EEL6814
Neurual Network and Deep Learning
Project 2

Multilayer Perceptron Variants — Stacked Autoencoder and Quadratic Mutual Information

Authors

Hao Li (UFID: 5431-1149): generation, training and testing of the SAE model with MLP classifier; writing paper sections of SAE and comparison among SAE+MLP, MLP+QMI and CNN.

Chenxing Dong (UFID: 6899-6096): generation, training and testing of the MLP model with QMI loss; writing paper abstract, sections of introduction, data, MLP+QMI and conclusion.

Multilayer Perceptron Variants — Stacked Autoencoder and Quadratic Mutual Information

Hao Li and Chenxing Dong

***Abstract*—We modified the multilayer perceptron (MLP) classification architecture through two approaches. The first approach is to construct a stacked autoencoder (SAE) with MLP layers. We extracted the bottleneck layer of the trained SAE and did the classification using another MLP. The second approach is to keep only the encoder MLP and train it with the quadratic mutual information (QMI) loss, and classify the outputs using a maximum a posteriori (MAP) estimator. The SAE approach achieved a testing accuracy of 89.65% while the QMI approach achieved a testing accuracy of 82.53%. The two approaches were compared with a standard CNN classifier, who can achieve an accuracy greater than 90%, but runs slower than both approaches.**

***Index Terms*—multilayer perceptron, stacked autoencoder, quadratic mutual information**

I. Introduction

MULTILAYER perceptron (MLP) is a feedforward artificial neural network (ANN) that consists of only fully connected layers. For a layer i in the MLP with n_i nodes, each node (i, j_i) in layer i is connected to all n_{i-1} nodes in the previous layer and all n_{i+1} nodes in the next layer. It receives the output $y_{i-1,j_{i-1}}$ from each node $(i-1, j_{i-1})$ in the previous layer and multiply them by the weights in these nodes w_{j_{i-1},j_i} that correspond to the node (i, j_i). The products for all nodes in layer $i-1$ are summed up, added a constant bias b_{i-1,j_i}, and then transformed by an activation function f, to form the output y_{i,j_i} of the node (i, j_i). $y_{i,j_i} = f\left(\sum_{j_{i-1}=1}^{n_{i-1}} w_{j_{i-1},j_i} y_{i-1,j_{i-1}} + b_{i-1,j_i}\right)$. The weights w_{j_{i-1},j_i} are the parameters that will be learnt from some learning algorithm, e.g. backpropagation. The architecture of an MLP can be controlled by the number of layers and the number of nodes in each layer.

MLPs often serve as basic bricks in more comprehensive ANNs. In this paper, we take two approaches to enhance the MLP network, and train them on the Fashion-MNIST dataset [1]. First, we construct a stacked autoencoder (SAE), which is an MLP (the encoder) followed by its mirrored MLP (the decoder), forming a symmetric layer structure. We train the SAE with the training data images, and then keep only the encoder part, and connect it with another MLP to serve as a classifier. Second, we directly train the encoder MLP with the quadratic mutual information (QMI) loss, which measures the mutual dependence among different dimensions of the MLP outputs, and use the maximum a posteriori (MAP) method to classify the MLP outputs.

In the following subsections of this section, we introduce the SAE, the QMI loss and the MAP classifier for the QMI loss. In Section II, we briefly introduce the Fashion-MNIST data used in this paper. We elaborate our SAE and QMI approaches in Section III and Section IV, respectively, and compare the two approaches in Section V. Finally, we draw our conclusion in Section VI.

A. Stacked Autoencoder

Stacked autoencoder (SAE; [2]) is a special case of the MLP, where the number of nodes in the hidden layer decreases at first but then increases, as the network forwards. The layer with the least number of nodes is called the bottleneck layer, and the architecture of the network is usually symmetric with respect to the bottleneck layer. The network prior to the bottleneck layer is called the encoder, and the network after the bottleneck layer is called the decoder. The number of dimensions in the output layer is usually the same as the number of dimensions in the input layer. To train an SAE, the input in the same as the target. The network first reduces the number of dimensions of the input, until it reaches the bottleneck layer, when it starts to increase the number of dimensions until it recovers the dimensions of the input. The learning algorithm makes the network learn to recover the input data in its output. After training the SAE, the bottleneck layer alone can represent the input data. For a classification problem, one can use another MLP or a support vector machine (SVM), which takes the bottleneck layer outputs as their inputs, and classify them into desired classes.

B. Quadratic Mutual Information

Mutual information is a measure of the statistical dependence between several random variables $\boldsymbol{x} = (x_1, x_2, ..., x_d)$, which is defined as

$$\begin{aligned} M(\boldsymbol{x}) &= D_{\mathrm{KL}}\left(P(\boldsymbol{x}), \prod_{i=1}^{d} P(x_i)\right) \\ &= \iint P(\boldsymbol{x}) \log \frac{P(\boldsymbol{x})}{\prod_{i=1}^{d} P(x_i)} \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{1}$$

where D_{KL} is the Kullback-Leibler divergence [3], also called the relative entropy, which is a measure of the statistical independence among these probability distributions. If $P(\boldsymbol{x})$ is divergent from $\prod_{i=1}^{d} P(x_i)$, it means the random variables in $\boldsymbol{x}$ are mutually dependent. Therefore, the greater the mutual information is, the more dependent the random variables are.

