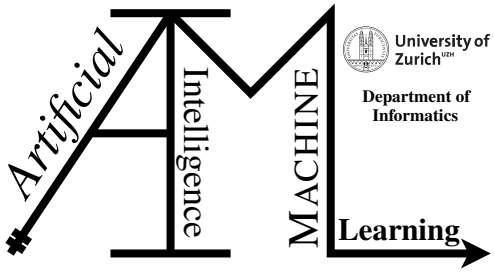
Combined-GAN: Utilizing GAN for Open-Set Recognition by Generating Effective Unknown Samples

Master Thesis

Peng Yan
19-762-780

Submitted on
May 15 2022

Thesis Supervisor
Prof. Dr. Manuel Günther



Master Thesis

Author: Peng Yan, peng.yan@uzh.ch

Project period: November 15 2021 - May 15 2022

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

First of all, I would like to express special thanks to Prof. Dr. Manuel Günther who supervised my thesis. Thank you for inspiring me in the field of deep learning and sparking my interest in the Generative Adversarial Network in your course. Thank you for giving me the opportunity to do my Master thesis in Artificial Intelligence and Machine Learning group. Thank you for giving me constructive suggestions and productive inspirational ideas and always pointing me in the right direction when I had difficulties.

I would like to thank the University of Zurich for providing me an opportunity to further explore more unknown possibilities in my future professional career. I have greatly benefited from international academic atmosphere and learning resources at University of Zurich.

I would like to thank the open-source communities worldwide. To extend my gratitude, I thank all the scientists, researchers and influencers who are willing to help and inspire many others.

I would like to thank everyone who helps and assists me during my thesis. Specifically, many thanks to my colleagues and friends for proofreading my thesis.

Last but not least, I would like to express my deep gratitude to my parents for their endless and unconditional support.

Abstract

Discovering the unknown world is a big challenge. Different from traditional classification, for Open-Set Recognition (OSR), the open-set model needs to classify known data as well as tackle unknown data. In this thesis, we utilize Generative Adversarial Network (GAN) to generate effective open-set samples (unknown data) to assist open-set model to know more information about the open space (the space far from known/training data). In our Combined-GAN model, we adopt encoder-decoder network architecture for the generative model. By combining the latent space from two different known classes, the generated samples can acquire features from the corresponding known classes. We assume the generated samples locate around the decision boundaries of known classes and can be represented as open-set samples. The generated samples are fed into open-set model together with known samples for OSR. Compared with other OSR approaches in different open-set scenarios, the quantitative and qualitative results show our generative model can generate effective unknown samples for the open-set model to classify known classes and detect unknown classes at the same time.

Zusammenfassung

Die Entdeckung der unbekannten Welt ist eine große Herausforderung. Anders als bei der traditionellen Klassifizierung, muss das Open-Set Modell für die Open-Set Recognition (OSR) sowohl bekannte Daten klassifizieren, als auch unbekannte Daten angehen. In dieser Arbeit wenden wir das Generative Adversarial Network (GAN) an, um effektive Open-Set Proben (unbekannte Daten) zu generieren, um weiter das Open-Set Modell zu unterstützen, um mehr Informationen über den offenen Raum zu erhalten (den Raum, der weit entfernt von bekannten/zu trainierenden Daten liegt). In unserem Combined-GAN Modell übernehmen wir eine Encoder-Decoder Netzwerkarchitektur für das generative Modell. Durch Kombinieren des latenten Raums von zwei verschiedenen bekannten Klassen können die generierten Proben Merkmale von den entsprechenden bekannten Klassen erwerben. Wir nehmen an, dass sich die generierten Proben um die Entscheidungsgrenzen bekannter Klassen und als Open-Set Proben dargestellt werden können. Die generierten Proben fließen zusammen mit bekannten Proben für OSR in das Open-Set Modell ein. Im Vergleich zu anderen OSR Methoden in verschiedenen Open-Set Szenarien zeigen die quantitativen und qualitativen Ergebnisse, dass unser generatives Modell effektive unbekannte Proben für das Open-Set Modell generieren kann, um bekannte Klassen zu klassifizieren und gleichzeitig unbekannte Klassen zu erkennen.

Contents

Chapter 1

Introduction

With the development of machine learning and deep learning, various models have been used for classification tasks. Almost all machine learning based recognition algorithms only train and test on the datasets with finite known classes, which is known as Closed-Set Recognition (CSR). However, in real-world applications, there usually exist unknown classes that are not seen in the training process. Thus, any unknown sample is classified as one of the known classes during testing. With incomplete knowledge of the unknown classes, it can lead to wrong classification.

This has prompted the formalization of Open-Set Recognition (OSR), which tries to deal with more realistic scenarios, which accept the real-world is comprised of known objects and unknown objects. Since the data in the real-world is unpredictable and dynamic, robust open-set models should have the ability to recognize and reject the objects from unseen classes and classify the known objects correctly at the same time.

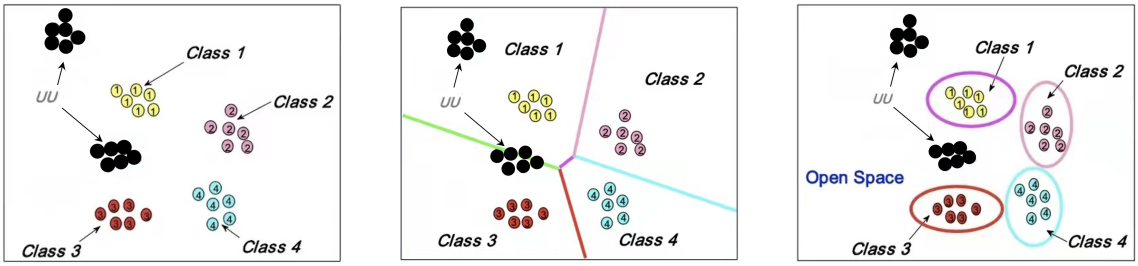
In order to better understand the problem, based on the definition of categories by Dhamija et al. (2018) and Geng et al. (2021), all classes in the OSR problem can be generally categorized into the following cases:

- Known Known Classes ($\mathcal{KK}$): the known classes of interest with distinctly labeled positive training samples. Data from $\mathcal{KK}$ is noted as $\mathcal{D}_{\mathcal{KK}}$.
- Known Unknown Classes ($\mathcal{KU}$): labeled negative samples, not necessarily grouped into meaningful classes, such as the background classes, garbage, the known classes of no interest, etc. The samples from $\mathcal{KU}$ are usually trained together with $\mathcal{KK}$ to get the information about the unknown world, but since the unknown classes contain all types of classes except for the $\mathcal{KK}$, the $\mathcal{KU}$ is only a subset of all unknown classes. Data from $\mathcal{KU}$ is noted as $\mathcal{D}_{\mathcal{KU}}$.
- Unknown Unknown Classes ($\mathcal{UU}$): classes without any information regarding them during training: not only unseen but also having no side information (e.g., semantic/attribute information, etc.) during training. They are served as a part of the testing dataset. Data from $\mathcal{UU}$ is noted as $\mathcal{D}_{\mathcal{UU}}$.

Figure 1.1 gives a toy example by visualizing $\mathcal{D}_{\mathcal{KK}}$, $\mathcal{D}_{\mathcal{KU}}$ and $\mathcal{D}_{\mathcal{UU}}$ in CSR scenario and OSR scenario. In CSR scenario (Figure 1.1(b)), any $\mathcal{D}_{\mathcal{UU}}$ is undoubtedly classified as one of the known classes, resulting in a high False Positive Rate (FPR[1]). Differently, in OSR scenario (Figure 1.1(c)), an open-set model/classifier is able to detect $\mathcal{D}_{\mathcal{UU}}$ and classify $\mathcal{D}_{\mathcal{KK}}$ effectively.

