2AMM10 2021-2022 Assignment 4: Group 22

Ruichen Hu(1674544) Simin Sun(1692933) Siyue Chen(1657402)

June 20, 2022

1 Problem Formulation

In this assignment, we need to design a model that is able to ssolve the following three problems:

1. Detect out of distribution data;
2. Give a low dimensional description of the dataset in terms of the class distribution
3. Classify the data into five classes

For a classification task like the third problem, both discriminative and generative models works. A discriminative model calculates posterior probability which is simpler, while a generative model calculates joint probability which can handle latent variables. For the second problem, we need to generate new images from a pre-specified class without using existing images as the basis. This is naturally a problem that a generative model can solve. For the first problem, detection of out-of-distribution (OOD) data is a common problem for deep generative models, such as variational auto-encoders (VAEs) [1]. Thus, we try to implement a generative model, in our case, a conditional variational autoencoder (CVAE).

2 Model Formulation

In terms of the problem formulation, as discuss in the previous chapter, we need to adopt a generative model in order to solve all three tasks with one model. In terms of data, we have a large dataset without labels and three relatively small datasets with labels. Thus unsupervised learning would be a good option. Convolutional network is also considered as it is invariant and equivariant to spatial image transformations due to the convolutional architecture. Considering all these aspect, we implement a CVAE. The architecture of the model is shown in 1

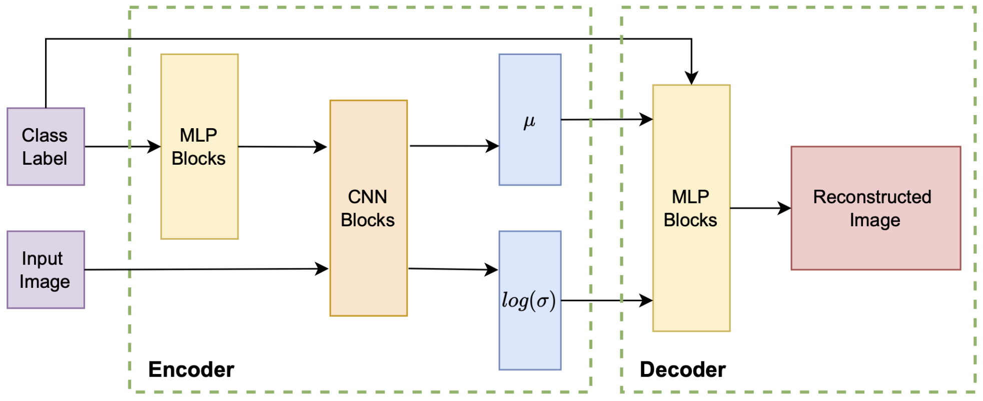

Figure 1: CVAE Architecture

In generative models, we can use neural networks to approximate the distribution of inputs: $x \sim P_\theta(x)$, where θ is the parameter determined in the training. We also want to find the joint distribution between the data x and the potential variable z. We have $P_\theta(x) = \int P_\theta(x,z)dz = P_\theta(x|z)P(z)dz = P_\theta(z|x)P(x)dz$. To simplify the problem, we can introduce the VAE, which takes high dimensional input data and compresses it into a smaller representation. The distribution is regularised to make sure that the latent space has good properties to generate new data. The encoder introduces $Q_\phi(z|x) \approx P_\theta(z|x)$, which generates the latent vector z from the input x and can be approximate by optimizing . To determine the distance between $Q_\phi(z|x)$ and $P_\theta(x|z)$, we use Kullback-Leibler (KL) divergence. Combine it with Bayes' theorem, we have

$$\log P_\theta(x) - D_{KL}(Q_\phi(z|x)||P_\theta(z|x)) = E_{z\sim Q}(\log P_\theta(x|z)] - D_{KL}(Q_\phi(z|x))||[P_\theta(z)) \tag{1}$$

Usually, Q_ϕ and $P_\theta(z)$ follow normal distribution. Thus we can simplify KL as:

$$-D_{KL}(Q_\phi(z|x)||P_\theta(z)) = \frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma_j)^2 - (\mu_j)^2 - (\sigma_j)^2) \tag{2}$$

