Comp5331 Group 23 Final Report

Topic: Supervised Class Distribution Learning for GANs-Based Imbalanced Classification
Type of this project: Implementation

Name: Yuen Zhikun Student ID: 20505288
Name: Tsui Ka Kit Student ID: 20509272
Name: Xie Yulong Student ID: 20509117
Name: Cheng Wai Kit Studnet ID: 20520707

Introduction

Balancing GAN(BAGAN) has been proposed in recent years as a data augmentation tool to restore balance in imbalance datasets. However, BAGAN is using a traditional autoencoder in training stage one. The traditional autoencoder may suffer from the mode collapse problem. Therefore, Wasserstein autoencoder is proposed to improve the performance in BAGAN. Wassertain autoencoder is an autoencoder that improves the mode collapse problem. We expect that WAE can achieve a better performance on mode collapse analysis. In training WAE, we use the maximum mean discrepancy(MMD)-based penalty.

Another improvement on BAGAN is utilizing the labelled data. The traditional autoencoder and Wassertain autoencoder are an unsupervised learning process, which may result in overlapping or fuzzy margin among classes in latent space. In fact, the label information of training data is given in BAGAN, which may be helpful to determine the margin among classes in latent space. The proposed SCDL-GAN considered these two possible improvements, and we try to reimplement the model.

Model Description and Implementation (SCDL-GAN)

The model is not so much different from WAE-GAN and the more famous BAGAN. The basic implementation of the SCDL-GAN is demonstrated in the following diagram of the paper.

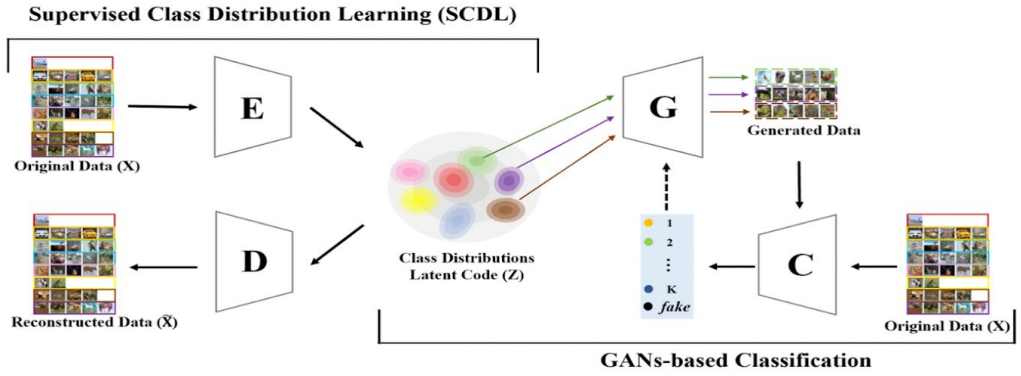


Figure 2. Framework of the proposed supervised class distribution learning for GANs-based imbalanced classification (SCDL-GAN).

Which can be divided in 2 parts:

1) Supervised Class Distribution Learning (SCDL)
2) GANs-based Classification

Supervised Class Distribution Learning (SCDL)

SCDL is no different from the famous WAE apart from the fact that it only works on labelled data and leverages a loss function for supervised learning to enforce the class distribution in the latent space is not entangled. Therefore, SCDL is just an encoder which seek to minimize the following loss function:

$$\mathcal{L}_{SCDL} = \mathcal{L}_{WAE}(P_X, P_{\tilde{X}}) + \lambda_2 \cdot \mathcal{L}_{SV} \qquad (3)$$

With L_SV defined as:

$$\mathcal{L}_{SV} = -\mathbb{E}_{E(Z|X)}[Y_X^T \log(P(Y_X|\tilde{X}))] \qquad (2)$$

which uses a predicted label Y and a real label Y. By introducing a loss function of cross entropy of real label Y and predicted label Y, the SCDL encoding will tend to preserve the class characteristics. As the class can be easily inferred by the data latent representation, the entanglement in the latent space of an imbalance dataset will hence be alleviated.

GANs-based Classification

After the SCDL is well-trained, GANs will hence be adapted in the latent space. The generator will have the exact same architecture as the decoder in the SCDL. A discriminator is built independently. A standard procedure of GANs training is adapted afterward. The noise vector used in this case will be the latent representation created by SCDL. During training, we did not sample our noise vector (latent space vector), instead we sample a real image uniformly and get its latent representation. Due to this, we are able to train the generator in a controllable setting and achieve the final goal: if we feed a latent representation of a specific class of image, the generator will try to generate the exact class of image.

