Autoencoder and Quadratic Mutual Information

Pengyu Zhang(UFID:94598565)
Dept. Electrical and Computer Engineering
University of Florida
pengyu.zhang@ufl.edu

Zihan Liao(UFID:19613125)
Dept. Electrical and Computer Engineering
University of Florida
liaozihan1@ufl.edu

***Abstract*—This project is separated into two parts. The first part is first to train an autoencoder followed by feeding the output of the encoding part to a multi-layer perceptron (MLP). The Fashion-Mnist dataset is consistently used through this whole project. The Second implementation is to use the quadratic mutual information (QMI) criterion to train the MLP. The performance of using QMI and the combination of the autoencoder and MLP will be compared with the performance of the convolutional neural network (CNN). The performance includes the accuracy of the classification result of different models on the test datatset, and the running time will also be discussed.**

***Index Terms*—Neural Network, autoencoder, QMI, CNN**

I. Introduction

Autoencoder has been widely used in many areas of machine learning such as image coloring, feature variations, dimensionality reduction and denoising. The bottleneck layer of the autoencoder produces the low-dimension and great informative feature for a high order input, which provides the ability of feature extraction and low-dimensional solution. Neural network has been widely used both in academic and industry. The MLP is one of the earliest architecture that has been proved to have the ability to recognize different images. The combination of autoencoder and the MLP is basically using the high informative features that extracted by the autoencoder from the high order as inputs which are further fed into the MLP which act as a classifier. The QMI criterion, however, abandoned the concept of the loss between prediction and true value when evaluating the performance of the model and doing backpropagation. It uses the mutual information between the output distribution and the distribution of the ground truth. Due to the definition that it computes the difference between the distribution instead of the error between ground truth and the prediction, the QMI also does not ask the number of the output units and the ground truth are identical. When evaluating the performance, the predominant CNN model is used as a canon due to its great power dealing with images.

In this project, we set up a variety of choices on hyperparameters to explore how different architectures, learning rates affect the final classification performance.

The first part(autoencoder) was finished by Zihan Liao, the second part(QMI) was finished by Pengyu Zhang.

II. Experiments Design

A. Autoencoder experiment design

Five layers autoencoder will be chosen as the basic model. The encoder part and the decoder part are symmetric. The number of hidden units of layers besides the bottleneck layer will be chosen as 500 and 200 symmetrically. The number of the hidden units of the bottle neck layer will be tuned in order to achieve a best reconstruction in the decoding part.The autoencoder uses mean squared error(MSE) criterion and Adam algorithm to do training. After the best autoencoder model is chosen, the output of the bottle neck layer will be fed into the MLP. The two layer MLP acts as a classifier.Tuning the hyperparameters such as learning rate, number of hidden units, and stopping timing. Crossentropy and Adam algorithm will be used in training the MLPs. Putting the testing data through the autoencoder and feeding the output into the MLPs do evaluate the performance of the model. The evaluation of the performances includes the classification accuracy and the time consumption of the model.

B. MLP with QMI and MAP design

Since QMI outputs scales to estimated the distribution differences, unlike cross-entropy, it doesn't own the ability to classify different clusters. For our classification purpose, we need to concatenate a maximum a posterior (MAP) classifier at the tail of the original MLP model.

Therefore, we built a Gaussian-based MAP classifier given the outputs and the corresponding labels after the neural network was trained. In testing stage, the probabilistic model is treated as classifier model once we obtained the network outputs on test data. For continuity, we kept the autoencoder and MLP the same as the best model configuration we got in the former experiments. To accelerate the training speed, we chosen 200 samples as the mini-batch size since QMI is of much more computational complexity. For the testing part, each testing data will compute MAP with each class's training data. The prediction equals to the class that has the greatest probability.

III. Corresponding Results

A. Autoencoder

1) Hidden units selection: As talked about earlier, the basic model of the autoencoder is a five-layer autoencoder which uses the training images as input and label. The hardest

