Contents

1 Introduction

Learning the underlying distribution of a dataset can help with performing various tasks such as detecting out of distribution data points and performing classification when only very few labels are presents. In this assignment, we are asked to train a model using the fashion images provided by the FashionMNIST, which includes the following four datasets:

- 26000 unlabeled in-distribution data points;
- 2000 labeled in-distribution data points in five classes;
- 1052 labeled data points in six classes and out of 52 images are anomalous images classified into sixth class.
- 1052 labeled data points in six classes and out of 52 images are anomalous images classified into sixth class.

Figure 1 shows the example of normal and anomalous fashion images. The main goal of this model is to:

(i) detect out of distribution data;

(ii) give a low (ten) dimensional description of the dataset and then visualization using t-SNE;

(iii) classify the anomalous data points into the five classes.

Those five classes are trouser, sandal, sneaker, bag, and ankle boot.

(a) normal images

(b) anomalous images

Figure 1: Examples of normal and anomalous images.

2 Problem Formulation

This is a machine learning to detect the out of distribution data and classification task of the anomalous images tasks. We need to detect the anomalous images and label those images into five classes (trouser, sandal, sneaker, bag, and ankle boot). The fashion in the image is symmetry to translation, translational invariance and translation equivariance, therefore, the output of the model is symmetry to translation, translational invariance and translation equivariance. We expect the feature in the image to be symmetry. Therefore, we will use the Convolutional Variational Autoencoder (VAE) combined with the multilayer perceptron (MLP) for those tasks because Convolutional VAE is best suited for out of distribution image data detection and it is symmetric to translation, translational invariance and translation equivariance, and the MLP is suited for classification problem. Furthermore, the MLP is easily integrated into Convolution VAE model. At the same time, for this specific problem, we will use the *construction loss*, *Kullback-Leibler divergence loss* and *mean squared error loss* functions.

3 Model Formulation

3.1 Network architecture

Convolution VAE is a neural network that is designed for unsupervised learning. It consists of two parts: encoder and decoder. The encoder aims to encode input images into encoding vectors, whereas the decoder

obtains the output features back from the encoding vector. In this assignment, we built a model which is a Convolution VAE integrated with MLP to transfer the fashion image into a latent variable (i.e. a compressed version of the input fashion image) by calculating probability distribution through the encoder part and then the encoder part is designed to decompress the latent variable back to the original dimensions. The MLP network in the model is for classification task.

Figure 2 shows the architecture of the model, the model composed of Convolution VAE and MLP. The Convolution VAE network in the model is an unsupervised learning model, it learns from the unlabeled images data set and updates its parameters. The MLP network in the model is a supervised learning model, it learns from the labeled images data set and updates its parameters. The model is used to learn the features given the grayscale value of the fashion images as the input data. The proposed feature learning is defined as follows:

Input

The size of the original fashion image is 32×32 pixels in grayscale. All unlabeled and labeled images were passed through the Convolution VAE networks. All labeled images were passed through the encoder of the Convolution VAE and then the MLP network.

Encoder

The encoder is a network that encodes the input to encoded data as output. There are three blocks in the encoder part. Each block contains three layers, the first layer is the convolution layer, the second layer is the max-pooling layer and then the third layer is $ReLU$ activation function.

The first layer in the first block is the convolution layer which contains 1 input channel, 8 output channels, the kernel size is 4, the padding is 0 and stride is 1. The tensor will be sent to *MaxPool2D* layer with kernel size 0f 4 and the stride is 1. The purpose of the max pooling is to calculate the maximum value for patches of a feature map and uses it to create a down-sampled feature map. Then, we use the $ReLU$ activation function to activate the tensor before sending it to the next block. The meaning of adding the $ReLU$ activation function layer is to add the nonlinearity in the model, otherwise, the output of each layer is a linear function of the input of the upper layer. After that, the tensor is sent to the second block.

The convolution layer in the second block contains 8 input channels and 16 out channels with 4×4 kernel and 0 padding. Then, the tensor is sent to the max-pooling with 4×4 kernel and stride of 1 and then to $ReLU$ activation function layer before passing to next block.

In the third block, the convolution layer is made up of 16 input channels and 32 output channels whose padding is 0 and stride is 1. Before the tensor is flattened and sent to the fully connected layer, w use *MaxPool2D* and $ReLU$ same as in first and second blocks to extract some useful features and to activate the function.

Latent space

The VAE model presents the latent attributes as mean and standard deviation value of a probability distribution. The latent attributes describe the features of the input data. So, the size of the latent attributes represents the number of features that are extracted from the input.