Putting together

To wrap up, the model consists of 2 parts (1) SCDL (2) GAN. After the SCDL(autoencoder) is well trained, GAN style of training is adapted. We can sample specific class's latent representation (assuming multivariate normal) to produce instances of specific class as the generator is trained in such a fashion. Therefore, a balanced dataset can be generated. Another baseline that we have implemented in this project, such as WAE-GAN and BAGAN, are done in a similar fashion with different kinds of auto-encoder.

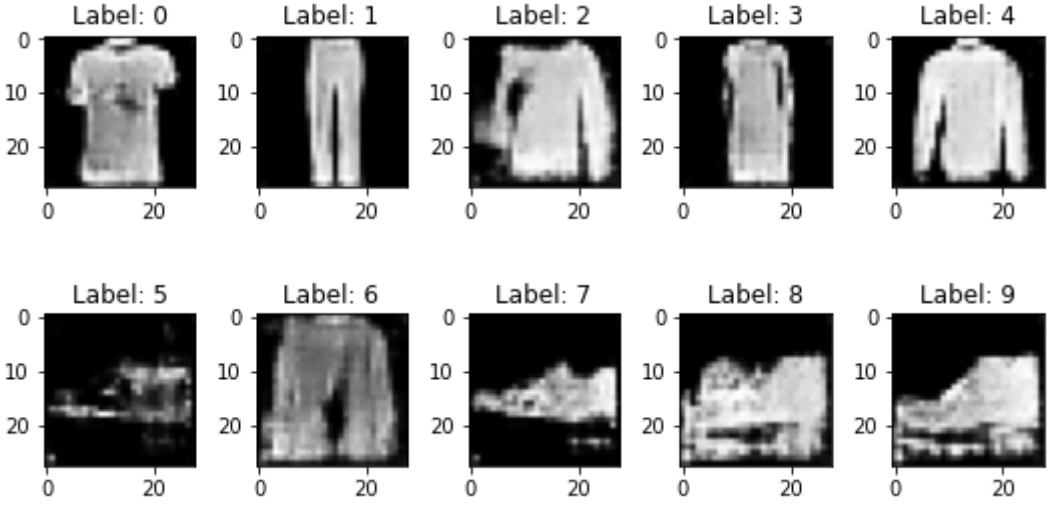
Experiment Result:

We use Fashion MNIST to do experiments with SCDL-GAN, WAE-GAN… We use these models' generator and the latent space from the autoencoder to generate related classes with labels from 0 to 9.

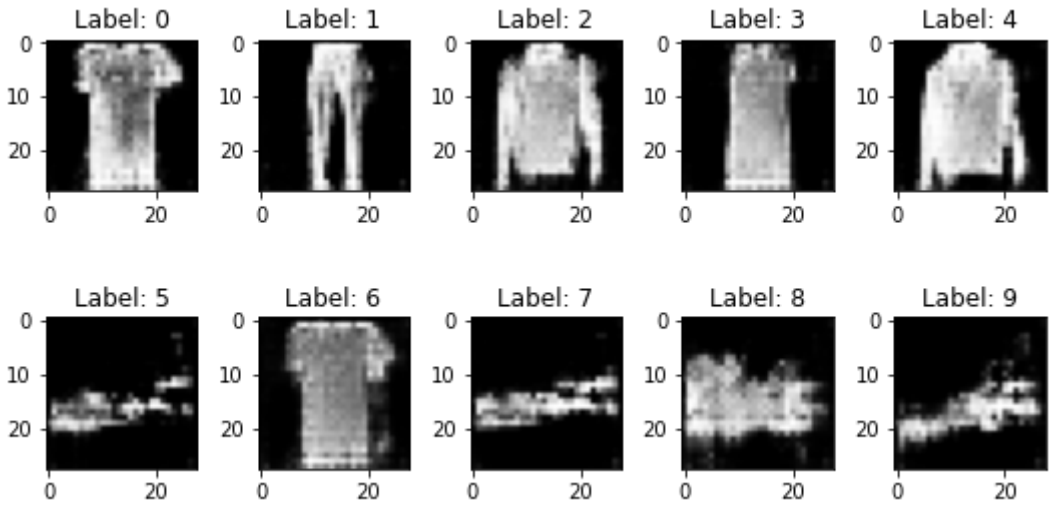
The following code is used to generate images from generator, i is the related classes from 0 to 9.

```
mean_0, std_0 = get_class_distribution(latent, i)
fake_0 = mean_0+torch.normal(0.0, 1.0, size=(100, 1)).squeeze()* std_0
```

