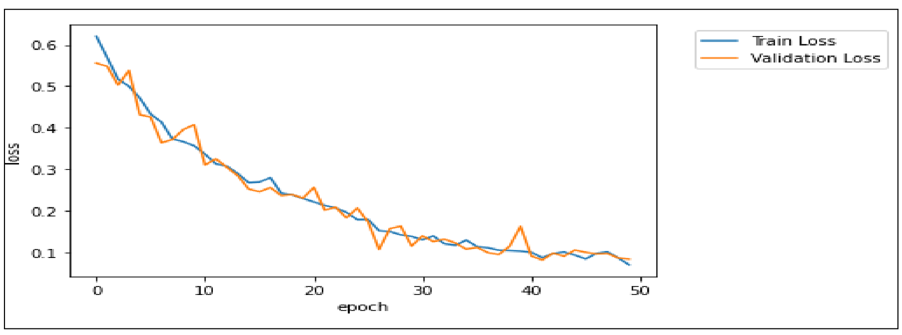

Figure 7: Training and Validation Loss over training epochs - CNN_5.

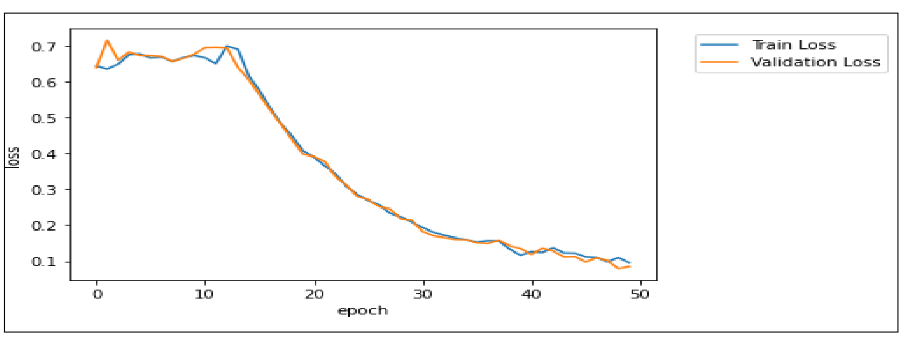

Figure 8: Training and Validation Loss over training epochs - CNN_6.

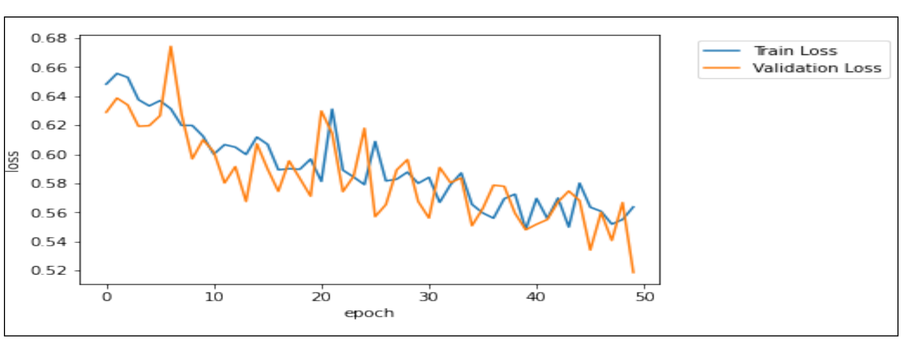

Figure 9: Training and Validation Loss over training epochs - MLP_1.

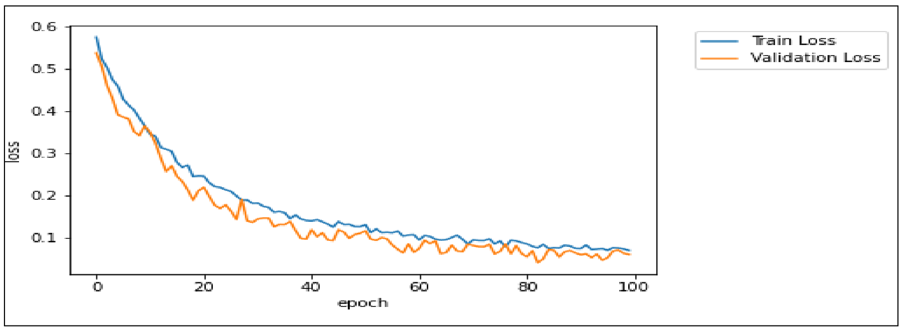

Figure 10: Training and Validation Loss over training epochs - MLP_1 with three batch normalization layers).