The flattened tensor generated by the encoder is input into the input linear layer, and then we will output the two vectors describing mean and standard deviation of the latent state distributions. We use the *reparameterization trick* so that the backpropagation is possible. The *reparameterization trick* suggests that we randomly sample ϵ from a unit Gaussian, and then shift the randomly sampled ϵ by the latent distribution's mean μ and scale it by the latent distribution standard deviation σ, which is $z = \sigma + \mu \cdot \epsilon$. This z tensor will be sent to the input layer of the decoder.

Decoder

The decoder is a network that decodes the latent attributes to output data. The networks generate a sample from each latent state distribution. Then, the sample data are decoded to the output. There are three blocks in the decoder. Each block contains three layers: the first layer is the transposed convolution layer, which is a deconvolution operation which attempts to reverse the effects of a convolution; the second layer is the

up-sampling layer by using the max-unpooling; the third layer is the *ReLU* activation function. The blocks in the decoder are some kind of mirror of the blocks in the encoder.

First, the sample tensor from the distribution is reshaped into the input dimensions for the transposed convolution layer and then *ReLU* function is applied for nonlinearity activation. The first layer in the first block is the transposed convolution layer which contains 32 input channel, 16 output channels, the kernel size is 4, the padding is 0 and stride is 1. Next, the tensor is sent to *MaxUnpool2D* layer with kernel size of and the stride is 1. The meaning of *MaxUnpool2D* layer is to up-sampling the feature map and it is used to revert the effect of the max pooling operation. Then, we use the *ReLU* activation function to activate the tensor before sending it to the second block.

In the second block, the convolution layer contains 16 input channels and 8 out channels with 4×4 kernel and 0 padding. Then, the tensor is sent to the max-unpooling with 4×4 kernel and stride of 1 and then to *ReLU* activation function layer before passing to next block.

In the last block of the decoder, the convolution layer is made up of 8 input channels and 1 output channel whose padding is 0 and stride is 1. Then, we use the *MaxUnpool2D* layer to up-sampling the feature map and *ReLU* activation function before the output layer.

Classifier MLP

The classifier MLP part of the model is the linear layers. The tensor after flattening and application of *ReLU* activation function is input into the input layer which has 10 neurons. After that, the *ReLU* is applied to include the nonlinearity in the classifier MLP model. Next, we use the *dropout* layer which is randomly sets input neurons to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. There are two hidden layers which contain 256 neurons and 64 neurons, respectively. In each hidden layer, the *ReLU* activation function and *dropout* layer are applied. Finally, the output layer has 5 neurons because there are 5 classes (trouser, sandal, sneaker, bag, and ankle boot).