To simplify the computation, we use the quadratic mutual information (QMI; [4], [5]), which uses the Euclidean distance

among the probability distributions instead of the Kullback-Leibler divergence. In this case,

$$\begin{aligned}\mathrm{QMI}(\boldsymbol{x}) &= D_E\left(P(\boldsymbol{x}), \prod_{i=1}^{d} P(x_i)\right) \\ &= \iint \left(P(\boldsymbol{x}) - \prod_{i=1}^{d} P(x_i)\right)^2 \mathrm{d}\boldsymbol{x} \\ &\equiv V_J - 2V_C + V_M,\end{aligned} \tag{2}$$

where D_E represents the Euclidean distance, and

$$\begin{aligned}V_J &\equiv \iint P(\boldsymbol{x})^2 \mathrm{d}\boldsymbol{x}, \\ V_C &\equiv \iint P(\boldsymbol{x}) \prod_{i=1}^{d} P(x_i) \mathrm{d}\boldsymbol{x}, \\ V_M &\equiv \iint \prod_{i=1}^{d} P(x_i)^2 \mathrm{d}\boldsymbol{x},\end{aligned} \tag{3}$$

which represent the joint, combined and marginalized probability distributions, respectively. With N data points sampled under these probability distributions, we use the sum of multiple normal distributions to estimate the true probability distributions, i.e.

$$\begin{aligned}\hat{P}(x_i) &= \frac{1}{N}\sum_{k=1}^{N} G(x_i - \hat{x}_{i,k}, \sigma^2), \\ \hat{P}(\boldsymbol{x}) &= \frac{1}{N}\sum_{k=1}^{N} G(\boldsymbol{x} - \hat{\boldsymbol{x}}_k, \sigma^2),\end{aligned} \tag{4}$$

where σ is the standard deviation used for the Gaussian kernels, and

$$\begin{aligned}G(x, \sigma^2) &\equiv \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}, \\ G(\boldsymbol{x}, \sigma^2) &\equiv \frac{1}{\left(\sigma\sqrt{2\pi}\right)^d} e^{-\frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}{2\sigma^2}}.\end{aligned} \tag{5}$$

For a classification problem, assume there are T classes. For each class p, there are N_p data points, while each data point i has an output $\boldsymbol{y}_p(i)$. In total, there are $N = \sum_{p=1}^{T} N_p$ data points and each data point i corresponds to the output $\boldsymbol{y}(i)$. Then the three terms in Equation 3 can be estimated as

$$\begin{aligned}\hat{V}_J &= \frac{1}{N^2}\sum_{p=1}^{T}\sum_{i=1}^{N_p}\sum_{j=1}^{N_p} G\left(\boldsymbol{y}_p(i) - \boldsymbol{y}_p(j), 2\sigma^2\right), \\ \hat{V}_C &= \frac{1}{N^2}\sum_{p=1}^{T}\frac{N_p}{N}\sum_{i=1}^{N_p}\sum_{j=1}^{N} G\left(\boldsymbol{y}_p(i) - \boldsymbol{y}(j), 2\sigma^2\right), \\ \hat{V}_M &= \frac{1}{N^2}\left(\sum_{p=1}^{T}\frac{N_p^2}{N^2}\right)\sum_{i=1}^{N}\sum_{j=1}^{N} G\left(\boldsymbol{y}(i) - \boldsymbol{y}(j), 2\sigma^2\right).\end{aligned} \tag{6}$$

To make the learning algorithm find the maximum QMI, we define the loss as the negative of the QMI, i.e.

$$L = -\mathrm{QMI} = -\left(\hat{V}_J - 2\hat{V}_C + \hat{V}_M\right). \tag{7}$$

C. Classifier for QMI loss

Since the d-dimensional output used in estimating the QMI does not need to have the same number of dimensions as the number of classes T, it is not straightforward to predict the class of a data point given the d-dimensional output. Therefore, we take a maximum a posteriori (MAP) approach to determine the class p_0 of a given testing data output $\boldsymbol{y}_0$ from the trained MLP.

For each class p, collect the estimated outputs of all the training data points within this class $\{\hat{\boldsymbol{y}}_p(t)\}$, and estimate the probability distribution $P(\boldsymbol{y}|p)$ using the sum of normal distributions centered at each training output, as defined in Equation 4:

$$\hat{P}(\boldsymbol{y}|p) = \frac{1}{N}\sum_{t=1}^{N_p} G(\boldsymbol{y} - \hat{\boldsymbol{y}}_p(t), \sigma^2), \tag{8}$$

where N_p is the number of all training data points within class p. Given a testing output $\boldsymbol{y}_0$, we estimate the probability $P(\boldsymbol{y}_0|p)$ for each class p, and select the class with the maximum $P(\boldsymbol{y}_0|p)$:

$$\hat{p}_0 = \arg\max_p P(\boldsymbol{y}_0|p). \tag{9}$$

II. Data

The Fashion-MNIST dataset [1] is used in training and testing of our SAE and QMI approaches. Initially inspired by the MNIST dataset [6], the Fashion-MNIST dataset is aiming to replace the MNIST dataset, since it is not capable of representing modern computer vision problems, and is too widely used in the literature, and is too easy for modern ANNs.