However, the VAE has no control over which class will be generated because the encoder models the latent variable z directly based on X. CVAE can solve this problem by including a condition for the target label. To do so, we reconstruct equation 1 with the condition c as:

$$\log P_\theta(x|c) - D_{KL}(Q_\phi(z|x,c)||P_\theta(z|x,c)) = E_{z\sim Q}(\log P_\theta(x|z,c)] - D_{KL}(Q_\phi(z|x,c))||[P_\theta(z|c)) \tag{3}$$

Now, the real latent variable is distributed under $P\theta(z|c)$ which follows a conditional probability distribution. We will minimize the loss from reconstruction loss and KL loss. We could now generate data with a specific attribute.

We still need to be careful about the way we sample from the distribution returned by the encoder during training. The sampling process has to be expressed in a way that allows the error to be backpropagated through the network. We can adopt the reparametrisation trick. It takes the parameters of $q(z|x)$ as input, and returns a sample from $q(z|x)$. This is done by first sampling from a standard Gaussian (Normal) distribution, and then applying the proper transformation to obtain a sample from $q(z|x)$ with the given mean and variance. [2]

3 Implementation

3.1 Data Handling

The full dataset is divided into four subsets:

tag	**data shape**	**label shape**
unlabeled_data	(26000, 1, 32, 32)	unlabeled
labeled_data	(2000, 1, 32, 32)	(2000, 5)
representative_set_1	(1052, 1, 32, 32)	(1052, 6)
representative_set_2	(1052, 1, 32, 32)	(1052, 6)

Table 1: Full Dataset

The first thing we do is to normalize the image data from the range $[0, 255]$ to $[0, 1]$ for convenience. The labels are already described in One-Hot vectors, thus they remain what they are. But we need to add a sixth label to 'labeled_data' for the third task.

The first dataset with 26000 unlabeled data is used only to visualize the results of the trained model in classification task. We construct a dataloader 'vis_dataloader'.

The second dataset with 2000 labeled data is used to train the CVAE. We construct a dataloader 'train_cls_dataloader'

The third dataset consisting 1052 labeled anomalous and nonanomalous data is combined with the trained model to get the ROC curve to determine the threshold of anomaly detection. We construct one dataloader 'val_in_dis_dataloader' for in-distribution data and one 'val_out_dis_dataloader' for anomalous data.

The fourth dataset consisting 1052 labeled anomalous and nonanomalous data is to test whether the threshold works for unseen data. We construct one dataloader 'test_in_dis_dataloader' for in-distribution data and one 'test_out_dis_dataloader' for anomalous data.

We also construct a dataloader 'classification_dataloader', which consists of all labeled data in the third dataset and the fourth dataset, to test the performance of the trained model on classifying unseen data and obtain the test accuracy.

3.2 Model Implementation

The structure of the model is shown in Figure 2.