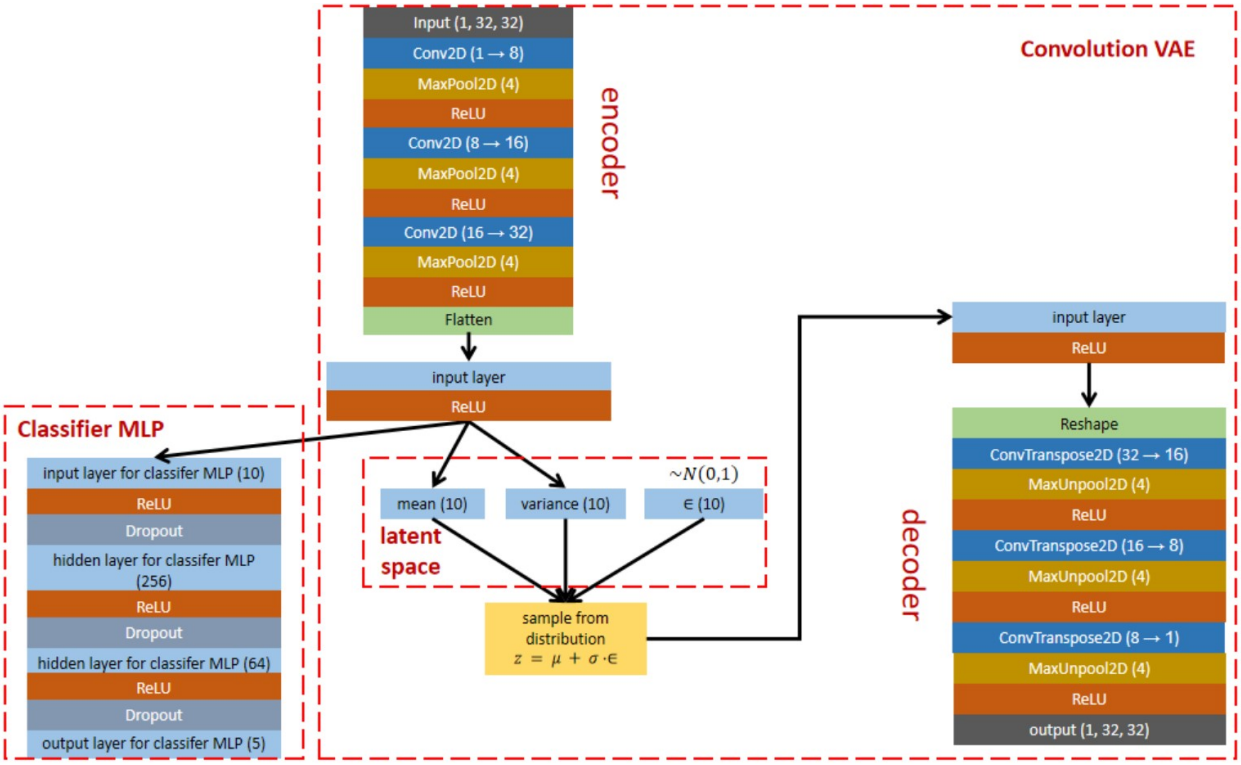

Figure 2: The architecture of the model composed of Convolution VAE and MLP.

3.2 Loss functions

The loss function is used to estimate the gap between the model's output and the true value and guide the optimization of the model. In this assignment, for the model with Convolution VAE integrated with the classifier MLP, four loss functions are required. They are reconstruction loss, Kullback-Leibler divergence loss and the cross entropy loss.

3.2.1 Reconstruction loss

The reconstruction loss enables the distribution to correctly describe the input, by focusing only on minimizing the reconstruction loss, the network learns very narrow distributions, akin to discrete latent attributes. The reconstruction loss defined in this assignment is as follow

$$\text{reconstruction loss} = \|x - \hat{x}\|^2 = \|x - \text{decoder}(\text{encoder}(x))\|^2 \tag{1}$$

where x is the input of the encoder and $\hat{x}$ is the output of the decoder. The output of the encoder is written as encoder(x) and the output of the decoder is decoder(encoder(x)).

3.2.2 Kullback-Leibler divergence loss

The Kullback-Leibler (KL) divergence loss prevents the network from learning narrow distributions and tries to bring the distribution closer to a unit normal distribution, i.e. it is a regularization term on the latent layer that tends to regularize the organization of the latent space by making the distributions returned by the encoder close to a standard normal distribution. The KL divergence loss is defined as

$$\text{KL divergence loss} = \sum_{i=1}^{n}(\sigma_i^2 + \mu_i^2 - \log\sigma_i - 1) \tag{2}$$

where μ is the latent distribution's mean and σ is the latent distribution standard deviation. The KL divergence loss is minimized when the μ_i approaching zero and σ_i approaching 1.

3.2.3 Cross entropy loss

The cross entropy loss is added to the total loss of the model for adjusting the parameters during training when the labeled data set is fed into the model. The cross entropy loss is computed in the classifier MLP part for mainly adjusting its weights. The cross entropy is defined as

$$\text{cross entropy loss} = -\sum_{i=1}^{n} t_i \log p_i \tag{3}$$

where n is the total number of classes, t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

3.3 Accuracy

Accuracy is a metric used in our classification problem to tell the percentage of accurate predictions. We calculate it by dividing the number of correct predictions by the total number of predictions for all 5 classes classified by the classifier MLP with labeled test data set size of 2000.

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}} \tag{4}$$

4 Implementation and training

4.1 Data preprocessing

In Section 1, we have mentioned there are four data sets. Firstly, since all images are in grayscale between 0 and 255, we normalize all the images by dividing 255 such that all the pixels are range from 0.0 to 1.0, with a value of 0.0 representing white, a value of 1.0 representing black, and in between values representing

gradually darkening shades of grey. In this way, the numbers will be small and the computation becomes easier and faster.

After that, the given labeled data sets are label in one-hot vector with five classes. Therefore, we transformed each one-hot vector into an integer between 0 and 4. In this way, the class of each data point can be referred to the Fashion class dictionary:

$$\text{Fashion class dictionary} = \{0 : \text{"Trouser"}, 1 : \text{"Sandal"}, 2 : \text{"Sneaker"}, 3 : \text{"Bag"}, 4 : \text{"Ankle boot"}, 5 : \text{"Anomaly"}\}.$$

The sixth class (5:"anomaly") is for anomalous data points. Next, the 26000 labeled in-distribution data points is shuffled and then split into train data set and validation data set, with ratio of 80:20 by using the data sampler *SubsetRandomSampler*, i.e. 80% of the data set is reserved for training and 20% of the data set is reserved for validation. We performed the same operation for 2000 labeled in-distribution data points. The unlabeled data set is used for training and validation of the Convolution VAE part of the model because it is an unsupervised learning model. Whereas, the labeled data set is used to train and validate mainly the classifier MLP because it is a supervised learning model.