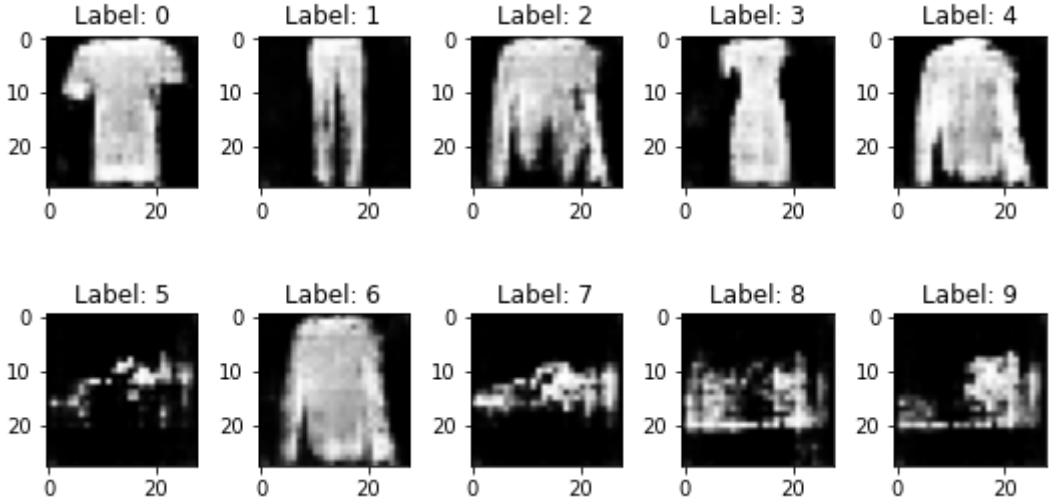
Real image instance



Each class generated by SCDL-GAN



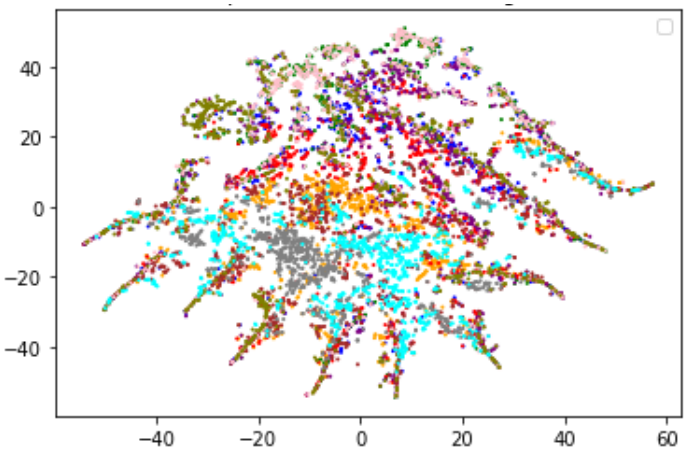
Each class generated by WAE-GAN



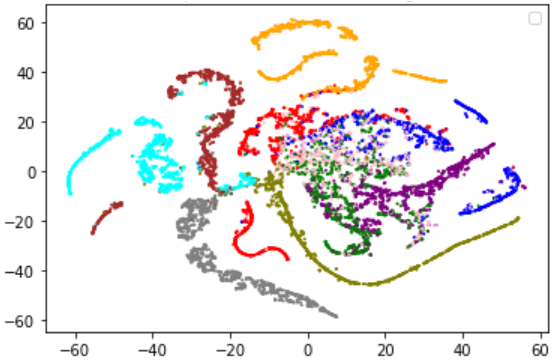
Each classes generated by BAGAN

We can observe that images with labels 0, 1, 2, 3, 4 and 6 generated by these three models are not that obvious. However, images with labels 5, 7, 8 and 9 generated by SCDL-GAN are much clearer than others (All of them are shoes). The fake images of shoes look worse than shirts or trousers because most of the shoes' classes have insufficient data and the models may rely on some similar classes with sufficient data to generate images for the imbalance class. Label 6 is a good example, this class only has 400 data but the three models can generate some complete images of this label.