The OSR problem is often challenging because they must balance maintaining accuracy on the known classes while handling the uknown classes. Open-set models should learn more information about the unknown world during training and be able to tighten the decision boundaries limiting the scope of known data, reserving space for the unknown data, as shown in Figure 1.1(c).

[1]FPR is the probability that an actual negative sample will test positive.



(a) Data distribution (b) CSR scenario (c) OSR scenario

Figure 1.1: COMPARISON BETWEEN CSR AND OSR. Figure 1.1(a) shows the data distribution of the original toy dataset, which consists of 4 $\mathcal{KK}$ (denoted as colored dots) and $\mathcal{UU}$ (denoted as black dots). Figure 1.1(b) depicts the CSR scenario, where the whole dataset is divided into four areas separated by decision boundary lines (noted in different colors). Each area represents a distinctive class. Figure 1.1(c) describes the OSR scenario, an open-set model/classifier can recognize the open space by pushing the known areas as small as possible. *Source*: Geng et al. (2021).

A vast majority of recognition models are designed for CSR, where the primary assumption is that all categories are known in advance. Since Deep Neural Network (DNN) has the strong representation learning ability, DNNs have been widely in visual recognition/classification tasks. For dealing with CSR, the output of the last fully-connected layer (logit layer) in DNNs is fed to the softmax function, which produces a probability distribution over a certain number of classes. Specifically, given a sample x, let $l_k(x)$ represents the logit value for class $k(k \in \{1, \ldots, K\})$, we use Equation (1.1) to compute the softmax score (probability of membership for each class).

$$S_k(x) = \frac{e^{l_k(x)}}{\sum_{k'=1}^{K} e^{l_{k'}(x)}} \tag{1.1}$$

However, we need to handle unknown samples in the OSR problem. So how can we recognize whether the sample is unknown or not? There are basically two ways to estimate the possibility of being detected as unknown. One way is to directly provide a probability for x by adding one more dimension on the logit layer. Another way is to compute $\arg\max_k P(k|x)$ and reject the sample as unknown if $\arg\max_k P(k|x)$ is lower than a certain threshold.

However, these type of global thresholding assumes well calibrated probabilities, and breaks down in many real-world tasks. For example, some fooling images, especially generated by the adversarial attacks (Goodfellow et al., 2015), which are hard for humans to recognize and can be classified incorrectly with high confidence using DNNs.

Some researchers used unknown samples for OSR. During training, some unknown classes are labeled as the background class. However, it would only work for the unknown samples that are related to the ones in the training time. This can also bring about an issue: accessibility and selection of additional data as effective unknown samples.

To solve the problem, one way is generating synthetic images by exploring the potential of known classes. Since Generative Adversarial Network (GAN (Goodfellow et al., 2014)) has proven to be a notable generate model with wide attention, it can be used for data augmentation for OSR. In GAN, the generator and the discriminator are trained adversarially. It generates sharp and convincing realistic data with the same statistics as the training set.

In this thesis, based on GAN, we proposed Combined-GAN model to generate synthetic samples by combining the deep features of two different known classes. We assume the generated samples obtain the deep features from both classes. In other words, they are assumed to located around the decision boundaries of different known classes, i.e., the generated samples are

assumed to have equal probabilities on two different known classes. We assume the generated samples can be used as effective augmented data for training the open-set model to learn more knowledge about the unknown world. During testing, the open-set model is expected to effectively detect the unknown classes and classify known classes simultaneously.

Thesis Outline

In this thesis, we mainly focus on utilizing the generative model to generate open-set samples for tackling the the OSR problem. The main content of this thesis includes the following chapters:

Chapter 2 introduces the background and related work. First, the OSR topic is described in detail, including the basic concept and main methods to solve the OSR problems. Then, GAN and some GAN-based solutions for the OSR problem are introduced. In the end of this chapter, some important network architectures will be introduced.

In chapter 3, we will formulate our OSR approach and introduce a new generative model to generate unknown samples for OSR. Then, chapter 4 explains how we set up the experiments, including benchmark datasets used in the experiments, detailed network architectures, training parameters and evaluation metrics. Chapter 5 shows some results of the experiments and comparison between our approach and other OSR approaches.

Next, some observations and discussion about the results of the experiments will be delivered, followed by some limitations in chapter 6. Finally, in the last chapter, I will summarize the main results and insights from this thesis and provide potential improvement for future work.

Chapter 2

Related Work

2.1 Open-Set Recognition

2.1.1 Basic Concept

The OSR problem was first introduced and formalized by Scheirer et al. (2013). As discussed by Scheirer et al. (2013), open space $\mathcal{O}$ is usually considered far away from known data. So labeling any samples as known samples in the open space incurs risk, which is called Open Space Risk $R_{\mathcal{O}}$. It is formalized as the relative measure of open space compared to the overall measure space S_o:

$$R_{\mathcal{O}}(f) = \frac{\int_{\mathcal{O}} f(x)dx}{\int_{S_o} f(x)dx} \tag{2.1}$$

where f is a measurable recognition function, x is a feature vector representation of the detected target from open-set classes, $f(x) = 1$ indicates that some known classes are recognized, otherwise $f(x) = 0$. From Equation (2.1), $R_{\mathcal{O}}$ is considered as the fraction of the positively labeled open space compared to the overall positively labeled space. Therefore, $R_{\mathcal{O}}$ will be larger if more open space is labeled as positive.

With the concepts of Open Space Risk in mind, the OSR problem is defined as follows:

Definition 1 (The Open-Set Recognition Problem (Scheirer et al., 2013)) *Given the training data V, an Open Space Risk function $R_{\mathcal{O}}$, and an empirical risk function R_{ε} , open-set recognition is to find a measurable recognition function $f \in \mathcal{H}$, and f is defined by minimizing the following Open-Set Risk:*

$$\arg\min_{f \in \mathcal{H}} \{R_{\mathcal{O}}(f) + \lambda_r R_{\varepsilon}(f(V))\} \tag{2.2}$$

where λ_r is a regularization constant.

OSR is defined as minimizing the open-set risk, which combines the Open Space Risk and empirical risk, over the space of allowable recognition functions. Although this initial definition mentioned above is more theoretical, it provides an important guidance for subsequent OSR modeling (Geng et al., 2021).

2.1.2 Open-Set Recognition Methods

In the traditional classification task, the classifier learns to return the most likely class for input samples. For tackling the OSR problem, it is not sufficient to just calculate the most likely class, the classifier should have ability to recognize the unknown samples and reject them effectively.

Many open-set models and training methods have been proposed to make recognition models robust to the open-set samples. Generally, the taxonomy of OSR methods can be categorized into two categories, the DNN-based methods can be further divided into two subcategories:

- Traditional ML-based methods
- Deep Neural Network (DNN)-based methods
  - DNN-based discriminative models
  - DNN-based generative models

DNN-based discriminative models will be explained in the following subsection. DNN-based generative models, mostly focusing on providing effective open-set samples for OSR, will be introduced in Section 2.2.2. More detailed taxonomy and corresponding literatures about OSR methods can be seen in the OSR review (Geng et al., 2021). Recently more and more researchers tried to solve the OSR problem by using deep learning models. In this thesis, we will mainly focus on DNN-based methods.

Traditional ML Methods-based Methods

Early researchers in this area primarily focused on Support Vector Machine (SVM (Cortes and Vapnik, 1995)). SVM has been widely used in traditional classification tasks. However, in open-set scenarios, when $\mathcal{D}_{\mathcal{UU}}$ appear during testing, the classification will decrease dramatically, because the $\mathcal{D}_{\mathcal{UU}}$ usually falls into space separated only by $\mathcal{KK}$. To overcome this problem, many SVM-based OSR methods have been proposed.

Scheirer et al. (2013) proposed 1-vs-Set machine based on SVM, which incorporates an Open Space Risk term in modeling to account for the space beyond the reasonable support of $\mathcal{KK}$.