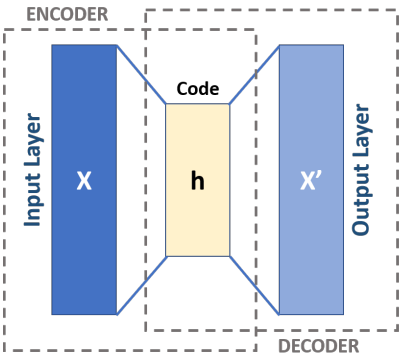

Fig. 1: Architecture of a classical autoencoder[1]

part to design the autoencoder is to design the number of the hidden units of the bottleneck layer. In this project, the mission of the autoencoder has to complete can be thought as two parts. The first part we can see this process as a dimensionality reduction process which project a thousands dimension image to a low dimension feature. Therefore, for the purpose of dimensionality reduction, the number of the hidden units should not be too big. Another way to think of this process is to do feature extraction from the original image which means we should preserve sufficient number of hidden units to keep sufficient information from the original image. In short, the goal is to keep the bottleneck layer simple enough but keep the ability to reserve most information when doing reconstruction. The image reconstruction by using 16, 64 and 128 units in the bottleneck layer are shown by Fig2, Fig3 and Fig4. Both three different autoencoder are trained in ten epochs by using Adam algorithm.

Fig. 2: Image reconstruction(16 hidden units)

Fig. 3: Image reconstruction(64 hidden units)

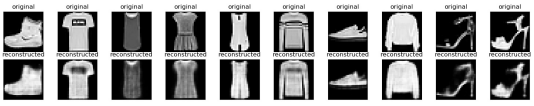

Fig. 4: Image reconstruction(128 hidden units)

We can clearly see from these pictures that when we are using more hidden units, we can get better reconstruction. For example, for the first shoe picture, the 16 units model cannot preserve the shape of the shoe accurately comparing to the reconstruction picture of 64 and 128 hidden units layer. Meanwhile, comparing the 64 and 128 units layer, the information of the clothes like color and stripes on the clothes is blurred when using 64 hidden units layer comparing 128 hidden units layer.

By experimenting, when the number of hidden units are greater than 128, the improvements on the reconstruction picture are trivial. Therefore, the number of hidden units we finally set equals to 128.

2) Classifier: The architecture of the classifier is a two-layer MLP. The learning algorithm I used is the Adam algorithm. The loss function is crossentropy criterion. The activation function of the first layer is Relu, and the activation function of the output layer is the softmax function. The model is trained consistently for 20 epochs. When using 64, 128 and 256 hidden units for the first hidden layer, the accuracy of the model on the test dataset are 86% 87% and 88% respectively. For more hidden units, we saw worse performance due to the overfitting. Meanwhile, adding more layer to the model cannot significantly improve the performance. Fig5 and Fig6 shows the performance of the MLP classifier. The running time of this model is totally 85.4s.

The performance of CNN model can reach 93% accuracy on classification as shown by Fig7. The total time of training the CNN model is 39.59s. Comparing to both the accuracy and efficiency of the model, CNN is much better than the combination of the autoencoder and MLP.

B. QMI and MAP Architecture

The QMI uses Euclidean Distance on the expect value to estimate how different the joint distribution of $(\boldsymbol{X}, \boldsymbol{Y})$ to the product of their marginal distribution. The goal is to maximize the entropy, and the update functions are as follows:

$$\hat{V_J} = \frac{1}{N^2}\sum_{p=1}^{T}\sum_{i=1}^{N_{Cp}}\sum_{j=1}^{N_{Cp}}[G_{\sigma\sqrt{2}I}(\boldsymbol{y_p(i)} - \boldsymbol{y_p(j)})] \tag{1}$$

$$\hat{V_C} = \frac{1}{N^2}\sum_{p=1}^{T}\frac{N_{Cp}}{N}\sum_{i=1}^{N_{Cp}}\sum_{j=1}^{N}[G_{\sigma\sqrt{2}I}(\boldsymbol{y_p(i)} - \boldsymbol{y(j)})] \tag{2}$$

$$\hat{V_M} = \frac{1}{N^2}\left\{\sum_{p=1}^{T}(\frac{N_{Cp}}{N})^2\right\}\sum_{i=1}^{N}\sum_{j=1}^{N}[G_{\sigma\sqrt{2}I}(\boldsymbol{y(i)} - \boldsymbol{y(j)})] \tag{3}$$

$$\text{QMI} = \hat{V_J} - 2\hat{V_C} + \hat{V_M} \tag{4}$$

$$\hat{W}_{n+1} = \hat{W}_n + \gamma\nabla_{W_n}QMI \tag{5}$$

where N_{Cp} denotes the number of samples in p^{th} class, N is the number of samples in each batch.