Finally, We combined the third and fourth into one data set, i.e. this combined data set has 2104 labeled data points in six classes and out of 104 images are anomalous images classified into sixth class. Then, we split this data set into normal data set (2000 data points) and anomalous data set (104 data points). These normal and anomalous data sets are used for testing because it contains the anomalous images.

4.2 Training

In the training section, we compute the training loss and validation loss for each epoch. The training loss is used to update the parameters after every backpropagation. On the other hand, in the validation process, we compute the validation loss for each epoch instead of compute the gradient or update parameters by using back propagation. By comparing both training loss and validation loss, we can prevent the overfitting in our model during training.

We need to use all the data in the data loader in each epoch. By iterating the train data loader for the train data set, we obtain the train data points, put them in the model we built before, and we can get embeddings in the latent space through the encoder. After that, these embeddings will be used to compute the latent distribution's mean μ and the latent distribution's standard deviation σ. Next, these μ and σ will be used to calculate the sample z by using the *reparameterization trick* as explained in section 3.1. This z will be reshaped and then feed into the decoder. At the end, we will get the output which has the same dimensions as input, and then the reconstruction loss can be computed. The KL divergence loss can be computed by using the μ, σ and Equation 2. For computing the cross entropy loss, the embeddings in the latent space will be input into the input layer of the classifier MLP and then the output of the classifier MLP and the label of the labeled data set will be used for computing the cross entropy loss. We need to zero out the gradient every time before starting the backpropagation for computing the current gradient. Otherwise, Pytorch will accumulate the gradient calculated from previous iterations. After that, the parameters will be updated after computing the gradients through backpropagation. Next, we compute the average train loss by dividing the length of the data loader. The same procedure can be applied in the validation part. Therefore, we can get average train and validation losses after every epoch.

As for the optimizer, we choose Adam as our optimizer. This optimization algorithm is a further extension of stochastic gradient descent (SGD) to update network weights during training. Furthermore, Adam optimizer does not need large space, and it requires less memory space which is very efficient. Intuitively, it is a combination of the gradient descent with momentum algorithm and the RMSprop algorithm. Adam optimizer is used to accelerate the gradient descent algorithm by taking into consideration the exponentially weighted average of the gradients. Using averages makes the algorithm converge towards the minima at a faster pace. Therefore, we choose the Adam optimizer and set the learning rate as 0.001 to compute the gradient and update the parameters.

4.3 Evaluation

Firstly, we calculate the Evidence Lower Bound (ELBO) values for the test data set and re-constructed data points. The ELBO value of a single data point is presented in equation 5

$$\log p(\mathbf{x}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x} \mid \mathbf{z})] - KL(q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z})) \tag{5}$$

Then, we visualize the distribution by histogram and density of these two categories (normal and anomalous data points). To better detect the anomaly, we plot the Receiver Operating Characteristic (ROC) curve to select a good threshold. More specifically, the ROC curve plots the True Positive Rate and the False Positive Rate on a plane, while the deciding thresholds varies. This means that every point of the curve corresponds to a specific threshold value. By comparing the corresponding true positive rate and false negative rate, we select an appropriate threshold.

Secondly, we plot a Precision-Recall curve (PRC) and calculate AUC (Area Under Curve) to evaluate the performance of the Convolution VAE model in anomaly detection.

5 Experiments and Results

This section discusses the setup of the experiments. We focus on the presentation and interpretation of the performance of both training and test sets.

5.1 Experimental setup

In this experiment, we use the data set contains 26000 unlabeled in-distribution data points, 80% of the data set is divided into training set and 20% is a validation set. We performed the same operation for the data set of 2000 labeled in-distribution data points. These two data sets are used for training and validating the model. The loss functions from equation 1, 2 and 3 are used to compute the loss and then for optimization. We use Adam as our optimizer and train the model by iterating 50 epochs.