In Jain et al. (2014), based on the statistical Extreme Value Theory, a Weibull distribution is used to model the posterior probability of inclusion for each $\mathcal{KK}$ and an example is classified as $\mathcal{UU}$ if the probability is below a rejection threshold.

Further, Weibull-calibrated SVM (W-SVM) model was proposed in Scheirer et al. (2014). They formulated a compact abating probability (CAP) model, where probability of class membership abates as points move from known data to open space.

Deep Neural Network-based Methods

A typical softmax cross entropy loss is usually used for classification in training DNNs, which inevitably incurs the normalization problem, the closed set nature of deep networks forces them to choose from one of the known classes. In OSR, since a unknown sample does not belong to any known class, it is expected to have a lower probability on all known classes. A natural approach is to apply a threshold on the output probabilities, as mentioned in Section 2.1. Thus, the classifier can detect the unknown by thresholding the uncertainty. As explained in Section 2.1, some adversarial samples can cheat DNNs with high confidence, hence, thresholding the uncertainty is not sufficient to determine what is unknown. To solve the problem, many different approaches using DNNs have been proposed to address the OSR problem.

The OpenMax approach (Bendale and Boult, 2016) is the first deep network approach to solve OSR formally. They allowed the rejection of the unknown images from open-set world. Instead of softmax layer, they introduced a new model layer OpenMax, which estimates the probability of an input from an unknown class. First, each class is represented as a mean activation vector (MAV) with the mean of the activation vectors (only for the correctly classified training samples) in the penultimate layer of that network. Next, the training samples' distances from their corresponding class MAVs are calculated and used to fit the separate Weibull distribution for each class. Further,

the activation vector's values are redistributed according to the Weibull distribution fitting score. In this way, a maximum radius is fit around each class in the activation vector feature space, and any activation vectors outside of this radius are detected as open-set examples. OpenMax effectively recognizes and rejects fooling or unrelated open-set data.

Later, by using logits optimized targeting system (LOTS), Rozsa et al. (2017) analyzed and compared the adversarial robustness of DNNs using SoftMax layer with OpenMax: although OpenMax provides less vulnerable systems than SoftMax to traditional attacks, it is equally susceptible to more sophisticated adversarial generation techniques directly working on deep representations. Therefore, adversarial samples are still serious challenge for OSR.

Entropic Open-Set (EOS) Loss

In Dhamija et al. (2018), EOS loss is introduced to address the OSR problem. Rather than approximating the probability of the unknown class, EOS focus on reducing $\arg\max_k P(k|x)$ for unknown samples. Instead of using softmax thresholding for the unknown classes, they use EOS loss which can maximize entropy at the softmax layer. The EOS Loss J_E is defined as

$$J_E(x) = \begin{cases} -\log S_k(x) & \text{if } x \in \mathcal{D}_{\mathcal{KK}} \text{ is from class } k \\ -\frac{1}{K}\sum_{k=1}^{K} \log S_k(x) & \text{if } x \in \mathcal{D}_{\mathcal{KU}} \end{cases} \tag{2.3}$$

We can see from Equation (2.3), they keep the softmax loss calculation untouched for $\mathcal{KK}$, they modify it to train on samples from $\mathcal{KU}$ and seek to equalize their logit values. In other words, for the unknow samples, EOS loss is used to achieve maximum entropy distribution of uniform probabilities over the known classes.

The minimum of the loss J_E for the unknown samples is achieved when the softmax scores $S_k(x)$ for all known classes are identical (Dhamija et al., 2018), i.e., $S_k(x) = \frac{1}{K}$.

2.2 Generative Adversarial Network

Generative Adversarial Network (GAN) was first introduced by Goodfellow et al. (2014). As an implicit deep generative model, GAN can approximate the original data distribution.

As shown in Figure 2.1, GAN consists of two adversarial networks: the generator G and the discriminator D. G tries to generate images similar to original images to deceive D while D tries to distinguish real images and generated fake images.

The adversarial process in GAN can be formulated as a minimax objective function:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.4}$$

Where $p_{\boldsymbol{x}}$ is the image data distribution and $\boldsymbol{z}$ is usually a noise variable with prior distribution such as uniform or normal distribution.

During training, G and D are usually updated separately using the objective function (Equation (2.4)) by using gradient descent. For D, the goal is to maximize the objective function, specifically, for any image $\boldsymbol{x}$, D wants $D(\boldsymbol{x})$ to approximate 1 (real) and $D(G(\boldsymbol{z}))$ to 0 (fake). Meanwhile, G aims at minimizing the value function, driving $G(\boldsymbol{z})$ close to $\boldsymbol{x}$ and $D(G(\boldsymbol{z}))$ to 1.

By continuously training the two adversarial networks, the ultimate goal of GAN is that G can capture the original image data distribution and generate fake undistinguished images, while D fails to distinguish the real and fake images. Theoretically, when the adversarial training reaches the Nash equilibrium[1], the minimax function achieves global optimum: $p_G(\boldsymbol{z}) = p_{\boldsymbol{x}}$ (Goodfellow et al., 2014). Although Nash equilibrium theoretically exits, it is hard to obtain in practice, the

[1] In a Nash equilibrium, each player is assumed to know the equilibrium strategies of the other players and no player has anything to gain by changing only their own strategy.

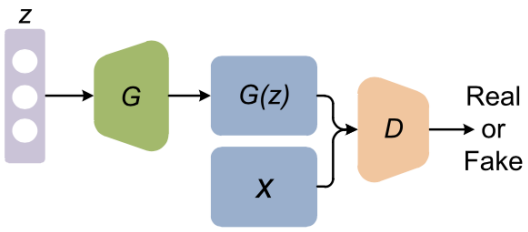


Figure 2.1: GAN NETWORK ARCHITECTURE. *Source*: Pan et al. (2019)

detailed explanation and discussion can be found in Goodfellow et al. (2014) and Farnia and Ozdaglar (2020).

It is quite challenging to train GAN because of problems like mode collapse (the generator only produces limited varieties of samples), unstable training and non-convergence. For example, when the generated samples are initially very poor, D can easily differentiate real and fake samples. Hence, for any generated samples $G(\boldsymbol{z})$, $D(G(\boldsymbol{z}))$ will be close to zero, leading to a very small gradient of $log(1 - D(G(\boldsymbol{z})))$, in this case, G will stop learning.

To address these problems and improve the performance of GAN, tremendous solutions have been provided based on better network architecture designs, new objective functions and other alternative strategies. DCGAN (Radford et al., 2016) adopted all convolutional networks which replace deterministic spatial pooling functions (such as max-pooling) with strided convolutions to allow the network to learn its own spatial downsampling. Besides, Batch Normalization (Ioffe and Szegedy, 2015) was adopted to stabilize training by normalizing the input to have zero mean and unit variance.

To avoid mode collapse and further improve the training stability, WGAN (Arjovsky et al., 2017) minimizes the Earth Mover's distance (EM distance) between the generated distribution and the real distribution as below,

$$\min_{G} \max_{\|D\|_L \leq 1} \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}}[D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}}[D(G(\boldsymbol{z}))] \tag{2.5}$$

where D is constrained to be 1-Lipschitz continuous and implemented by weight clipping. Instead of classifying the images as real or fake, the EM distance in WGAN is continuous and differentiable, which means we can measure how similar the generated distribution and real distribution are. Thus, the WGAN is trained in the direction of minimizing EM distance until it achieves an optimal solution.

Further, WGAN-GP (Gulrajani et al., 2017) improves WGAN by imposing 1-Lipschitz continuity on the discriminator instead of weight clipping. The gradient penalty is used to make sure that D grows at most linearly and thus maintains the stability of training process. In our work, we also adopt WGAN-GP for the adversarial training.