The maximum likelihood estimator is:

$$\hat{\theta}_{\text{MLE}}(x) = \arg\max_{\theta} f(x|\theta)$$

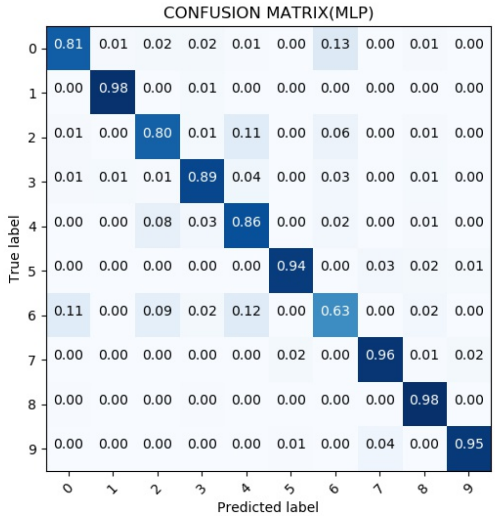

Fig. 5: Confusion matrix

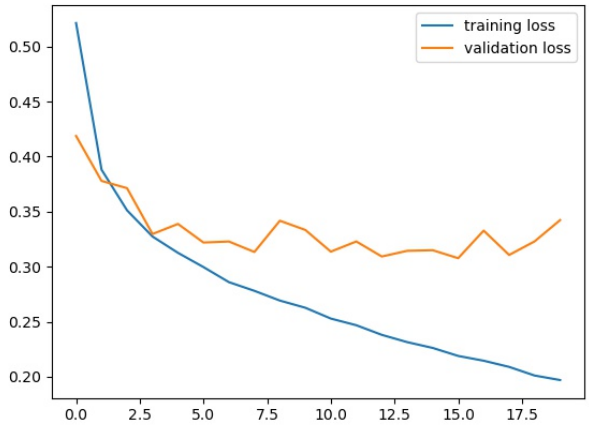

Fig. 6: Learning curve

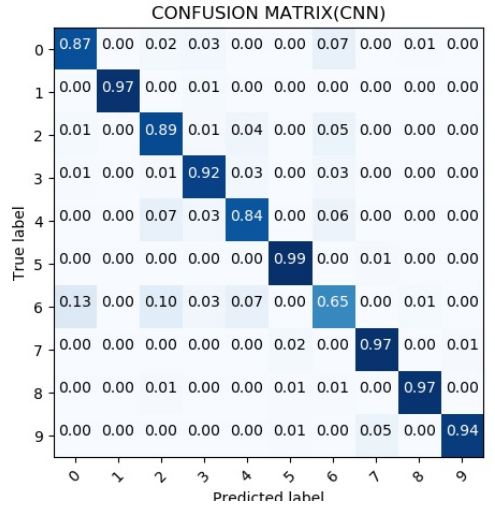

Fig. 7: Confusion matrix(CNN)

Assuming a prior distribution g over θ exists, thus the posterior distribution of θ is:

$$\begin{aligned}\hat{\theta}_{\mathrm{MAP}}(x) &= \arg\max_{\theta} f(\theta|x) \\ &= \arg\max_{\theta} \frac{f(x|\theta)g(\theta)}{f(x)} \\ &= \arg\max_{\theta} f(x|\theta)g(\theta) \end{aligned} \tag{6}$$

The MAP classifier basically compute the distance between the test sample and every training sample of each class, use Gaussian kernel to estimate the probability of each class over the test sample. Maximize the posterior probability with the prior class information.

In our experiments, we chose Gaussian distribution as the estimator. Due the number of classes in this data is 10, we trained 10 distributions in parallel. The index of the largest value of the 10 MAP models given one test data is taken as the predicted label. All of the experiments are implemented on Intel Core i7-10750H@2.60GHz and Nvidia Geforce RTX2060.

1) Training step: The mini-batch size we used in training is 200, the output dimension (dimensions of feature) is chosen as 10. After we obtained a well-trained network, the following step is classification. As we discussed before, the QMI itself cannot be utilized to separate clusters. Therefore we introduce MAP as classifier since the training data and testing data share the same prior distribution. The Algorithm 1 shows the steps of training.

Algorithm 1: MLP and MAP

Input: .A Mini-batch of data $\boldsymbol{X}$ and corresponding labels $\boldsymbol{Y}$ without shuffling.

Initialization: Initialize the model parameter with Xavier normal initializer. Set 0.3 as the value of Gaussian kernel σ

MLP stage:

for *each mini-batch* **do**

1) Pass the mini-batch through the MLP model.
2) Compute the QMI and update model parameters by performing (1) to (5).