For testing the model, we use the data set which contain 2104 labeled data points where out of 104 data points are anomalous images. Therefore, we divided the data set into 2000 normal labeled images and 104 anomalous unlabeled images. Both data sets are used to detect out of distribution data. The 2000 normal labeled images are used to generate the low (we used 10 dimensions in our model) dimensional description of the fashion data set and then we used the dimensionality reduction technique t-Distributed Stochastic Neighbor Embedding (t-SNE) for visualizing these 5 class data points in two-dimensional view. We use the *TSNE* function from scikit-learn to perform the dimensionality reduction with perplexity of 2000, automatic learning rate and 10000 number of iterations. Besides that, the 2000 normal labeled images are also used for classification accuracy testing of the classifier MLP. Finally, the 104 anomalous unlabeled images are used for classification into one of the five classes by using the classifier MLP. All three main goals as mentioned in section 1 are achieved by using this model, Convolution VAE integrated with classifier MLP, see Figure 2.

5.2 Results and analysis

5.2.1 Results

Figure 3 shows the decrease of training loss and validation loss with the iteration of 50 epochs. As we can see from Figure 3, our model fits well throughout the epochs, and overfitting does not seem to occur.

The classification accuracy of the model on the in-distribution labeled test data set is 95.25%. Figure 4 shows some examples of the classification. We can see that the shoe type images are considered difficult to classify.

The visualization of the normal and anomalous data points distribution through histogram and density are shown in Figure 5 and Figure 6, respectively. The ROC curve and the PRC of our implementation are shown in Figure 7 and Figure 8, respectively. The t-SNE plot to visualize the test data points of all five classes in two-dimension is shown in Figure 9. Table 1 tabulated the number of predictions for each five class of the anomalous data points and Figure 10 shows some example of the classified anomalous images.

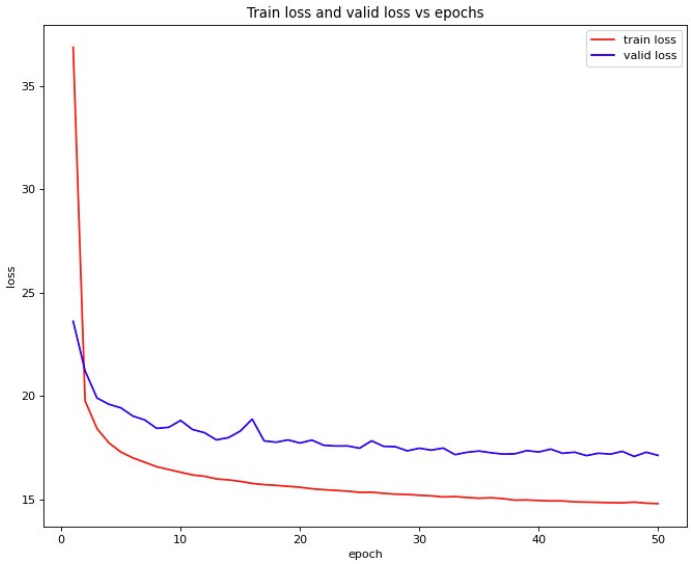

Figure 3: Training and validation losses versus the epochs.

Figure 4: Example of the classified fashion images. The above row is the images with correctly classified and below are the misclassified images.

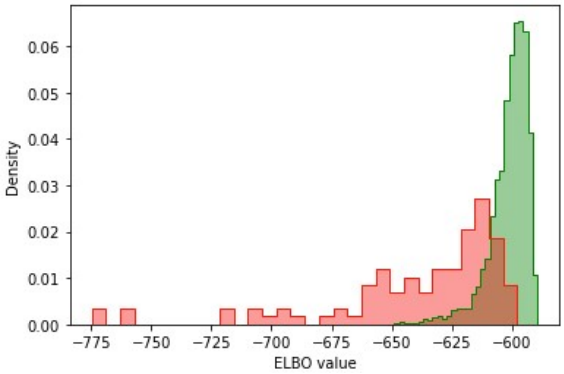

Figure 5: ELBO values Histogram of normal(green) and abnormal(red).

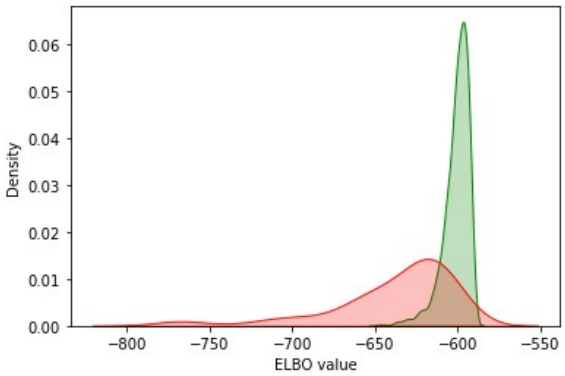

Figure 6: ELBO values density of normal(green) and abnormal(red).