The images in the Fashion-MNIST dataset are 28×28-pixel grayscale images, which is the same as the MNIST dataset. The images also come from 10 categories, but the categories are all fashion products, instead of digits. These categories are T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot, which are labeled 0 to 10. The specific labeling of the categories and example images are shown in Figure 1.

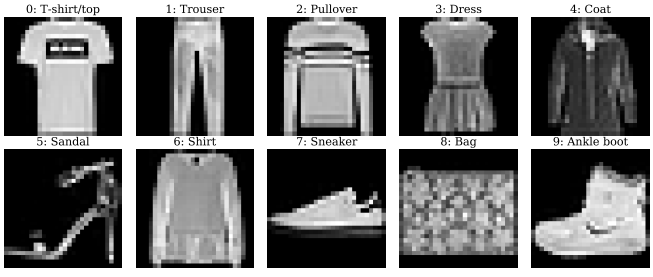

Fig. 1. Example images of the 10 categories of the Fashion-MNIST dataset.

The dataset is already separated into a training dataset with 60,000 images and a testing dataset with 10,000 images. In our practice, we split 10% of the training dataset into a validation set to be used in the hyperparameter search.

III. Stacked Autoencoder with MLP classifier

A. Stacked Autoencoder

To select the optimal network architecture of the stacked autoencoder network, a base model is utilized to experiment through various values of hyperparameters. This model consists of a three-layered stacked autoencoder, which is trained with a two-layered decoder. The first two layers of the encoder are to reduce the dimensionality of the data, which has a low impact on the final result. The sizes are set to 500 and 200. Differing from the first two layers, the third layer, which is called the bottleneck layer as well, is critical to the final result since it represents the final generalization before the decoder receives the data. The decoder contains two layers that receive the output from the bottleneck layer and reconstruct the data with the same amount of units from the first two encoder layers.

1) Output layer size: The third layer in the encoder is important for the optimal result of generalization. The layer size is determined by experimenting through a few values with the established SAE network. We utilized the mini-batch gradient descent for optimization and minimum squared error as the loss function. The learning rate is set as default at 10^{-4} with a batch size of 300 and epoch number up to 20. The training process is set to optimize based on the validation loss. Figure 2 exhibits the differences between each layer size.

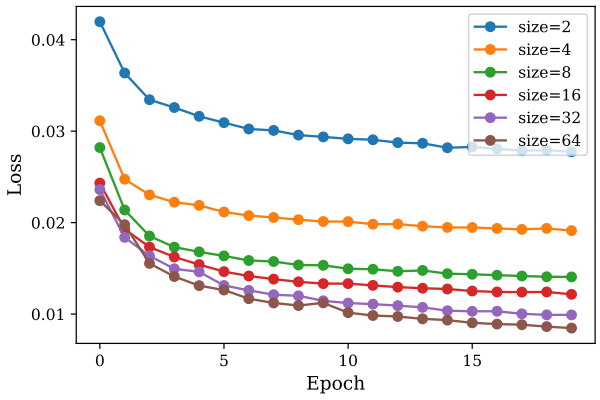

Fig. 2. Performance in terms of test loss for different bottleneck layer size of SAE network

The goal of this network is to reduce the dimensionality of the data and extract features. Thus, the lower the encoder layer size, the better the generalization is. As a result, we started with layer size at 2 and an increment of 1 on the power basis. It is observed that the test loss decreases clearly as the layer size increases. On the other side, the declining trend of the loss yields after the loss dropping under 0.02. This finding indicates that the increasing value of layer size does not make much contribution if the layer size is greater than 4. To better prove this finding, the reconstructed image from the trained network is presented in Figure 3. According to the image comparison, the reconstructed image with a layer size of 16 possesses more features comparing to the ones with a layer size of 8 particularly for the T-shirt top class. Therefore, layer size of 16 for the encoder output is the optimal choice because both the reconstructed image and loss achieve the goal of reducing dimensionality while extracting the features.

The final stack autoencoder network has an architecture as follows.

- An input layer with $28 \times 28 = 784$ nodes
- A fully connected layer with 500 nodes, activated by ReLU
- A fully connected layer with 200 nodes, activated by ReLU
- A fully connected layer with 16 nodes, activated by ReLU
- A fully connected layer with 200 nodes, activated by ReLU
- A fully connected layer with 500 nodes, activated by ReLU
- A fully connected output layer with 784 nodes, activated by Sigmoid

B. Multilayer Perceptron

Based on the SAE network, an MLP model is further developed for classification purposes. Using the trained encoder layers, the MLP model takes the output from the encoding part and implement classification for a potential improvement on MLP performance. In this section, the work is focused on finding the optimal hyperparameters for the SAE-MLP combined model with the previous knowledge of MLP model optimal parameters.

1) Network Architecture: The MLP network inherits the optimal hyperparameters from the previous experiment, which uses Adam optimizer with MSE for loss function. There are two hidden layers with 500 and 256 units and one output layer of 10 units. The input is the output of the trained encoder network. Based on this model, we tested various layer setting and size for the best performance. In Figure 4, the performance of the network in terms of no hidden layer, one hidden layer, and two hidden layers are presented with 50 epochs and an encoder output layer size of 16. The observation from the graph illustrates that the performance with two hidden layers is the smoothest and highest.