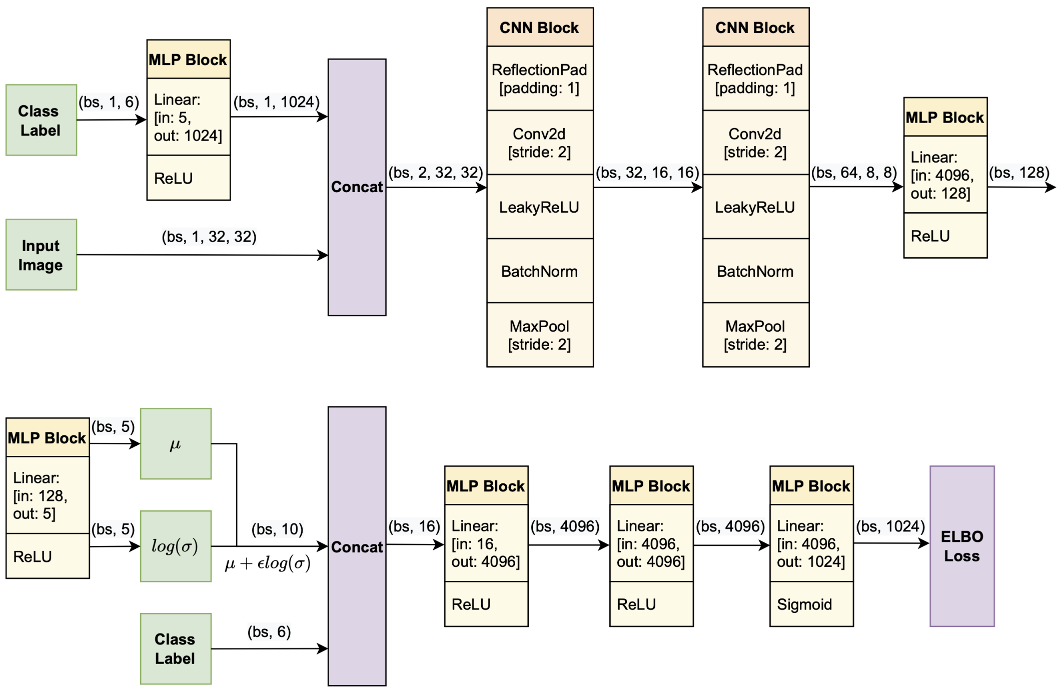

Figure 2: Model Structure

For the decoder, we take the image data and labels from 'labeled_data' as input. The label will pass through a simple MLP in the model that takes a one-hot encoded label as the input, and outputs a tensor with the same size as the images.Then the concatenated input will be put into two 2D convolutional layers. The introduction of CNN can improve the perceptual quality and decrease the number of parameters of the network compared to MLP [3]. After the flatten layer, we use two dense layers to get the μ and $\log \sigma$ parameters of the Gaussian distribution. Finally, we use a lambda function to implement the reparameterization trick.

For the encoder, it takes the two inputs: two-dimensional result from encoder, and the class labels. We concatenate the two inputs and pass it to a dense layer and a reshape layer. Then we have two hidden layers and an output layer to get the reconstruted image.

To apply the trained CVAE to classification tasks, we first generate for each mode/class 100 images by feeding the labels into the network. Then, we compare each image, which is going to be classified, with the generated images and calculate the average cosine similarity with respect to each mode.

4 Experiments and evaluation

4.1 Training

In the experiments, we constantly adjust the hyperparameters like: number of epochs, the learning rate, and size of hidden layer, etc. We have an optimal set of parameters that makes our model works fine. The learning rate is 0.001, the hidden dimension is 128 and the total epochs is 50 since the loss converges and changes slightly even if we increase the number of epochs. The reconstruction loss decrease as the epoch increase while the KL loss is the opposite. Adding these two parts together, we get negative ELBO, which is to be minimized. As the average loss is shown in Figure 3, this number indeed decreases during the training process. It's worth noting that the it decreases significantly during the first five epochs, then gradually dropped after that.

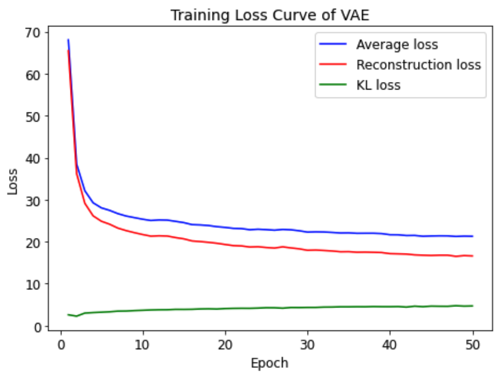

Figure 3: Loss and accuracy of C

4.2 Evaluating

4.2.1 Image Generation

To evaluate the performance of model on generating images by given labels, we plots some generated images by feeding a specific label. That means, there is no true image fed into the network. We generate a image purely besed on the latent variables and the target label. The results are shown as Figure 4. The images were all generated properly. The created photos' only drawback is their lower resolution and fuzzy quality. This might be mostly as a result of the use of VAE. Besides, we use MLP as the decoder rather than deconvolution. Deconvolution is something we tried, but it results in a large loss value, and we don't have enough time to make more improvements.

The Figure 4 also indicates that our model can generate a distribution of each class given a single one-hot class label, such that we can generate images belonging to a specific class. The latent variable, combined with the class label vector, can be regarded as a low dimensional description for the provided dataset.