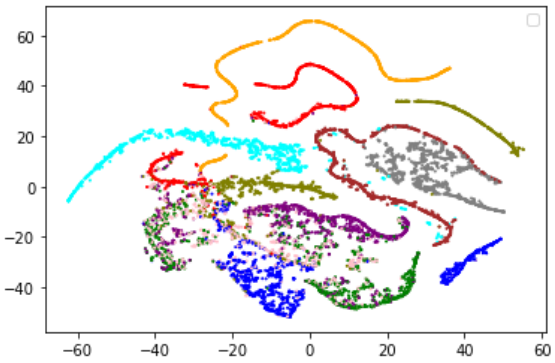
t-SEN analysis:



a) SCDL-GAN



b) WAEGAN

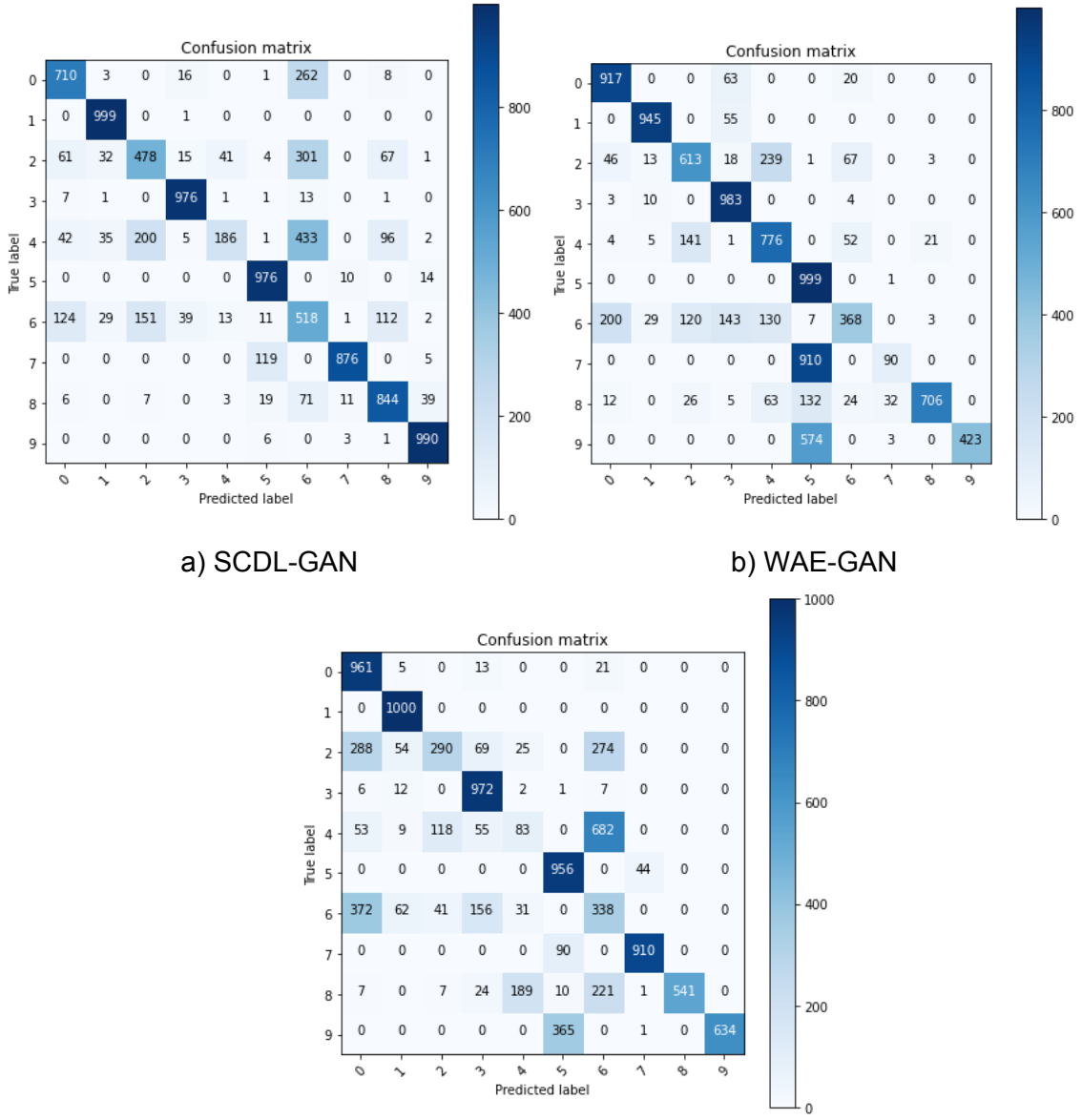


c) BAGAN

The result of t-SEN analysis is different from the paper in which case the discriminator of SCDL-GAN model developed in our project failed to separate the datasets by its class in the latent space apparently, in comparison to the results achieved by discriminators of WAEGAN and BAGAN trained in the project. One possible reason might be that the discriminator of SCDL-GAN is overfit to the artificially generated dataset and some images from different classes in the generated dataset are similar, as a result, it fails to correctly classify the test datasets though some clusters could be identified in the diagram.

Model Collapse analysis:

Model collapse is that the generator learns to produce the one plausible output only. In all following iterations, the generator would only generate that output to the discriminator only. Following the assumption in the paper, if GAN is trained on an imbalanced dataset and it can produce the discriminative structure, then it will not suffer from model collapse. The paper proposes an analysis for the models to analyse their performance of model collapse. We follow the approach in the paper to use a pre-trained ResNet18 provided by pytorch to do model collapse analysis. First, we use the pre-trained ResNet to train on the original 60000 balance dataset. Then, we use this model to predict the balance testing data generated by SCDL-GAN, WAE-GAN and BAGAN. All the dataset has 1000 data on each class with total 10000 data.



a) SCDL-GAN

b) WAE-GAN

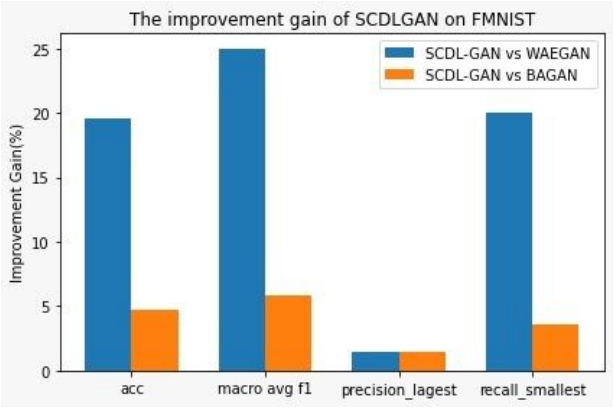