2) Bottleneck layer size: In the previous section of the stacked autoencoder, we discussed the optimal output layer size of the encoder in terms of the reconstruction effect. In this section, the performance of classification in terms of bottleneck layer size from the encoder is further developed for the optimal layer size.

To better find the optimal bottleneck layer size, the MLP network is trained and tested with the same set of layer size set from the previous section in terms of 2, 4, 8, 16, 32, and 64. The result is displayed in Figure 5 with accuracy score. It is seen that the performance has a similar trend as the stacked autoencoder network. The accuracy increases as the size of the bottleneck layer increases. The difference in performance drops critically after the layer size reaches 8. As the figure shows, all final accuracy scores after a layer size of 8 converge to around 0.88 despite few outliers. Note that layer sizes under 16 all start from low and raise to a certain level until later epoch. This feature is found from all layer sizes that are under

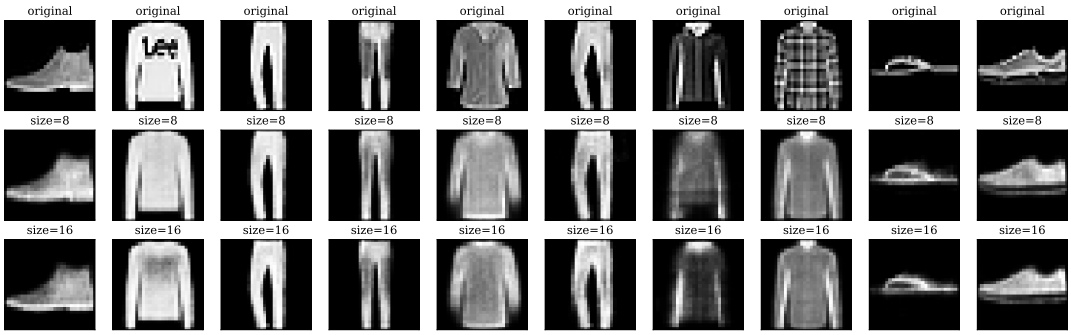

Fig. 3. Decoded image with bottleneck size of 8 and 16 comparing to the original image

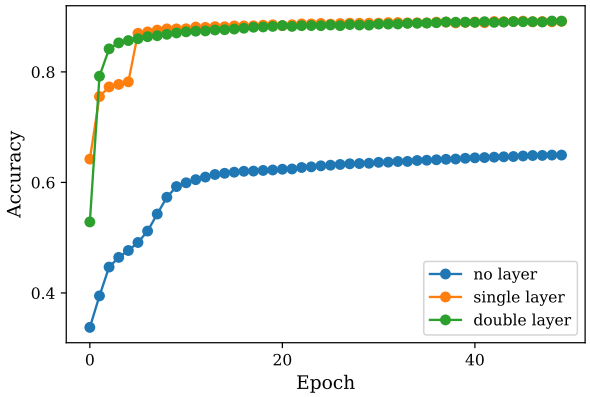

Fig. 4. Performance of different layer settings for SAE-MLP network

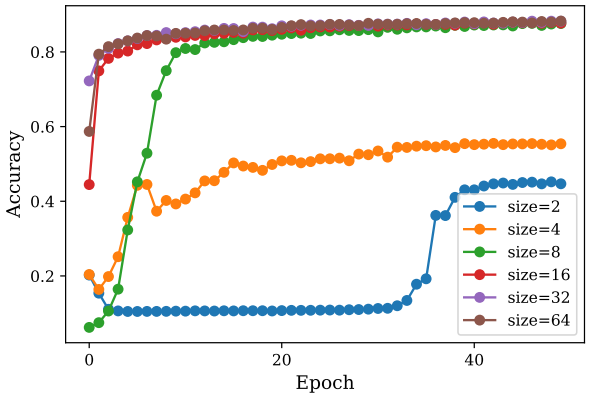

Fig. 5. Performance in terms of test accuracy for different bottleneck layer size of SAE-MLP network

the output layer size of 10. Thus we conclude that the layer before the output layer in the MLP model should be larger than the output layer size to avoid the progress of raising from a poorer performance. Considering the average performance and the relationship between output layer size and bottleneck layer size, we thus select 16 units as the ideal layer size since this is the smallest layer size that maintains the highest average and maximum accuracy score.

C. Results

After thoroughly considering and testing for the hyperparameter set, the optimal MLP architecture is as follows.

- An input layer with 16 nodes
- A fully connected output layer with 500 nodes, activated by ReLU
- A fully connected output layer with 256 nodes, activated by ReLU
- A fully connected output layer with 10 nodes, activated by Sigmoid

The process of the overall SAE-MLP network is to first train the SAE network with a decoder. Then the output of the encoder layers is set as the input of the MLP network. The SAE thus performed the feature extraction for the MLP network while the MLP model classifies the data into 10 classes. Figure 6 shows the loss and performance of the optimal network. Both the loss and accuracy curves are smoothly reaching the best performance level. The highest accuracy score is 89.65% while the average accuracy is 88.52%. Figure 7 is the confusion matrix of the SAE-MLP network classified results compared to the original labels of the data.