4.2.2 Anomalies Detection

To evaluate the performance of the trained CVAE on anomaly detection, we plot the distribution of the reconstruciton loss and ELBO with respect to the class of the data, i.e., anumalous and nonanomalous as shown in Figure 5. This is done with the first dataloaders we mentioned above.

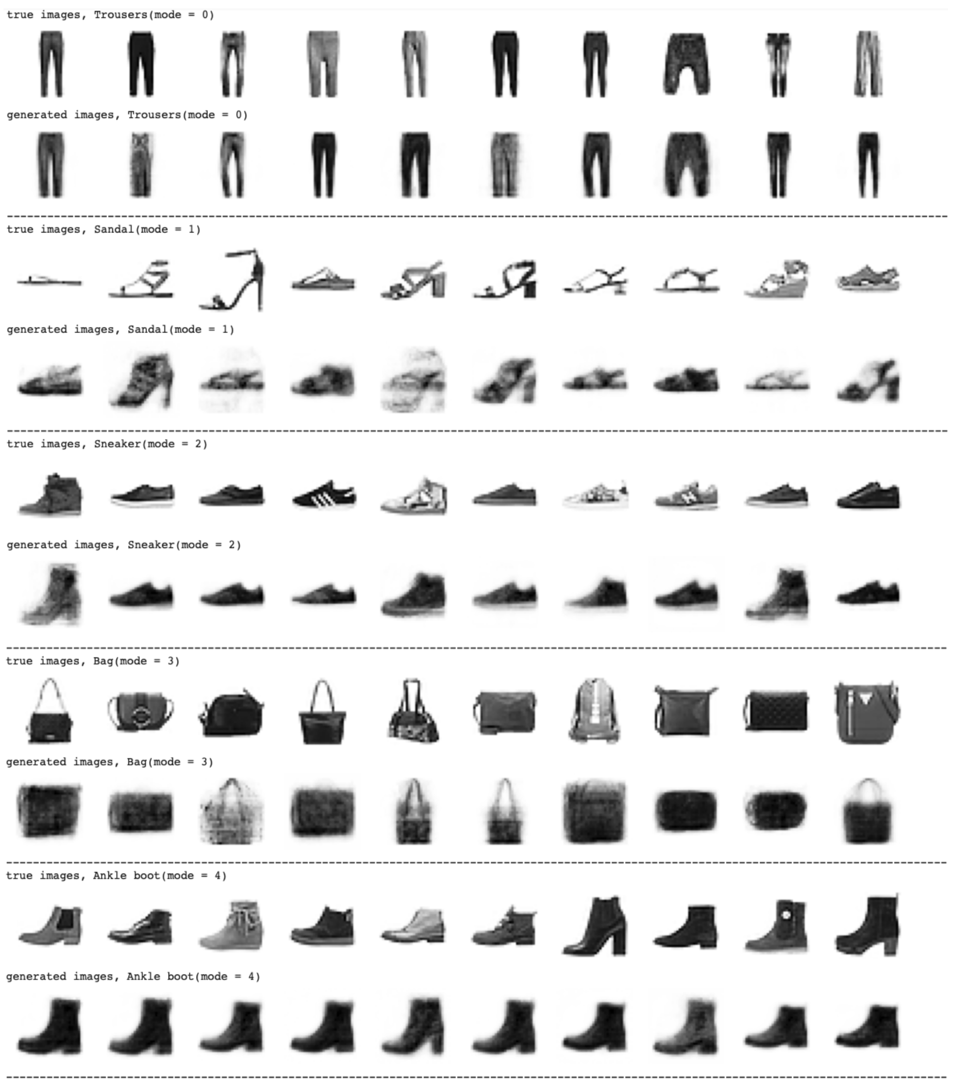
true images, Trousers(mode = 0)
generated images, Trousers(mode = 0)
true images, Sandal(mode = 1)
generated images, Sandal(mode = 1)
true images, Sneaker(mode = 2)
generated images, Sneaker(mode = 2)
true images, Bag(mode = 3)
generated images, Bag(mode = 3)
true images, Ankle boot(mode = 4)
generated images, Ankle boot(mode = 4)