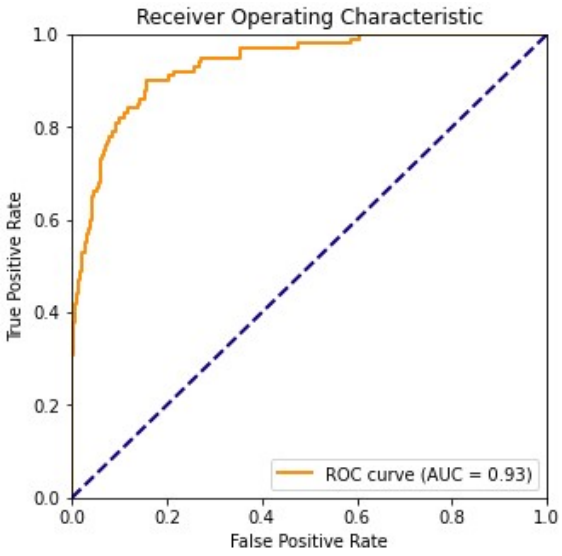

Figure 7: The ROC curve (orange line) of the proposed Convolution VAE model.

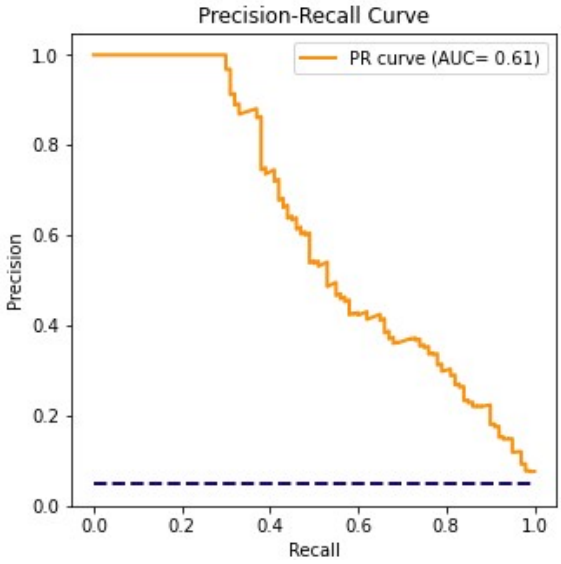

Figure 8: The PRC curve (orange line) of the proposed Convolution VAE model.

	number of predictions
trouser	30
sandal	5
sneaker	5
bag	58
ankle boot	4

Table 1: Number of predictions for all five classes.

5.2.2 Results analysis

in-distribution and out of distribution data

Using Equation 5, we can calculate the ELBO values for both normal and anomalous data points, and visualize the distribution by histogram and density plots of these two categories as shown in Figure 5 and Figure 6, respectively. We can observe that the anomalies are of lower ELBO, and the distribution of normal and anomalous data points are clearly separated, which illustrated the feasibility of our model. Based on the histogram, we can choose a proper threshold, namely the corresponding ELBO value of the intersection point between the normal distribution (green) and anomalous distribution (red), the threshold we have selected is -612.0, then we get at least 89.92% accuracy.

As mentioned in Section 4.3, an ROC curve is created by plotting the True Positive Rate against the False Positive Rate at various threshold settings. The ROC curve is shown in Figure 7. Ideally, an optimal identifier will achieve a square where the vertex is at (0,1), whereas a random classifier will result in a diagonal line (dashed line in Figure 7). The smaller the values on the x-axis of the plot, the lower False Positive and the higher True Negative, and the larger the values on the y-axis of the plot, the higher True Positive and the lower False Negative. ROC curves are widely used because we can simultaneously see the performance in general for different thresholds, and the Area Under the Curve (AUC) can be used as a summary of the model skill. Concretely, the larger the AUC, the better the overall performance of the model. For our model, we obtain $AUC_{ROC} = 0.93$, the optimal AUC is 1.00. This demonstrates that our model is well effective according to this $AUC_{ROC} = 0.93$ value.