IV. MLP with QMI loss

We now discard the decoder part of the autoencoder after the bottleneck layer, thus keep a 2-hidden-layer MLP as the network, and train it with the QMI loss. Because the QMI loss computation takes a large amount of time, for hyperparameter search, we used only 1% of the Fashion-MNIST training data that consists of only 600 images. We split these 600 images into a training set with 540 images and a validation set with 60 images. For each set, the classes of the images are equally distributed, i.e. each class has the same number of images in each set. Because the QMI loss takes the distributions of the input data, the more data points in a batch, the higher accuracy the distributions can be sampled and thus the better performance can the QMI loss achieve. However, the computation time of the QMI loss goes cubicly with the batch

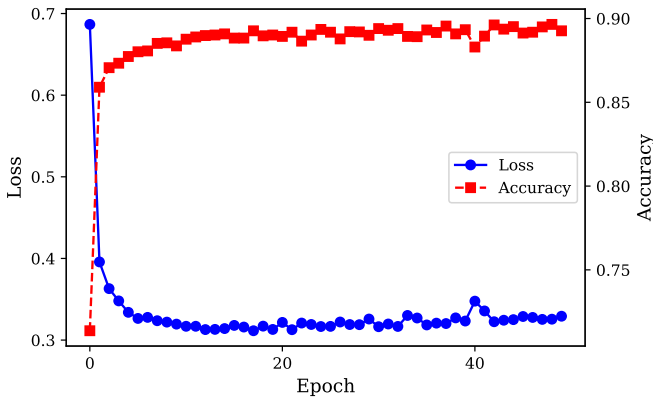

Fig. 6. Performance of optimal SAE-MLP network

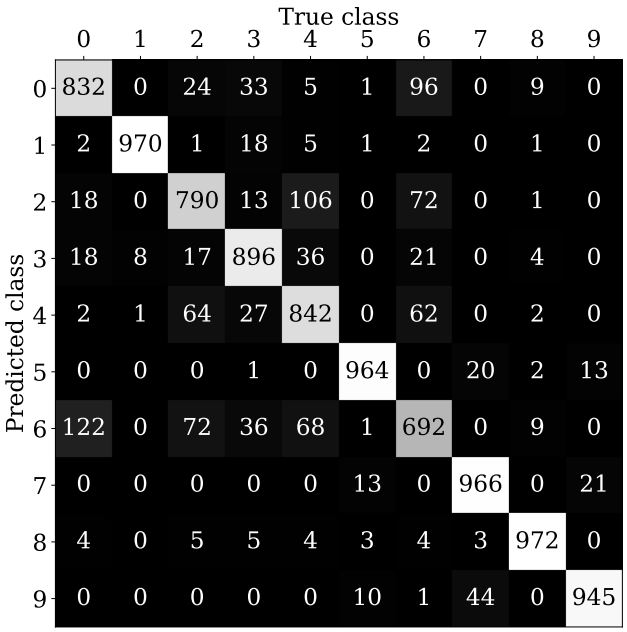

Fig. 7. Confusion matrix of optimal SAE-MLP network

size. As a compromise, we choose to use a batch size of 50 in the hyperparameter search to reduce computation time.

For the MLP architecture, we used 500-200 nodes in the two hidden layers, with ReLU activation functions in the hidden layers. We used the QMI loss and the Adam learning algorithm [7] as inspired from [8]. For a fiducial model used in the hyperparameter search, we took an output layer with 2 nodes, the QMI kernel size $\sigma = 1$, a learning rate of 10^{-3} and training for 40 epochs.

A. Output layer size

We started with 2 nodes in the output layer, i.e. a 2-dimensional output, and tested output dimensions of 2, 3, 4, 5, 6, 8, 12, 16 and 32. The corresponding best validation accuracies are shown in Figure 8. The accuracy raised up from output dimension of 2 to 4, and then fell down for larger output

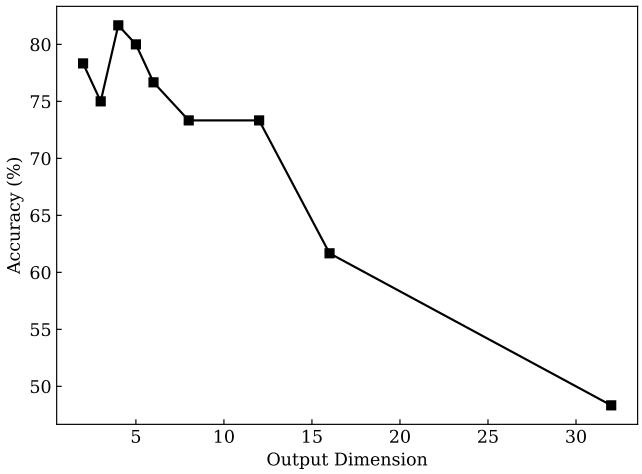

Fig. 8. The best validation accuracies achieved for the MLP model as a function of epoch for different number of dimensions in the output layer, trained using only 1% of the data as the hyperparameter search procedure.

dimensions. Therefore, we choose to use an output size of 4 as our new fiducial model.