c) BAGAN

Model collapse analysis for SCDL-GAN, WAE-GAN and BAGAN on Fashion-MNIST dataset. The diagonal blocks of the confusion matrix imply how many images of that class are classified correctly. The darker the diagonal blocks are, more images are classified correctly. Although our result is not exactly the same as the one in paper. However, we can still obtain some similar results, especially label 6. In the paper, they mentioned that the performance of WAE-GAN and BAGAN on label 6 is worse than SCDL-GAN with 28.5% and 58.9% only. In our result, we face the same situation, the result of label 6 of SCDL-GAN (51.8%) is better than WAE-GAN (36.8%) and BAGAN (33.8%). With their expectation, SCDL-GAN has the ability to generate class-separable artificial instances, so the confusion matrix looks much clearer than BAGAN and WAE-GAN. Also, it shows that SCDL-GAN does not fall into model collapse problem because this model has ability to detect the discriminative class distribution in latent space for the minority classes.

Improvement Gain:

	Accuracy	macro-averaged F-measure	precision on the largest class	recall on the smallest class
SCDL-GAN	0.77	0.78	0.85	0.96
WAEGAN	0.65	0.63	0.83	0.81
BAGAN	0.74	0.75	0.83	0.92
GAMO	0.72	0.67	0.82	0.96

Some evaluation metrics are listed in the paper, namely macro-averaged F-measure, precision on the largest class as well as recall on the smallest class. A pretrained model, ResNet18, is used to train on the original Fashion MNIST training dataset. Datasets composites of 1000 data per set are generated by the SCDL-GAN, WAEGAN, BAGAN and GAMO trained in the project. The performance of SCDL-GAN is generally better than the others.