Instead of directly maximizing the output of the discriminator, Salimans et al. (2016) proposed feature matching to address the instability of GAN by specifying a new objective for the generator that prevents it from overtraining on the current discriminator. Let $f(\boldsymbol{x})$ denote activations on an intermediate layer of the discriminator, the new objective for the generator is minimizing the following funtion:

$$\|\mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}} f(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}} f(G(\boldsymbol{z}))\|_2^2 \tag{2.6}$$

For original GAN models, there is no control on the modes of generated samples. Thus, the Conditional GAN (Mirza and Osindero, 2014) is introduced to generate data conditioned on some condition feature both on generator and discriminator. By adding additional information on the

model, it is possible to direct the generative process and improve the discriminator's learning efficiency.

To generate new images blending features from different data, CycleGAN (Zhu et al., 2017) implements adversarial loss and cycle consistency loss to train a feature transfer model to translate images from one domain to another domain. CoGAN (Liu and Tuzel, 2016) is designed for learning a joint distribution of images in two different domains by sharing a subset of parameters of a pair of GANs.

2.2.1 Semi-Supervised GAN

Salimans et al. (2016) showed that using class labels improves the quality and quantity of the generated data. In the original GAN model, the discriminator D learns in an unsupervised way. Given original data without labels, D can predict unseen samples as real or fake. In their work, they selected part of the known data for supervised training, the rest for unsupervised learning. In the implementation, the output of the penultimate layer of the discriminator is used as a closed set classifier C for supervised learning. To do the supervised learning, the class labels of known data are provided, and C classifies known data as one of the known classes by minimizing the cross entropy between the one-hot encoded class labels and the softmax scores.

For the unlabeled data, to do the real/fake task, one solution is adding one more dimension for the output of the logit layer. Since subtracting a general function from each output of the logit layer does not change the output of the softmax score, they fixed $l_{k+1}(\boldsymbol{x}) = 0$. In this way, the original classifier C with K classes can be reused as K+1-class classifier by augmenting the K-dimensional vector of logits with an additional constant 0. Then, the discriminator D is defined by the following formula by applying a normalized sum of the exponential output of the logit layer in C:

$$D(\boldsymbol{x}) = \frac{Z(\boldsymbol{x})}{Z(\boldsymbol{x}) + 1}, \text{ where } Z(\boldsymbol{x}) = \sum_{k=1}^{K} \exp\left[l_k(\boldsymbol{x})\right] \tag{2.7}$$

Thus, $D(\boldsymbol{x})$ is converted as a binary activation function, which can be used to predict real or fake. If $\boldsymbol{x}$ is classified as one of the known classes, $Z(\boldsymbol{x})$ will be larger, leading $D(\boldsymbol{x})$ close to 1, which means $\boldsymbol{x}$ is classified as real data. Otherwise, if C does not have high confidence on $\boldsymbol{x}$ for all classes, $D(\boldsymbol{x})$ will be close to 0, which means $\boldsymbol{x}$ is classified as fake data.

2.2.2 Generative Model for Open-Set Recognition

Since GAN models have the strong generative ability and gained huge success in image generation, some researchers also apply GAN models to solve the OSR problem (Zongyuan Ge and Garnavi (2017) , Neal et al. (2018), Jo et al. (2018)). In this case, the key goal for GAN is to generate appropriate open-set samples that can be used to represent unknown data distribution.

Zongyuan Ge and Garnavi (2017) proposed Generative OpenMax (G-OpenMax) by using conditional GAN (Mirza and Osindero, 2014) to generate synthetic samples. The synthetic samples are generated from mixture distributions of known classes in latent space, which leads to plausible representation with respect to the known classes domain. All incorrectly predicted samples are selected as open-set samples, explicit representation of unknown classes enables the classifier to locate the decision margin with the knowledge of both known and unknown samples. Finally, they provided explicit probability estimation over unknown categories based on OpenMax. Although G-OpenMax effectively detects unknown classes in monochrome digit datasets, it has no significant performance improvement on natural images.

Neal et al. (2018) adopted an encoder-decoder GAN architecture to generate counterfactual samples. Firstly, images were encoded as latent representations after adversarial training. Secondly, in order to generate counterfactual images which are close to known classes, but do not belong to any known classes, they updated the latent representations using gradient descent to find the optimal latent point and then decoded the latent point to generate open-set images. Lastly, after the generative model was fully trained, they labeled the generated images as unknown classes and assigned them as an additional class, they were trained together with known classes using an open-set classifier. Similarly, we also adopt encoder-decoder GAN architecture to generate open-set samples. The difference is that, in their approach, after extracting deep feature representations from an image, they tried to optimize the feature representations to make sure it can be decoded into an open-set sample. Instead, we combine the feature representations of image pairs to generate open-set samples.

Similarly, Jo et al. (2018) also adopted GAN to generate fake samples and enhanced classifier's robustness for unknown data. Marginal denoising autoencoder (MDAE) based on semi-supervised GAN (Salimans et al., 2016) was introduced to generate fake images that reside around feature space of known images. They tried to capture the deep representation of discriminator, by modeling adjacent feature space of known classes, the goal of MDAE is to generate data similar to the data of known classes but not the same one that is considered as fake negative data. The drawback is that, since MDAE is also a neural network, the approach needs to deal with more hyper-parameters.

Inspired by the adversarial learning, Yu et al. (2017) proposed adversarial sample generation (ASG) framework for the OSR problem. ASG generates negative instances of seen classes by finding data points that are close to the training instances, given that they can be separated from the seen data by a discriminator. Meanwhile, ASG also generates positive instances that cannot be discriminated from the seen class instances to enlarge the training dataset. With the generated samples, ASG then learns to tell the seen from the unseen in a supervised manner.

2.3 Classic Convolutional Neural Networks

Convolutional Neural Network (CNN), one class of DNNs, has been popular especially in Image Recognition area. In fully connected DNNs, each neuron in one layer is connected to all neurons in the next layer. However, image datasets are usually high dimensional. Thus, using the traditional fully connected DNNs can easily fall into the curse of dimensionality phenomena, resulting in high computation and low efficiency. Since CNNs are based on shared-weight architecture of convolution kernels/filters to extract deep features, they can effectively reduce the overall parameters of network while capturing the deep feature and detecting complex patterns of image data. A typical CNN consists of an input layer, an output layer and hidden layers including multiple convolutional layers, pooling layers, fully connected layers and normalization layers.

In this section, we introduce two classic CNN models: LeNet++ (Wen et al., 2016) and ResNet-18 (He et al., 2016).

2.3.1 LeNet++

LeNet++ is a deeper and wider network based on LeNet network (LeCun et al., 1998), which was first introduced by LeCun in 1989.

The comparison of LeNet++ and LeNet network architecture can be seen in Table 2.1. LeNet++ consists of 3 convolution blocks, each block is made up of 2 cascaded convolution layers with different filters of size 5×5, where the stride and padding are 1 and 2 respectively. At the end of each convolution block, batch normalization (Ioffe and Szegedy, 2015) is applied to the output.

	Stage 1		Stage 2		Stage 3		Stage 4
Layer	Conv	Pool	Conv	Pool	Conv	Pool	FC
LeNets	$(5,20)_{/1,0}$	$2_{/2,0}$	$(5,50)_{/1,0}$	$2_{/2,0}$			500
LeNets++	$(5,32)_{/1,2} \times 2$	$2_{/2,0}$	$(5,64)_{/1,2} \times 2$	$2_{/2,0}$	$(5,128)_{/1,2} \times 2$	$2_{/2,0}$	**2**

Table 2.1: LeNet++ and LeNet Architectures. Some of the convolution layers are followed by max pooling. $(5,32)_{/1,2} \times 2$ denotes 2 cascaded convolution layers with 32 filters of size 5×5 where the stride and padding are 1 and 2 respectively. $2_{/2,0}$ denotes the max-pooling layers with grid of 2×2, where the stride and padding are 2 and 0 respectively. *Source*: Wen et al. (2016).