Similarly, we can generate a precision-recall curve (PRC) by plotting precision against recall at various threshold configurations. The PRC is shown in Figure 8. An ideal identifier will achieve a horizontal line at

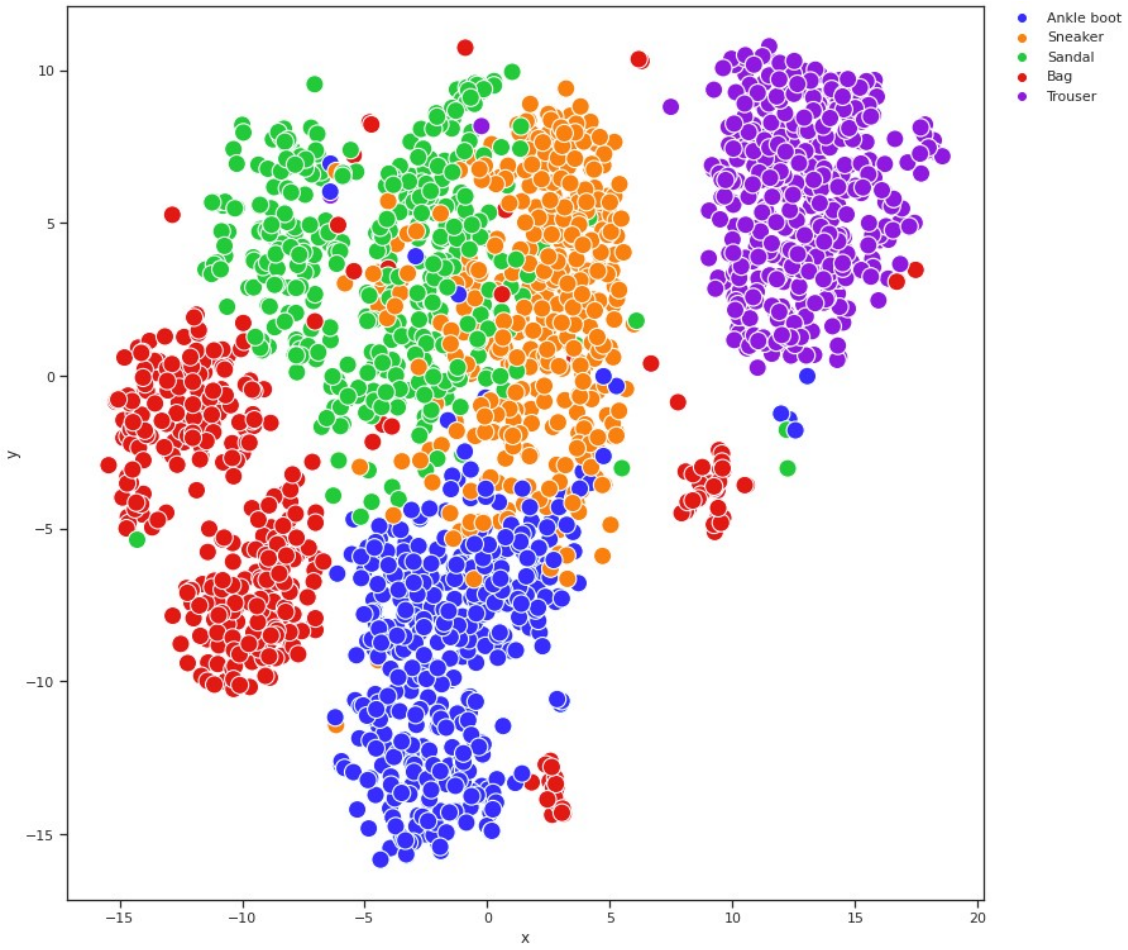

Figure 9: The t-SNE plot to visualize the data points in five classes in two-dimensional view.

$y = 1$, whereas a no-skill classifier will result in a horizontal line at about $y = 0.05$ (the horizontal dashed line in Figure 8). For our model, we obtain $AUC_{PRC} = 0.61$ which is larger than 0.5, the optimal AUC is 1.0.

low dimensional description of the data set

Figure 9 shows the t-Distributed Stochastic Neighbor Embedding (t-SNE) plot of the low dimensional description of the normal labeled data set. t-SNE is an unsupervised, non-linear technique primarily used for data exploration and visualizing high-dimensional data. In other words, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space. It was developed by Laurens van der Maatens and Geoffrey Hinton in 2008. In our experiment, we want to visualize the distribution of normal labeled data set in order to evaluate the performance of the KL divergence loss (Equation 2). This plot shows us how each class are distributed across the latent space. It is noteworthy that the fashion that belong to the same class seem to be generally clustered around each other, although there is a messy region in the middle. This is a reasonable result: while we would expect trousers to be fairly easy to distinguish from, say, bags. Fashions like ankle boot, sneaker and sandal might look very similar, and hence appear mixed as a lump in the fuzzy region in the middle. Additionally, we observed that there is a clear gap between the trousers cluster (purple) and other clusters. This indicates that we might need to give a higher weight to the KL divergence loss with a parameter (for example, $\beta > 1$), encouraging the model to learn broader distributions.

classification of the anomalous data points

Finally, we investigate the performance of the classifier MLP part in the model. Table 1 tabulates the number of predictions for each fashion class for all anomalous images. We observed that the classifier MLP has high

Figure 10: Examples of the classification of the anomalous data points.

preference to predict an anomalous image as a trouser or bag, although we obtain high accuracy (95.25%) of prediction for the labeled normal images. This might be caused by corrupted part in an anomalous image, i.e. the rectangle black corrupted region an anomalous image has significant impact on the accuracy of the classification, see Figure 10. In conclusion, our classifier MLP part in the model does not perform well on the anomalous image classification.