B. QMI kernel size

The Gaussian kernel size σ affects how the distributions of the network output differences are sampled. We tried $\sigma = 0.5$, 0.7, 0.8, 0.9, 1, 1.1, 1.2, 1.5, 2, 4 and 8. The corresponding best validation accuracies are shown in Figure 9. The accuracy

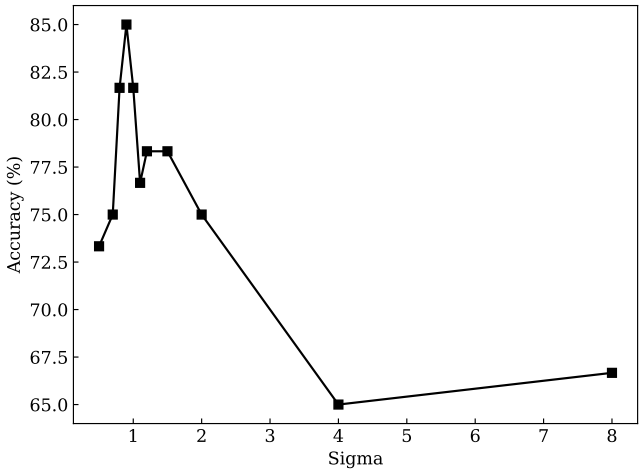

Fig. 9. The best validation accuracies achieved for the MLP model as a function of the Gaussian kernel size σ used in computing the QMI loss, trained using only 1% of the data as the hyperparameter search procedure.

raised up from σ of 0.5 to 0.9, and then fell down for larger σ values. Therefore, we choose to use a kernel size of $\sigma = 0.9$ as our new fiducial model.

C. Learning rate

We tested learning rates of 10^{-2}, 10^{-3}, 10^{-4} and 10^{-5} running for up to 40 epochs. The corresponding validation

loss and accuracies are shown in Figure 10 and Figure 11, respectively. The models with learning rates of 10^{-3} and

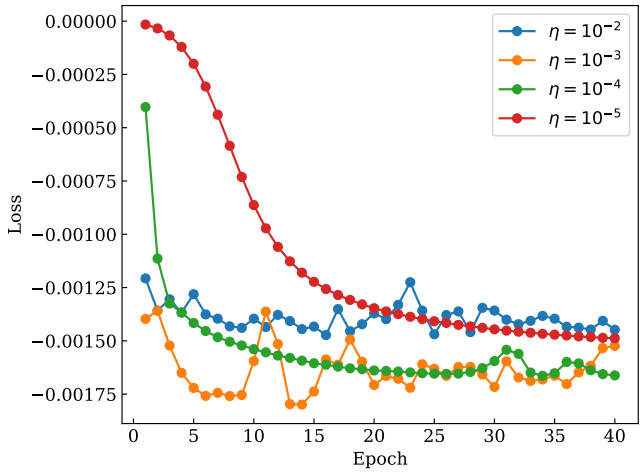

Fig. 10. The validation loss for the MLP model as a function of iteration steps for different learning rates, trained using only 1% of the data as the hyperparameter search procedure.

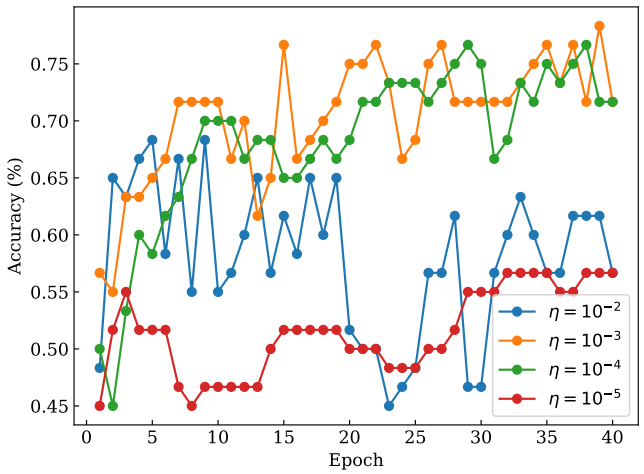

Fig. 11. The best validation accuracies achieved for the MLP model as a function of iteration steps for different learning rates, trained using only 1% of the data as the hyperparameter search procedure.

10^{-4} are able to reach a lower loss and higher accuracy. The accuracies achieved by them are comparable. However, the loss change for the learning rate of 10^{-3} is not monotonous, indicating a step size too coarse to converge to a local minimum. Therefore, we decided to use a learning rate of 10^{-4}.

D. Results

After searching in the hyperparameter space, we decided to use the MLP architecture as follows.