Figure 4: Image Generation giving labels

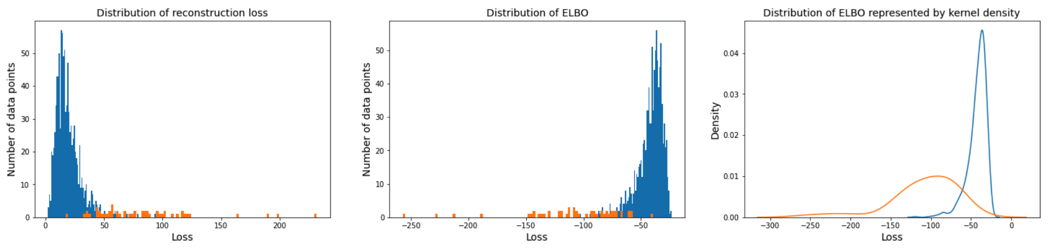

Figure 5: Distribution of the reconstruciton loss and ELBO

Then we use ROC as an assessment approach, which results in higher accuracy. To find the threshold, we try to balance between the False Positive Rate (FPR) and True Positive Rate (TPR) as shown in Figure 6. The selected threshold is 0.14100000000000001 and corresponding roc_auc=0.9782307692307692 and pr_auc=0.833888483426052. We note that when the false positive rate is low, our model can achieve a rather high true positive rate. Our model's AUC, or Area Under the ROC Curve, is fairly close to 1, indicating that the model is doing well.

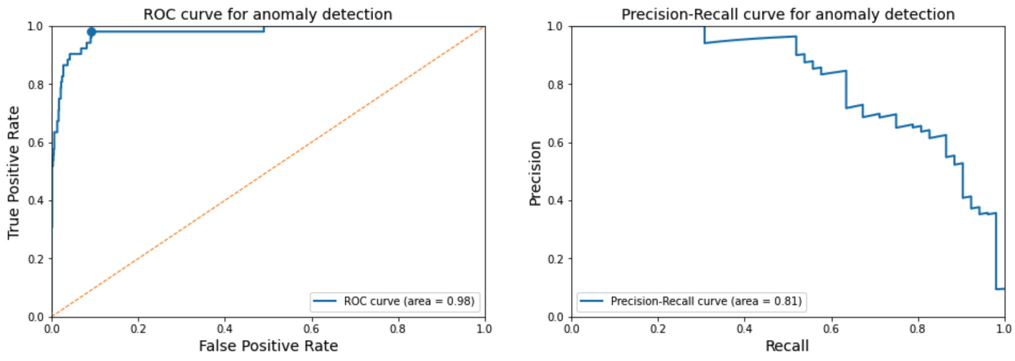

Figure 6: ROC and Precision-Recall curve for anomaly detection

As shown in Figure 6, we draw a ROC curve with the first anomaly-containg dataset and find the threshold which corresponds to the point closest to (0, 1) on the ROC curve. We evaluate whether this threshold works by looking at the last two dataloaders, and the accuracies of detect anomaly and unseen anomaly are 0.9401140684410646 and 0.8365019011406845 respectively. We ran it several times, and the thresholds for each iteration are around 0.14, with unseen anomaly detection accuracy ranging from 0.8 to 0.9.

4.2.3 Image Classifier

To evaluate the performance of the model on classification, we first test it on test data and get corresponding accuracy. The test data for classifier is extracted from the last two datasets, each of which contains 1052 data points. We only need the in-distribution data, thus, the number of the extracted data points is 2000. These data points are unseen during training. The accuracy of our model is ...., and additionally we also plot predicted images to investigate some cases where your model fails.

To assess how well the model performs in terms of classification, we test it on test data and obtain the corresponding accuracy. The last two datasets, which each have 1052 data points, are used to extract the test data for the classifier. 2000 data points were chosen because we only require the in-distribution data. During training, these data points are hidden and our model's accuracy is.... We also plot anticipated images labels to look into some scenarios in which our model success and fails.

It's worth noting that, based on the Figure 7 we printed below, our model is unable to predict all of the sandal's images Figure 8. The first reason could be that the images we generate are vague and lack some feature information. The second reason might be because it resembles sneakers and ankle boots in some ways and is therefore susceptible to classification errors.