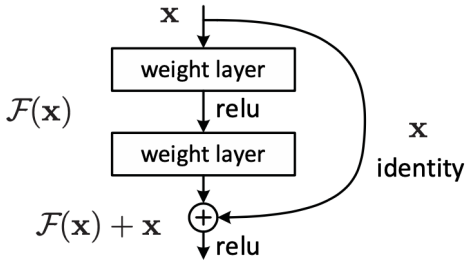


Figure 2.2: Residual block. *Source* : He et al. (2016)

Each convolution block is followed by a max-pooling layer with size 2×2, where the stride and padding are 2 and 0 respectively. After that, a Parametric Rectified Linear Unit (PReLU) (He et al., 2015) is used as the activation function. There is a 2 dimensional fully connected layer after the convolutional layers. On top of them, another fully connected layer is applied to extract the logit for calculating the probability of corresponding classes.

2.3.2 ResNet-18

Many neural network architectures have been introduced to recognize more complicated images. ResNet is one of the most powerful deep neural networks which has achieved great performance in the ILSVRC 2015 classification challenge[2].

DNNs can extract more features with enough stacked layers. Simonyan and Zisserman (2014) reveals that the network depth is crucial for achieving better results in challenging datasets. However, when deeper networks start converging, with increasing network depth, the accuracy gets saturated and degrades rapidly. To solve this degradation problem, He et al. (2016) introduced a deep residual learning framework, as shown in Figure 2.2. For a few stacked layers in the network, instead of directly fitting a desired underlying mapping $\mathcal{H}(\boldsymbol{x})$, they fit a residual mapping $\mathcal{F}(\boldsymbol{x}) := \mathcal{H}(\boldsymbol{x}) - \boldsymbol{x}$, under the assumption that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. The formulation $\mathcal{F}(\boldsymbol{x}) + \boldsymbol{x}$ can be realized by feedforwarding neural networks with "shortcut connections". Shortcut connections (Bishop et al., 1995) are those skipping one or more layers for fully-connected networks . In their case, the shortcut connections simply perform identity mapping for convolutional layers, and their outputs are added to the outputs of the stacked layers (Figure 2.2).

The ResNet-18 network architecture can be seen in Figure 2.3. It consists of 18 weighted layers.

[2] https://image-net.org/challenges/LSVRC/2015/

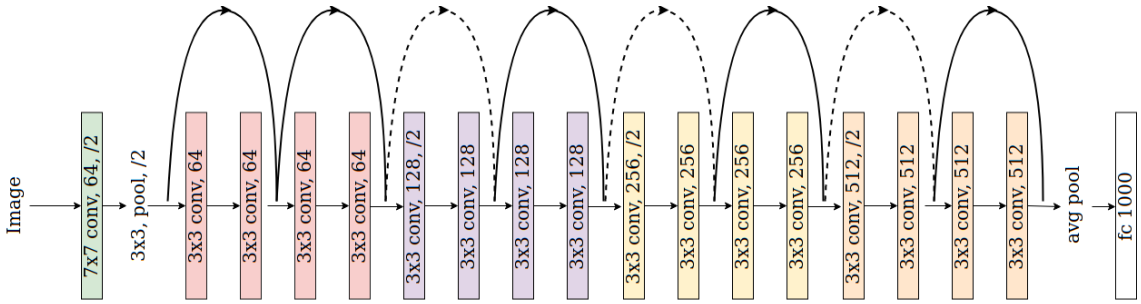


Figure 2.3: RESNET-18 NETWORK ARCHITECTURE. *Source* : He et al. (2016)

It first starts with one convolutional layer with 64 kernel size of 7×7, where the stride and padding are 2 and 3 respectively, then followed by one max pooling layer with filters of size 3×3, where the stride and padding are 2 and 1 respectively. And next, it followed by 4 blocks of residual layers with different number of filters (64, 128, 256, 512). Each block consists of 2 residual blocks, each residual block is made up of two convolutional layers with a filter of size 3×3. The network ends with a global average pooling layer and 1000-output fully-connected layer.

Chapter 3

Methodology

In our approach, we propose a new generative model (named as Combined-GAN) based on supervised GAN to generate unknown samples for OSR.

Provided known image pairs from different classes, e.g., $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2)\}(\boldsymbol{x_1}, \boldsymbol{x_2} \in \mathcal{D}_{\mathcal{KK}}, y_1 \neq y_2)$, we generate synthetic samples that are similar to $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ but they do not belong to any known class. Next, we augment the original known samples with the generated samples, which are considered as open-set samples. Lastly, we train an open-set model on the combined samples to tackle the OSR problem. We assume that, after the open-set model is trained on the generated samples and, it will detect the new unknown samples as well as classify known samples.

We assume that, the generated samples from our generative model can enable the open-set model to identify and reject unknown samples.

3.1 Model Architecture

Figure 3.1 shows an overview of our whole model architecture. The whole model consists of two sub models: our generative model Combined-GAN and an Open-Set Classifier C_{os}. Our generative model has three components: encoder G_{enc}, decoder G_{dec}, discriminator D (including a closed-set classifier C_{cs} and an activation function A (mentioned in Equation (2.7))).

We assume we have a pair of images $(\boldsymbol{x_1}, \boldsymbol{x_2})$ from different classes. Firstly, G_{enc} is employed to map original images into corresponding latent space, hence, we get 2 latent representations: $G_{enc}(\boldsymbol{x_1})$ and $G_{enc}(\boldsymbol{x_2})$ from $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ respectively. Further, we obtain 3 latent representations, $\Phi(\boldsymbol{x_1}, \boldsymbol{x_2})$, $\Phi(\boldsymbol{x_1})$ and $\Phi(\boldsymbol{x_2})$. $\Phi(\boldsymbol{x_1}, \boldsymbol{x_2})$ is concatenation of $G_{enc}(\boldsymbol{x_1})$ and $G_{enc}(\boldsymbol{x_2})$, while $\Phi(\boldsymbol{x_i})(i \in 1, 2)$ is simply created by concatenating $\Phi(\boldsymbol{x_i})$ with itself.

After obtaining latent representations, G_{dec} decodes the corresponding latent representations to new images. On the one hand, $\Phi(\boldsymbol{x_1})$ and $\Phi(\boldsymbol{x_2})$ are decoded to reconstructed images, which ensures the encoder provides the right deep feature from the original images; On the other hand, $\Phi(\boldsymbol{x_1}, \boldsymbol{x_2})$ is decoded to a fake unknown image which is used as an open-set sample.

Based on supervised GAN, we upgrade the discriminator D in original GAN (Goodfellow et al., 2014) with a C_{cs}, where output of C_{cs} is served as logits to predict class labels of the original images and reconstructed images, as well as the open set samples for supervised learning. Apart from that, an activation function A is added for D to identify the generated images as fake and original images as real.

After generating open-set samples, we train C_{os} to classify the known images and recognize fake images.