- An input layer with $28 \times 28 = 784$ nodes
- A fully connected layer with 500 nodes, activated by ReLU
- A fully connected layer with 200 nodes, activated by ReLU
- A fully connected output layer with 4 nodes, without activation

The QMI loss function with a kernel size of $\sigma = 1$ and the Adam learning algorithm with a learning rate of 10^{-4} are used. The data is organized into batches with size 200 and is trained for 40 epochs. The testing loss and accuracy evolution for different iterations are shown in Figure 12. The best testing

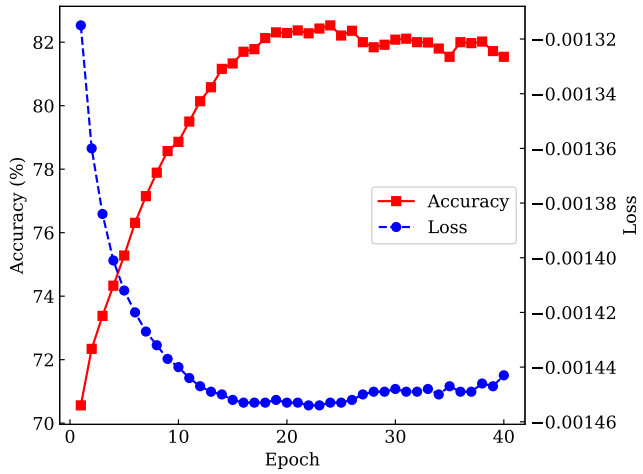

Fig. 12. The testing accuracy and loss change as a function of number of iterations for the MLP model with QMI loss trained and tested using the full Fashion-MNIST data, using our previously determined hyperparameters.

accuracy of 82.53% is achieved at the 24th epoch, and the average accuracy for the 18th to 26th epochs reaches 82.32%, with a standard deviation of 0.12%.

The testing confusion matrix for the best performance MLP model is shown in Figure 13. From the confusion matrix, we see that the MLP with QMI loss performs well for most cases except for only 4 pairs of true classes and predicted classes: (1) The true class is 4 (coat) but the predicted class is 2 (pullover), (2) The true class is 6 (shirt) but the predicted class is 0 (T-shirt/top), (3) The true class is 6 (shirt) but the predicted class is 2 (pullover), and (4) The true class is 2 (pullover) but the predicted class is 4 (coat), which is the counter case of (1). These 4 cases are among only 4 classes, all of which are upper clothes, and 3 of them all have long sleeves, which are hard to distinguish even for humans. If we exclude these 4 cases, the model performance can have a large improvement, reaching a testing accuracy of 92.72%.

V. Comparisons

In this section, we will compare the MLP model with the SAE network, the MLP model with QMI loss function, and the CNN model from former work. To quantify the comparison, the models are distinguished based on the computation time, performance, and network architecture. All results are displayed with 9 epochs.

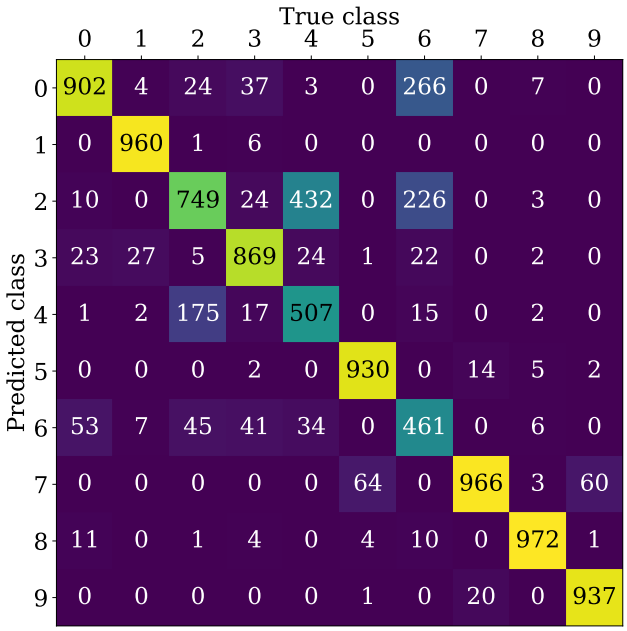

Fig. 13. The confusion matrix for the best performance QMI loss based MLP model with a testing accuracy of 82.53%.

A. Computation time

Figure 14 shows the computation time for each epoch in the entire span of training. It is observed that the CNN spends the most time for computation and MLP with QMI loss function follows the second. The MLP with the SAE model has the least amount of computation time. Considering the computation complexity, the CNN model requires a kernel to swipe through each image from the data. It needs to perform matrix multiplication for every image and thus requires the most amount of time to complete one epoch. The MLP with QMI model and the MLP with the SAE model both take encoder output as input, however, the loss function choice made a difference. Comparing to the QMI loss function requiring the computation of statistical independence among probability distributions, the MSE loss function simply calculates the mean squared error. There is a gap between computation complexity. As a result, this computation time graph matches our understanding of each model.

B. Performance

Figure 15 exhibits the accuracy of testing, which indicates different findings from the loss graph. The CNN model has the highest accuracy score and the MLP with the SAE model is the second highest while the MLP with the QMI model has the lowest accuracy.

Although the graph is plotted based on 9 epochs, which is limited to CNN training time, the finding behind the graphs stays the same. We can state that the CNN model has a leading performance comparing the two models using MLP. It is assumed that this is caused by the essential computation method CNN uses that captures more details than the MLP model. Between the two MLP models, the MLP with QMI model performs relative poor comparing to the MLP with the SAE model.