6 Conclusion

In this report, we first formalize this task as detection of the out of distribution and classification task of the anomalous images' problem. To deal with this, we argue that the generative models are proper choices. Next, we provide a detailed model architecture description on our Convolution VAE integrated with classifier MLP model. The Convolution VAE part in the model is used for the out of distribution detection, and the classifier MLP is used for classification of the anomalous images. Concretely, we choose a VAE model as the framework, and convolution-based is used to effectively encode and decode the normal data represented by a grayscale image. Then, according to the ELBO distribution comparison, we choose a threshold of -612.0 and get 89.92% accuracy. We also perform another two evaluations powered by ROC curves and PRC. Both results ($AUC_{ROC} = 0.93$ and $AUC_{PRC} = 0.61$) shows the effectiveness of our model, both AUC values are higher than 0.5. The t-SNE plot is also plotted, and the plot clearly shows the clustering of each fashion class. Bag and trouser clusters are clearly separated. because they are visually distinct. However, some data points of the ankle boot, sneaker, and the sandal mixed up in the middle of the plot because the images of these three classes are visually similar. Finally, we classified the anomalous images by using the classifier MLP part in the model. The classification result is not satisfied because the classifier MLP has higher tendency to classify an anomalous image as trouser and bag. This misclassification might due to the rectangle black corrupted region appear in the anomalous images.

7 Discussion

In this section, we will discuss the following research questions:

i In what situations would you be able to use transfer learning to aid you in the given tasks?

ii How would you use transfer learning to perform the given tasks?

iii In what situations would you not be able to use transfer learning / when could transfer learning be

detrimental to performance on (some of) the given tasks?

For the research question i, we are able to use the transfer learning to aid us in the given tasks if the pre-trained model is trained in the similar domain and task. For example, the pre-trained model is trained in the fashion domain and the task is classification of the fashion images. This is feasible, because utilization of the pre-trained model (for example, VGG-16 which has been trained with dataset of over 14 million images belonging to 1000 classes) allow us to leverage the already existing labeled data of some related task or domain. Leveraging the pre-trained VGG-16 model might improve the performance of the anomalous images classification and the low dimensional description of the data set because we expect the important features of normal images have been captured.

For the research question ii, we would use the pre-trained convolution neural network (CNN), e.g. VGG-16, which has 13 convolutional layers, 5 pooling layers and 3 dense layers in the encoder of our convolution VAE model. We then remove the 3 dense layers in the VGG-16 and replaced them with our dense layers in the latent space, i.e. the VGG-16 is performed as feature extraction of the fashion images and then the extracted features are flattened and then fed into the dense layer in the latent space. We need to adjust hyperparameters of the first convolution layer in the VGG-16 so that it matches the dimensions of our data set. Next, we model the decoder similar to the VGG-16 architecture (note that the decoder architecture is the mirror image of the encoder. This is not a requirement but it is typically the case). During training, we unfreeze the pre-trained VGG-16 model and fine tune the parameters with small learning rate in order to ensure that we do not unlearn the previously acquired knowledge. This simple approach has been shown to achieve astounding results on an array of vision tasks as well as tasks that rely on visual input such as image captioning [1].

For the research question iii, the transfer learning will not work when the high-level features learned by the bottom layers are not sufficient to differentiate the classes in our problem. For example, the pre-trained model is good at identifying sandal from sneaker and ankle boot, but not between the sneaker and ankle boot. In this case, we can use the low-level features (of the pre-trained model) instead of the high-level features. Thus, we will have to retrain more layers of the pre-trained model or use features from earlier layers. Otherwise, the transfer learning could be detrimental to the performance of anomalous images classification and low dimensional description of the data set. In addition, the transfer learning will not work if the source domain and tasks of the pre-trained network are considerably different from our target domain and tasks.

References

[1] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. 2014. cite arxiv:1403.6382Comment: version 3 revisions: 1)Added results using feature processing and data augmentation 2)Referring to most recent efforts of using CNN for different visual recognition tasks 3) updated text/caption.