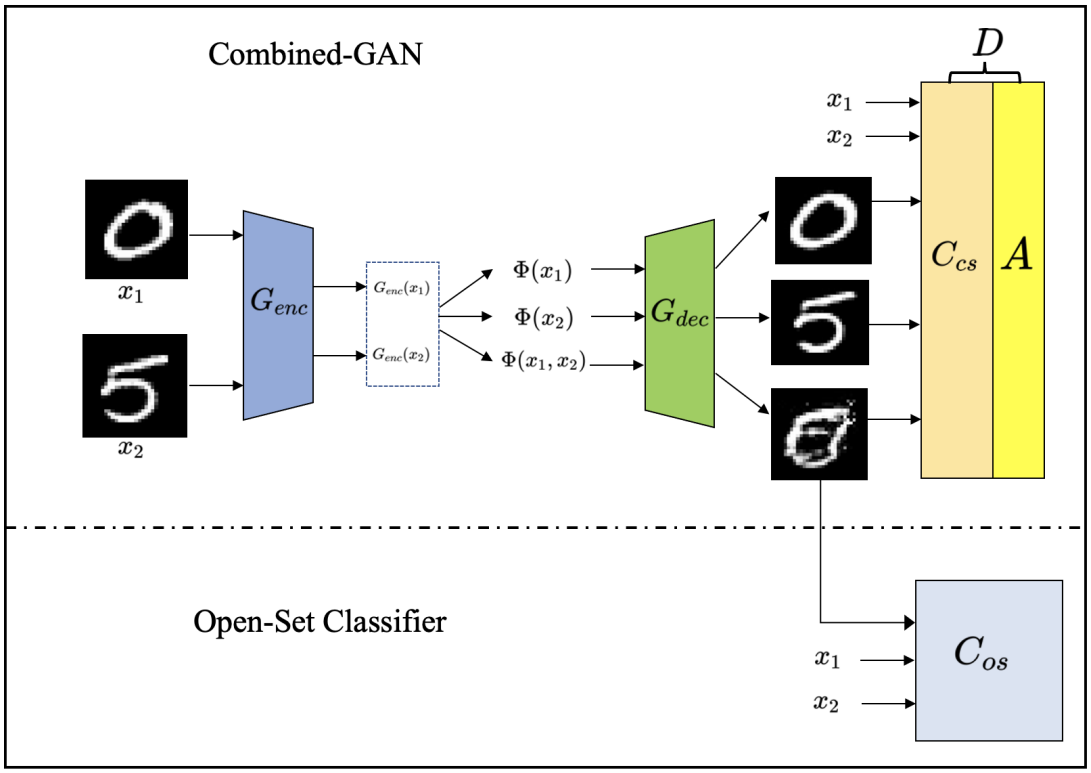


Figure 3.1: OVERVIEW OF THE WHOLE MODEL ARCHITECTURE. The whole model consists of two models: Combined-GAN model (on the top) and open-set model (in the bottom). The generative model has three components: encoder G_{enc}, decoder G_{dec}, discriminator D (including a closed-set classifier C_{cs} and an activation function A). Assume we have image pairs $(\boldsymbol{x_1}, \boldsymbol{x_2})$ from class "0" and class "5". From encoder-decoder generative model, reconstructed images from 2 classes and unknown images are generated. Then, C_{os} is trained on the known image samples and generated unknown image samples.

3.2 Loss Function for Generative Model

To generate effective open-set samples, we apply 4 different loss functions in our Combined-GAN model. The binary cross entropy loss is applied to guarantee the reconstructed images belong to the original classes. The mixed-class loss is used to classify the generated samples as unknown classes. The adversarial loss is used to ensure visual reality of the generated images. The reconstruction loss is used to ensure a specific image can be reconstructed to itself.

Binary cross entropy loss on the original images and reconstructed images. As mentioned above, C_{cs} is required to correctly classify the original images and reconstructed images, we use cross entropy loss to optimize C_{cs} for this task, the objective function can be formulated as,

$$l_{cls_b} = -\sum_{i=1}^{2} \mathbb{E}_{\boldsymbol{x_i} \sim p_{\boldsymbol{x_i}}} \sum_{k=1}^{K} t_k(\boldsymbol{x_i}) [\log p_C(k|\boldsymbol{x_i}) + \log p_C(k|G_{dec}(\Phi(\boldsymbol{x_i})))] \tag{3.1}$$

where p_C is the softmax score obtained from classifier C_{cs}. We use one-hot encoding method to encode the numerical class label y into binary target vectors $\boldsymbol{t(x)}$, where $t_k(\boldsymbol{x})$ indicates the kth index of $\boldsymbol{t(x)}$. One-hot encoding is defined as :

$$t_k(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad k = y \\ 0 & \text{otherwise} \end{cases}, \text{where} \quad 1 \leq k \leq K \tag{3.2}$$

Thus, l_{cls_b} is the sum of binary cross entropy loss of all original images and reconstructed images.

Cross entropy loss on the generated samples (mixed-class loss). As mentioned in Section 3.1, we expect the generated images have equal probability on two different classes of original images. The open-set samples are labeled with the target indicator $\boldsymbol{t(x_1, x_2)}$, which is a vector with K elements. We want to generate fake samples similar with the known image pairs $(\boldsymbol{x_1}, \boldsymbol{x_2})$. Correspondingly, the generated samples are expected to have the same probability on two known classes. In order to achieve this, we set the target indicator $\boldsymbol{t(x_1, x_2)}$ equal to the mean of the one-hot encoding of the corresponding known classes $(y_1, y_2)(y_1 \neq y_2)$, i.e., $\boldsymbol{t(x_1, x_2)} = \frac{\boldsymbol{t(x_1)} + \boldsymbol{t(x_2)}}{2}$. More specifically, the elements in $\boldsymbol{t(x_1, x_2)}$ will be 0 if the corresponding known classes are not involved, while the elements related to the known classes will be 0.5. The mixed-class loss can be formulated as,

$$l_{cls_m} = -\mathbb{E}_{(\boldsymbol{x_1}, \boldsymbol{x_2}) \sim p_{(\boldsymbol{x_1}, \boldsymbol{x_2})}} \sum_{k=1}^{K} t_k(\boldsymbol{x_1}, \boldsymbol{x_2}) \log p_C(k|G_{dec}(\Phi(\boldsymbol{x_1}, \boldsymbol{x_2}))) \tag{3.3}$$

where $p_{(\boldsymbol{x_1}, \boldsymbol{x_2})}$ is the joint distribution, l_{cls_m} indicates the sum of cross entropy losses on all image pairs.

Adversarial Loss. The adversarial learning between the generator and discriminator is introduced to make the generated images more realistic. The discriminator is expected to classify the original images as real and classify the reconstructed images as fake. Following the WGAN-GP, the adversarial losses can be written as the following formula:

$$\min_{G_{enc}, G_{dec}} \max_{\|D\|_L \leq 1} \sum_{i=1}^{2} \mathbb{E}_{\boldsymbol{x_i} \sim p_{\boldsymbol{x_i}}} [D(\boldsymbol{x_i}) - D(G_{dec}(\Phi(\boldsymbol{x_i})))] \tag{3.4}$$

To keep consistent with other minimization functions, we turn the min-max function into a min-min function. Therefore, the adversarial loss for the discriminator and generator are formulated as below:

$$\min_{\|D\|_L \leq 1} l_{adv_d} = \sum_{i=1}^{2} \mathbb{E}_{\boldsymbol{x_i} \sim p_{\boldsymbol{x_i}}} [-D(\boldsymbol{x_i}) + D(G_{dec}(\Phi(\boldsymbol{x_i})))] \tag{3.5}$$

$$\min_{G_{enc},G_{dec}} l_{adv_g} = -\sum_{i=1}^{2} \mathbb{E}_{\boldsymbol{x_i}\sim p_{\boldsymbol{x_i}}} D(G_{dec}(\Phi(\boldsymbol{x_i}))) \tag{3.6}$$

Reconstruction Loss. Inspired by the use of reconstruction loss to regularize the training of generator to avoid mode collapse in Berthelot et al. (2017) and He et al. (2019), we also implement reconstruction loss in our generative model to ensure that a specific image can be reconstructed. The reconstructed loss l_{rec} computes the sum of L1 loss between the reconstructed images and original images,

$$l_{rec} = \sum_{i=1}^{2} \mathbb{E}_{\boldsymbol{x_i}\sim p_{\boldsymbol{x_i}}} \|G(\Phi(\boldsymbol{x_i})) - \boldsymbol{x_i}\|_1 \tag{3.7}$$

By combining the 4 loss functions mentioned above, our Combined-GAN model is expected to generate effective open-set samples. We consider G_{enc} and G_{dec} as one component, C_{cs} and D as another component, and we train them alternatively.