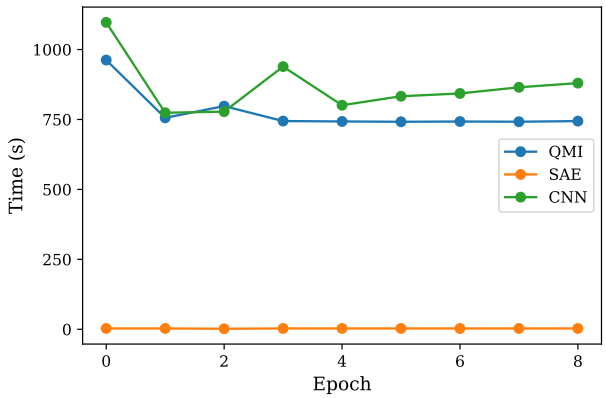

Fig. 14. The computation time for MLP with SAE, MLP with QMI and CNN models for the first 9 epochs in training.

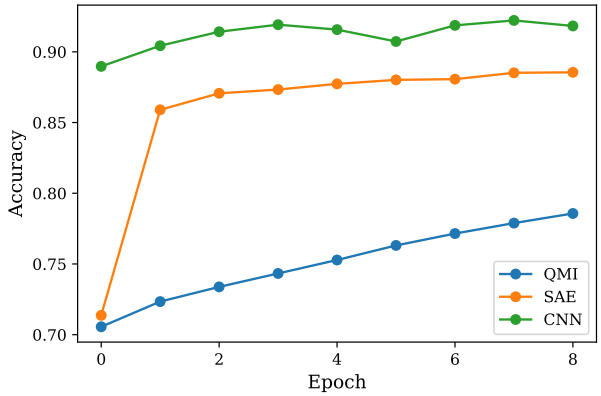

Fig. 15. The performance in terms of accuracy for MLP with SAE, MLP with QMI and CNN models for the first 9 epochs in training.

C. Network Architecture

Differing from the CNN model, the two MLP models rely on the output from the encoder network. The training of the SAE network performs feature extraction and reduces layer complexity during the classification process. One thing to notice is that the MLP with the QMI model does not require the same amount of bottleneck layer size for the encoder. Taking the performance from previous layer size testing of layer size 4, Figure 16 proves that the QMI loss function provides a better result with the same number of outputs than the MSE loss function.

Moreover, comparing the confusion matrices of the two MLP models (Figure 7, Figure 13) and the CNN confusion matrix [8], some features of each network model can be found. The CNN result has the highest performance, however, its true-false class is sparsely distributed compared to the MLP models using SAE output. This is due to the SAE feature extraction characteristics that provide a good generalization of the data.

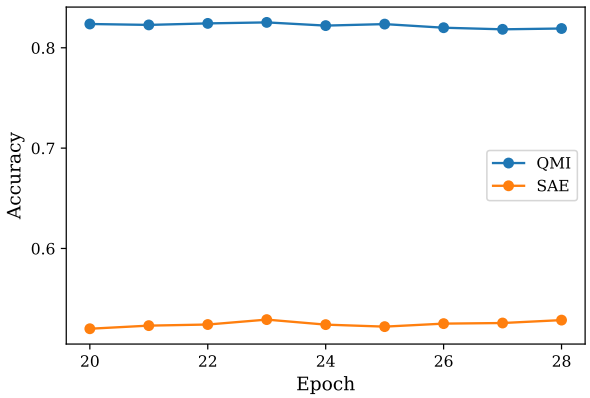

Fig. 16. The comparison of MLP models under the same bottleneck layer size (4) for the 20–28th epochs in training.

VI. Conclusion

We applied two variants of the MLP network on classification of the Fashion-MNIST dataset. The first one is an SAE with a 500-200-16-200-500 architecture. We took the bottleneck layer of the trained SAE and trained it with another MLP with an 16-500-256-10 architecture for classification. The second one is a normal MLP with a 500-200-4 architecture, but trained using the QMI loss, and the outputs was classified using the MAP estimator. We searched for the optimal hyperparameters in both approaches using the validation set. Our SAE with MLP classifier achieved an accuracy of 89.65%. Our MLP with QMI loss and MAP classifier achieved an accuracy of 82.53%, and most testing errors occur under only 4 special cases, excluding which results in an caccuracy of 92.72%. We compared the two MLP variants with the CNN classifier elaborated in [8], which achieved an accuracy of 91.39%. The CNN outperforms the two MLP variants in testing accuracy, but takes the most computation time per epoch. The SAE+MLP model outperforms the MLP+QMI model, but couldn't do well with the same number of nodes in the bottleneck layer as the MLP+QMI model.

References

[1] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv e-prints*, p. arXiv:1708.07747, Aug. 2017.

[2] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.

[3] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 03 1951. [Online]. Available: https://doi.org/10.1214/aoms/1177729694

[4] J. C. Principe, D. Xu, and J. Fisher, "Information theoretic learning," in *Unsupervised adaptive filtering*. Citeseer, 2000, vol. 1, ch. 7, pp. 265–319.

[5] J. C. Principe, D. Xu, Q. Zhao, and J. W. Fisher, "Learning from examples with information theoretic criteria," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 26, no. 1-2, pp. 61–77, 2000.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec. 2014.

[8] C. Dong and H. Li, "MLP and CNN on the Fashion-MNIST dataset," *EEL6814 Neurual Network and Deep Learning*, vol. Project 1, Nov 2020.