Until

Loss $\leq \epsilon$ or early stopping criteria is satisfied

MAP stage:

1) With the well-trained model, pass the training data again.
2) Utilize the outputs as new training data of MAP model by performing (6)

Output: A well-trained MLP model based on QMI, A MAP model for classification

2) Testing step: For classification assignments, we only need to pass all the testing data through the MLP model, then passing the outputs of MLp through the MAP model. The index of the largest value produced by MAP model is the

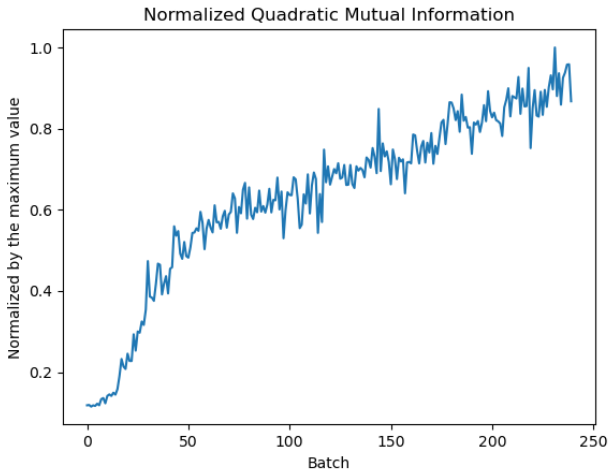

Fig. 8: Normalized QMI

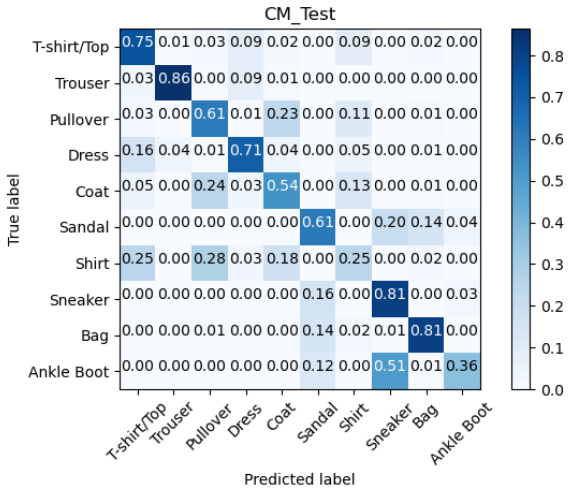

Fig. 9: Confusion matrix(MAP)

associated label.

Algorithm 2: MAP Testing

Input: Testing data $\boldsymbol{T}$, MLP and MAP models generated from training stage

for *each sample* **do**

1) Passing through the MLP model.
2) Passing the output of MLP through MAP.

Until

All samples are fully enumerated.

Output: Predicted labels on test data

3) Results: Figure 8 shows the QMI results over batches during the training. The curve is increasing since out goal is to maximize the QMI. For better representation regardless the influences caused by the constant terms, we normalized the curve by the maximum value of the sequence. It's worth mentioning that the QMI approach is really time consuming, and it took 87s to train each epoch. The accuracy of the model is 65.1%. The increasing of output dimension directly increases the training stage for the MAP. Considering the computational resources we had, we chose 4 epoch and 20 as the output dimension.

Compared with the results in experiments 1, the confusion matrix in Fig.9 implies that the Gaussian-based MAP doesn't achieve the same ideal classification results. The MLP first works as an feature generation process, then the MAP we introduce in performs as the supervised learning model. Thus, the dimension of output as well as the Gaussian kernel we implement play the two significant roles in how to extract features. However, the MAP itself can not be taken as an effective method for this classification assignment.

IV. Conclusion

Overall, In this project, two models are implemented. The first model uses autoencoder to do feature extraction and feed the features into MLP. The autoencoder has five layer which has 500, 200, 128, 200, and 500 hidden units respectively. The MLP was connected to the bottleneck layer which has 128 hidden units. The MLP has two hidden layers, the first hidden layer contains 256 hidden units, the output layer contains 10 hidden units. The second model is imply a three-layer MLP with QMI criterion. When doing prediction, the second model uses maximum a posterior methods to calculate the posterior probability of the testing data. The accuracy on classification of the first and second model are 88% and the 65.1% respectively. The CNN model can achieve 93% accuracy on the Fashion-mnist dataset. In terms of the running time, the first model takes 85.4s to train the model. CNN takes 39.59s to train the model. Due to the complex calculation, the second model takes a very long time on training. The model will spend 87s on each epoch training. QMI gives us a inspiring thinking on using information theory as a classification criterion which frees the constraint on the number of output units and the label. However,by comparing three different models, CNN model is still the dominant model on classifying the fashion-mnist dataset in terms of both accuracy and running time.

References

[1] By Michela Massi - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=80177333