The improvement gain is defined in the paper as $IG = (R_{SCDL-GAN} - R_{Baseline})/R_{Baseline}$. Since the architecture of GAMO is too different from the others, we do not use it as a baseline in baseline for improvement gain. It is shown that the performance of SCDL-GAN is better than the other 2 models over all of the evaluation metrics especially better than WAEGAN with around 20% improvement in 3 of the metrics. However, the improvement in precision on the largest class is subtle. This may be due to that the data inside the largest class should be enough so all models have good performance over it. Overall, SCDL-GAN is advantageous than the other GAN models with similar architecture.

Real Case Application:

In daily life, people may encounter some limitations in the complexity of the model such as the RAM is insufficient or there is a need for fast prediction. Under this circumstance, we cannot use a complicated model for prediction like VGG16 or ResNet. However, a simple model may neither be robust enough to achieve a satisfactory performance especially with an imbalanced dataset. In this case, we can make use of the SCDL-GAN developed in the project to generate data for imbalanced dataset or even to further augment the dataset to improve the performance of a simpler model.

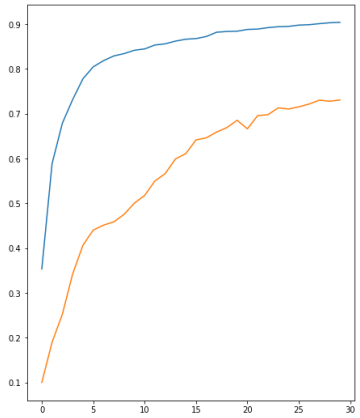
Setting:

- Dataset: Fashion MNIST
- Number of classes: 10
- Distribution before pre-processing:
  [4000, 2000, 1000, 750, 500, 350, 200, 100, 60, 40]
- Model Size: 18.59M
- Number of parameters: 4871340
- Model architecture:

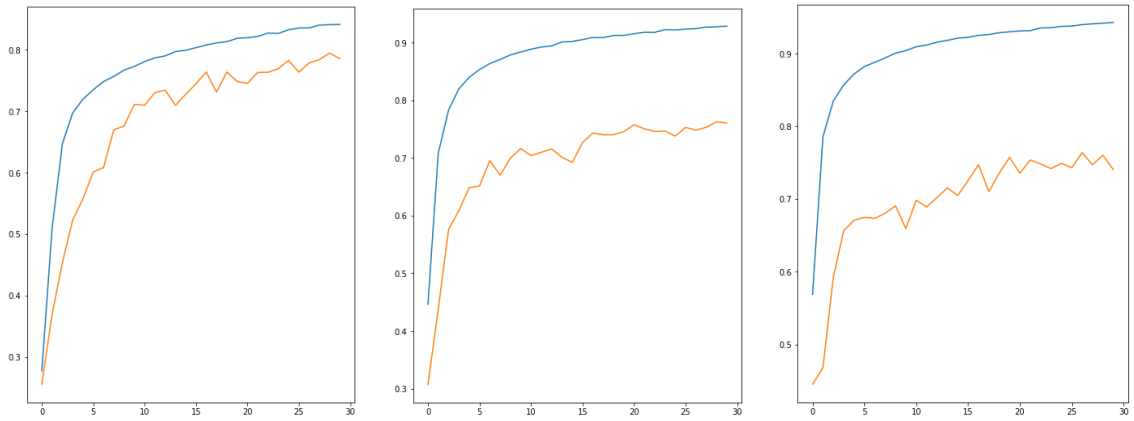
(0): BatchNorm2d(1, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(1): ReLU(inplace=True)
(2): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1))
(3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
(4): ReLU(inplace=True)
(5): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1))
(6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(7): Flatten(start_dim=1, end_dim=-1)
(8): Linear(in_features=4608, out_features=1024, bias=True)
(9): ReLU(inplace=True)
(10): Linear(in_features=1024, out_features=128, bias=True)
(11): ReLU(inplace=True)
(12): Linear(in_features=128, out_features=10, bias=True)

A simple CNN is applied to perform classification on Fashion MNIST dataset. The original graph shows its performance when training with imbalanced classes data created according to the aforementioned class distribution setting. Then, the SCDL-GAN trained in the project is applied to augment the imbalanced dataset to 4500, 6000 and 10000 data per class respectively including the original imbalanced data.