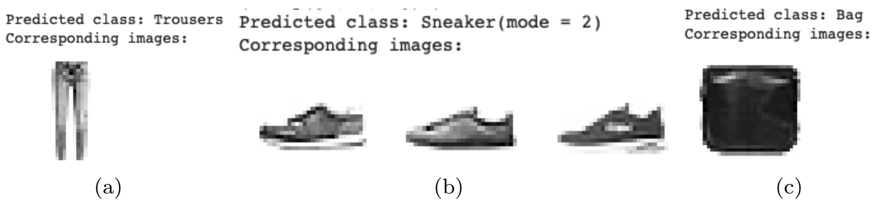

Figure 7: Correct Prediction

```
(array([1, 4, 7, 8, 9]),)
Predicted class: Ankle boot(mode = 4)
Corresponding images:
```

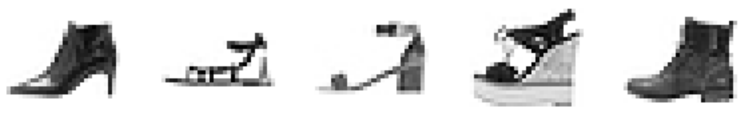

Figure 8: Fail Cases: sandals misclassify as Ankle boot

5 Conclusion

We implement a CVAE model to solve the given three tasks which perform well in the first two tasks. For the first task, we achieve a AUC of 0.988 in terms of ROC. With the selected threshold 0.125, it is able to detect anomalies. For the second task, we can see from the plot that the model is able to correctly generate images of given classes. For the third task, our classifier reach a 0.75 accuracy, but it perform poor when determining the class 'Sandal'. From the generative image in task 2, we can observe that the the ones of 'Sandal' are ambiguous, which might cause confusion with the class 'Anker Boot'.

We also attempted to apply transposed convolution layers to the decoder. But possibly due to the unsuccessful optimization on parameters, the loss is much higher and the result is worse. In the next step, we might rebuild the transposed convolution layers instead of MLP and optimize the parameters again to improve the performance in the third task.

6 Discussion

After this assignment, we would like to discuss the questions about transfer learning raised in the assignment:

Question 1: In what situations would you be able to use transfer learning to aid you in the given tasks?

There are mainly two general cases where it makes sense to consider using transfer learning.First, if the dataset is very small, it may be advantageous to use and even expand upon an existing model that has previously been trained on a larger dataset, such as that produced by another medical researcher. This is a good illustration of how the machine learning community can use transfer learning to collaborate on problem-solving. Many NLP tasks can be solved this way. The second situation is when it would be quicker to alter and/or retrain an existing model than to build and train it from scratch. Or, particularly when the source and target domains are the same, when you cannot or do not wish to fully understand how the source model generates its predictions.

Question 2: How would you use transfer learning to perform the given tasks?

There are, in our perspective, three approaches to use transfer learning. One is employing an already existing model, which is a smart choice when the work at hand is very comparable or when the domains of the model's source and goal are similar. The second involves extending an existing model when we are able to complete new tasks using the output of an existing model. To freeze a portion of the model is the third option. This indicates that the task and the existing task are somewhat identical, and we can use some of the trained models.

Question 3: 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?

When a previous learning strategy interferes with a new activity, transfer learning, therefore, is not recommended. This happens if the source and target are too dissimilar, which causes the first training to be inaccurate.

There are also many phenomenons that worth noting during the task.

- Vague generated images: might mainly because we use CVAE, especially we use MLP instead of deconvolution as decoder. We tried deconvolution but results in high loss value and we do not have enough time to further debug it.
- Loss curve: the reconstruction loss decrease as the epoch increase while the KL loss is the opposite. Adding these two parts together, we get negative ELBO, which is to be minimized. As the average loss is shown in Figure 3 this number indeed increases during the training process.
- ROC: from the Figure 6, we observe that when the false positive rate is low, our model can achieve a relatively high true positive rate. Our model's AUC, or Area Under the ROC Curve, is close to one, indicating that it is performing well.

References

[1] Xuming Ran, Mingkun Xu, Lingrui Mei, Qi Xu, and Quanying Liu. Detecting out-of-distribution samples via variational auto-encoder with reliable uncertainty estimation. *Neural Networks*, 145:199–208, 2022.

[2] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.