1) **Overall loss function for C_{cs} and D**

The overall loss function of C_{cs} and D combines binary cross entropy loss, adversarial loss and mixed-class loss. It is formulated as below,

$$\min_{C,D} \quad l_{cls_b} + l_{adv_d} + \lambda_1 \cdot l_{cls_m} \tag{3.8}$$

where λ_1 is the hyperparameter for balancing the losses.

2) **Overall loss function for G_{enc} and G_{dec}**

The training objective function for G_{enc} and G_{dec} is based on the combination of reconstruction loss, adversarial loss and mixed-class loss, which is formulated as,

$$\min_{G_{enc},G_{dec}} \quad \lambda_2 \cdot l_{cls_m} + \lambda_3 \cdot l_{rec} + l_{adv_g} \tag{3.9}$$

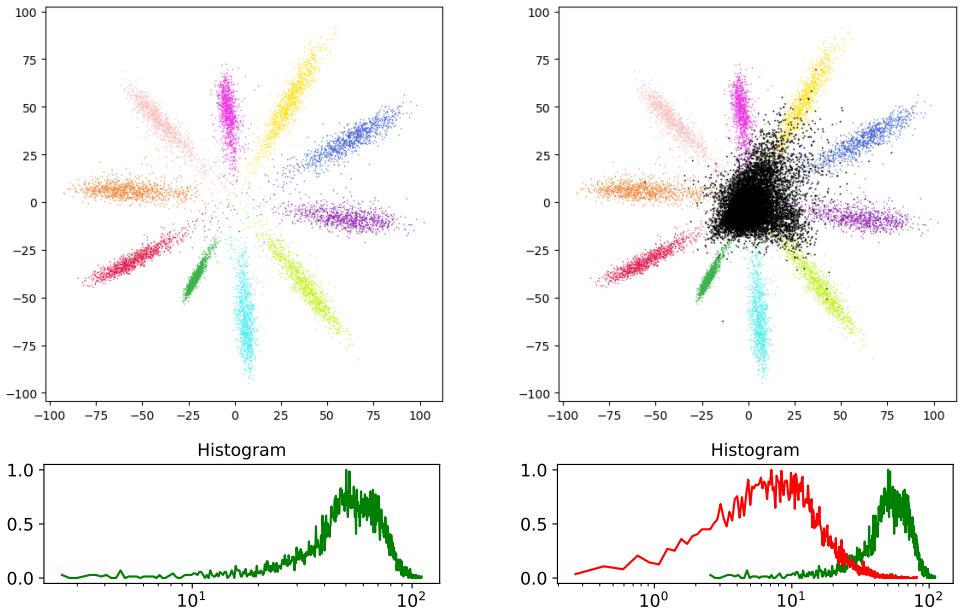
where λ_2 and λ_3 are the hyperparameters for balancing the losses.

3.3 Open-Set Classifier

3.3.1 Network Architecture and Loss Function

We adopt LeNet++ (Wen et al., 2016) as the network architectures for open-set classifier. In addition, we modify the ResNet-18 (He et al., 2016) by adding a 2 dimensional fully-connected layer before the last layer to extract deep features of input images. We also change the dimension of output in the last layer based the need of our experiments. We name the 2-D adaptive ResNet-18 network as ResNet-18++ network, which will be used in our experiments.

After fake samples are generated from our generative model, they are labeled as open-set samples for OSR. Since EOS loss (introduced in Section 2.1.2) can be used to improves the handling of unknown inputs, we use EOS loss to train the open-set classifier C_{os} on open-set samples and known samples.



(a) $D_{\mathcal{KK}}$ (b) $D_{\mathcal{KK}}$ + $D_{\mathcal{UU}}$

Figure 3.2: LeNet++ responses to $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$. 2-D feature visualization plots are displayed on the top. Colored dots represent known samples, and each color represents one known class, while the black dots represent unknown samples. The plots in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$.

3.3.2 Deep Feature Visualization

To understand the distribution of known samples and unknown samples, we provide 2-D visualization for images samples. The 2-D feature representations are obtained from the penultimate layer of LeNet++ and ResNet-18++ network after convolutional feature learning. Hence, they can be directly plotted on a 2-D plane for visualization and analysis.

Here, we provide an example using LeNet++ (more detailed visualization using LeNet++ and ResNet-18++ will be provided in Chapter 5). First LeNet++ is used as closed-set classifier and trained to classify $D_{\mathcal{KK}}$ (MNIST (detail in Section 4.1.1)) using softmax loss. Then, LeNet++ is tested on $D_{\mathcal{KK}}$ and $D_{\mathcal{UU}}$ (Fashion MNIST (detail in Section 4.1.1)), feature representations are obtained from LeNet++. Finally, we plot the feature representations on 2-D plane.

In Figure 3.2(a), samples from $D_{\mathcal{KK}}$ are mapped to 10 patterns like a flower with 10 petals. The deep features of different classes are distinguished by decision boundaries, while the feature magnitude can reflect the confidence to be classified as a certain class, and the larger feature magnitude indicates stronger confidence. In Figure 3.2(b) unknown samples are fed into LeNet++, they scatter without patterns and share a large overlap with $\mathcal{D}_{\mathcal{KK}}$.

Additionally, we also plot the histogram of feature magnitude by measuring the Euclidean length of feature representation, the x-axis indicates feature magnitude while y-axis indicates the normalized number of samples having corresponding feature magnitude. From the histogram we can see clearly that quite a large number of the unknown samples have large feature magnitude, which results in high softmax score on one of the known classes. Thus, it makes LeNet++ hard to recognize known samples from unknown classes.

Chapter 4

Experimental Setup

4.1 Dataset

In this section, we introduce several benchmark datasets used in our experiments and describe how we manipulate them for the experiments. By training our models on different datasets, it is worth finding how well the open-set models classify the known classes and detect unknown classes in different scenarios when they are trained on different datasets using different approaches.

4.1.1 Benchmark Datasets

We conduct experiments using MNIST, KMNIST, FMNIST, EMNIST, CIFAR-10, CIFAR-100 and SVHN datasets. All these datasets are provided by PyTorch[1]. Some samples are shown in Figure 4.1.

MNIST (LeCun et al., 2010) has a total of 70 000 gray images of 10 digit classes, where each class contains between 6313 and 7877 monochrome images with 28×28 feature dimension. The dataset is divided into a training set and a testing set containing 60 000 and 10 000 images respectively.

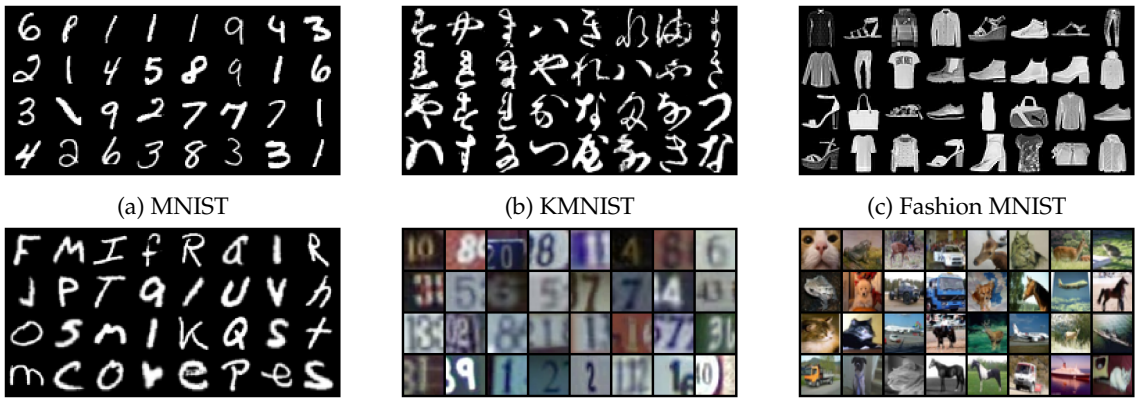
Kuzushiji-MNIST (KMNIST) (Clanuwat et al., 2018) is a dataset which focuses on Kuzushiji (cursive Japanese characters). It serves as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. KMNIST dataset consists of a training set of 60 000 examples and a test set of 10 000 examples. Each example is a 28×28 grayscale image, associated with a label from 10 classes.