Model performance with the imbalanced dataset:



Model performance with the artificially generated balanced dataset (4500, 6000 and 10000 per class):



Improvement:
The plots show that the accuracies at the beginning are hugely improved in the balanced dataset with artificially augmented data. The final performance when training with the balanced datasets also improves. Training on original imbalanced dataset only achieve 73.05% test accuracy while training on the artificially balanced datasets all achieve higher accuracies with 78.52%, 76.03% and 74.04% for the number of data in each class being augmented to 4500, 6000, 10000 respectively.

Limitations:
The accuracies after epoch 10 are generally more unstable. The reason might be that the data generated is not perfect and some may perform as noises and affect the training result. This might also be the reason why when more augmented data is used for training, the dataset is more likely to be overfit as shown in the last two graphs. Thus, the amount of data to generate should be carefully decided. However, with correct settings, it is able to improve the performance with the same level of model size and complexity.

Reference:
WAE-MMD implementation:
https://github.com/1Konny/WAE-pytorch
WAE:
https://arxiv.org/pdf/1711.01558.pdf
WAE tutorial:
https://github.com/sedelmeyer/wasserstein-auto-encoder
Pytorch GAN tutorial:
https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html
BAGAN:
https://arxiv.org/abs/1803.09655
BAGAN implementation:
https://github.com/IBM/BAGAN
GAMO:
https://openaccess.thecvf.com/content_ICCV_2019/papers/Mullick_Generative_Adversarial_Minority_Oversampling_ICCV_2019_paper.pdf
GAMO implementation:
https://github.com/SankhaSubhra/GAMO

Contribution:

Yuen Zhikun:
I did literature review on the baseline model BAGAN and modify the code of BAGAN on the GitHub: https://github.com/IBM/BAGAN provided by the original research team. In their original model, there is only one imbalance class in each training, but there are multiple imbalance classes in our paper to implementation. Thus, I need to modify the BAGAN and make it has multiple imbalance classes with different distribution in each training. Also, I try different hyperparameters and architectures on BAGAN source Code. After finishing modification, run BAGAN with dataset MNIST and CIFAR10 and observe their results. Following the paper's description to conduct experiments on SCDL-GAN, BAGAN, WAE-GAN on Fashion MNSIT. First, training the three models on the three models on Fashion MNIST with different hyperparameters and then generate three datasets with 10000 data (1000 data on each classes) through the generators on the three models. Then, training a pre-trained ResNet18 on the original balance data and then do model collapse analysis, mode improvement analysis and tSEN analysis on the three models. Help for the real case application for SCDL-GAN.

Tsui Ka Kit:
I did literature review on the Wasserstein autoencoder and the MMD penalty that was used in WAE. Also, I built a skeleton code as a starting point for the whole project code based on the WAE tutorial here:https://github.com/sedelmeyer/wasserstein-auto-encoder. Because we need to make the original dataset an imbalance dataset. I implemented an helper function to get the imbalance dataset on MNIST, MNIST Fashion, CIFAR10 and SVHN by giving the weight of each class. Also, I mainly focused on designing and evaluating the losses function in the AE stage, including the supervised loss and MMD loss. And implement the training procedure of the AE stage.

Xie Yulong:
I did literature review on the GAMO model and modified the codes provided by the authors on GitHub (https://github.com/SankhaSubhra/GAMO). The architecture and details of the model are explored, and some functionalities are integrated to improve the training process. The result and performance of GAMO on MNIST and Fashion MNIST were tested. Function to generate different amounts of data in different classes by the generator of GAMO was also implemented and could easily generate data for experiments. In the later part, I implemented the simple demo for the real world application in our project and the performance of the model with and without our data generator being applied was further tested and studied to draw conclusion and discuss the effectiveness in possible applications of SCDL-GAN. I also assisted in the experiment part to compare the performance of the four models developed and trained in this project.

Cheng Wai Kit:
I did literature review on the Wasserstein autoencoder and GAN. After understanding the GAN and also the special implementation of GAN in the paper (instead of only fake or real, the generator also predicts which class the instance belongs to), I designed the training process of GANs. I also designed the architecture of auto-encoder and GAN to ensure it performs well in the dataset we have (using MNIST as a starting point). Then I built the

skeleton code of the whole Autoencoder - GANs (including BAGAN, WAE-GAN, SCDL-GAN) training process to let my teammates to experiment. At the same time, I reviewed the paper to suggest what experiment to implement in this project and also provided assistance in the experiment.