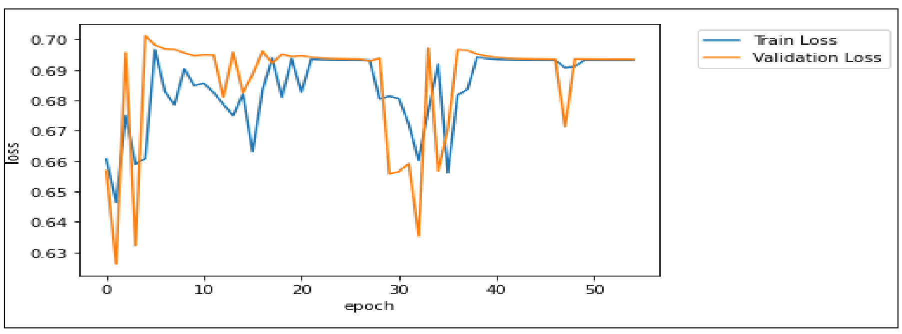

Figure 11: Training and Validation Loss over training epochs - MLP_2.

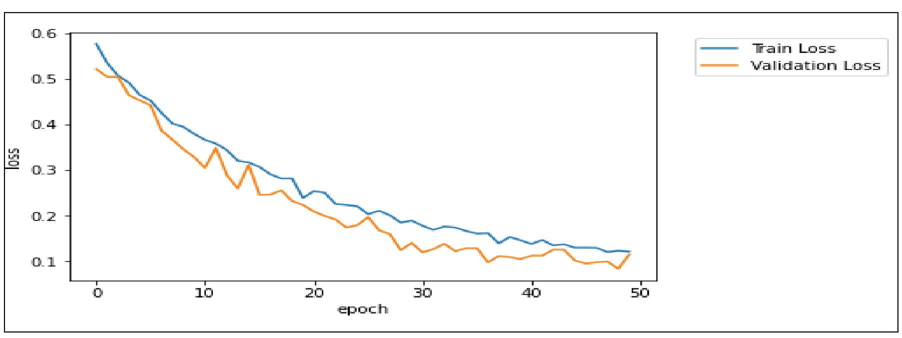

Figure 12: Training and Validation Loss over training epochs - MLP_2 with four batch normalization layers).

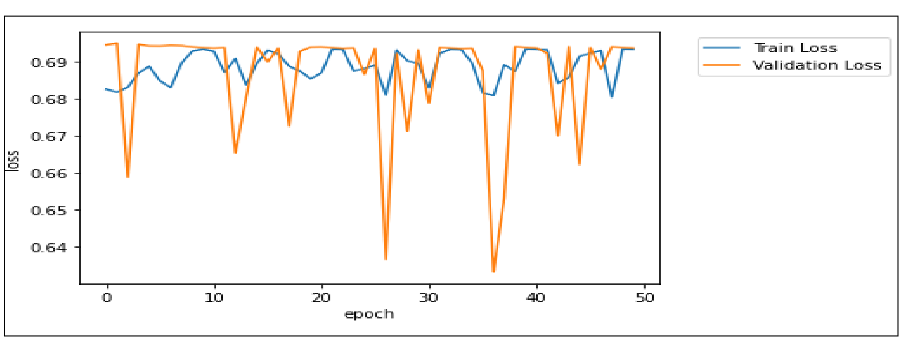

Figure 13: Training and Validation Loss over training epochs - MLP_3.

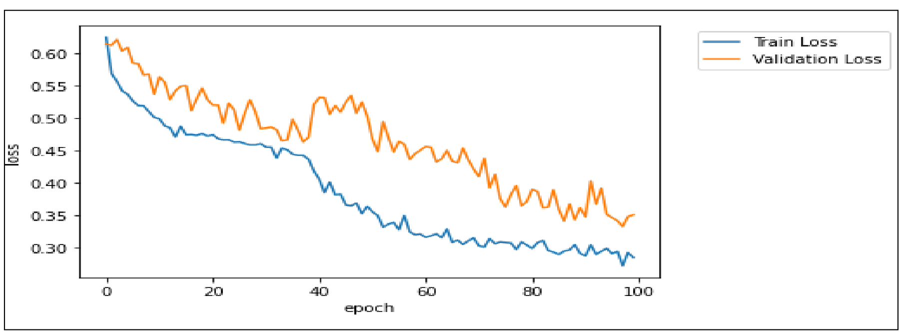

Figure 14: Training and Validation Loss over training epochs - MLP_3 (with five batch normalization layers).