Fashion-MNIST (FMNIST) (Xiao et al., 2017) is a dataset of Zalando's article images consisting of a training set of 60 000 images and a test set of 10 000 images. Each example is a 28×28 grayscale image, associated with a label from 10 classes. Zalando intends FMNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

EMNIST (Cohen et al., 2017) is a set of handwritten characters derived from the NIST Special Database 19[2] and converted to a 28×28 pixel image format and dataset structure that directly matches the MNIST dataset. There are six different splits provided in this dataset. We only use EMNIST letters for training and testing. The EMNIST letters dataset merges a balanced set of the uppercase and lowercase letters into a single 26-class task. In total, EMNIST letters dataset contains 145 600 characters from 26 balanced classes.

[1] https://pytorch.org/

[2] https://www.nist.gov/srd/nist-special-database-19



(a) MNIST (b) KMNIST (c) Fashion MNIST

(d) EMNIST Letters (e) SVHN (f) CIFAR-10

Figure 4.1: SAMPLES FROM DIFFERENT DATASETS.

CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton, 2009): The CIFAR-10 dataset consists of 60 000 32×32 color images in 10 classes, with 6000 images per class. The dataset is divided into a training set and a testing set containing 50 000 and 10 000 images respectively.

The CIFAR-100 dataset is similar to the CIFAR-10 dataset, differently it has 100 classes and contains 600 images for each class. There are 500 training images and 100 testing images per class. The 100 classes in CIFAR-100 dataset are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

Street View House Numbers (SVHN) (Netzer et al., 2011) is a digit classification benchmark dataset that contains 73 257 digits for training, 26 032 digits for testing, and 531 131 additional training data. We only use the training and testing data. The dataset includes 32×32 RGB images of printed digits cropped from pictures of house number plates. There are 10 classes for 10 digits. Digit 1 has label "1", 9 has label "9" and digit 10 has label "0". However, the label "0" is assigned as the digit 0 to be compatible with PyTorch loss functions which expect the class labels to be in the range $[0, K-1]$.

4.1.2 Data Splitting

For OSR, most researchers split the same dataset by choosing a part of the class labels as $\mathcal{KK}$ while the remaining classes as $\mathcal{UU}$. In our point of view, the data splitting is not realistic, because the unknown data can be totally different from the known data in real-world. In our experiments, we choose different datasets as $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$.

In total, we have 6 data sets for experiments, every set includes three subsets: a training dataset, a validation dataset and a testing dataset, as shown in Figure 4.2. The detailed data splitting infomation is listed in Table 4.1. To test the effectness of our generative model and robustness of the open-set model, we select multiple datasets as $\mathcal{D}_{\mathcal{UU}}$ for every set.

In Set 1, we use MNIST digits as $\mathcal{D}_{\mathcal{KK}}$ and a subset of EMNIST letters (A-M) as $\mathcal{D}_{\mathcal{KU}}$ during training and validation, and subsequently we test on another subset of EMNIST letters (N-Z) (used as $\mathcal{D}_{\mathcal{UU}}$). In addition, we also test on KMNIST and FMNIST datasets as $\mathcal{D}_{\mathcal{UU}}$. Similarly, in Set 2 and Set 3, KMNIST and FMNIST are labeled as $\mathcal{KK}$ respectively, while MNIST is served as $\mathcal{D}_{\mathcal{KU}}$. In Set 4, similar with Set 1, MNIST is used as $\mathcal{D}_{\mathcal{KK}}$, but here KMNIST is used as $\mathcal{D}_{\mathcal{KU}}$ for training and validation. Then we test on 2 different testing datasets, containing EMNIST letters

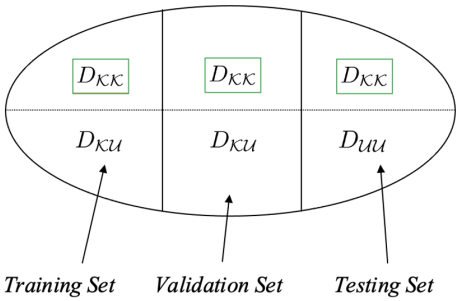


Figure 4.2: DATA SPLITTING METHOD. The whole oval represents the OSR scenario, where the training set and validation set contain $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{KU}}$, while the testing set is made up of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$. The upper oval represents the CSR scenario, in which all the sets only contain $\mathcal{D}_{\mathcal{KK}}$ (outlined in green color).

Set	Training dataset	Validation dataset	Testing dataset
1	MNIST(50k) + Subset of EMNIST Letter(52k)	MNIST(10k) + Subset of EMNIST Letter(10.4k)	MNIST(10k) + Subset of EMNIST Letter(10.4k)
			MNIST(10k) + FMNIST(10k)
			MNIST(10k) + KMNIST(10k)
2	KMNIST(50k) + MNIST(50k)	KMNIST(10k) + MNIST(10k)	KMNIST(10k) + EMNIST Letter(20.8k)
			KMNIST(10k) + FMNIST(10k)
3	FMNIST(50k) + MNIST(50k)	FMNIST(10k) + MNIST(10k)	FMNIST(10k) + EMNIST Letter(20.8k)
			FMNIST(10k) + KMNIST Letter(10k)
4	MNIST(50k) + KMNIST(50k)	MNIST(10k) + KMNIST(10k)	MNIST(10k) + EMNIST Letter(20.8k)
			MNIST(10k) + FMNIST(10k)
5	CIFAR-10(50k) + Subset of CIFAR-100(18.75k)	CIFAR-10(10k) + Subset of CIFAR-100(3.75k)	CIFAR-10(10k) + Subset of CIFAR-100(4.5k)
			CIFAR-10(10k) + SVHN(~26k)
6	SVHN(~61k) + Subset of CIFAR-100(18.75k)	SVHN(~12.2k) + Subset of CIFAR-100(3.75k)	SVHN(~26k) + Subset of CIFAR-100(4.5k)

Table 4.1: DATA SPLITTING DETAIL. $\mathcal{D}_{\mathcal{KK}}$ is labeled in green color, $\mathcal{D}_{\mathcal{UU}}$ in orange color and $\mathcal{D}_{\mathcal{UU}}$ in red color. The corresponding dataset size is given in the brackets.

and FMNIST as $\mathcal{D}_{\mathcal{UU}}$.

In Set 5, we use the CIFAR-10 dataset as $\mathcal{D}_{\mathcal{KK}}$, and we randomly select 45 subclasses from CIFAR-100 dataset as $\mathcal{D}_{\mathcal{KU}}$. Correspondingly, we select another 45 subclasses from CIFAR-100 testing dataset as $\mathcal{D}_{\mathcal{UU}}$ for testing. So in this case, $\mathcal{KU}$ and $\mathcal{UU}$ are mostly from same superclasses. In addition, we also test on SVHN dataset as $\mathcal{D}_{\mathcal{UU}}$.

Similarly with Set 5, In Set 6, we also use the subset of CIFAR-100 as $\mathcal{D}_{\mathcal{UU}}$ and $\mathcal{D}_{\mathcal{UU}}$ and SVHN dataset as $\mathcal{D}_{\mathcal{KK}}$.

4.2 Generative Network Architecture and Training Details

Network		Layers	Input(C, H, W)	Output(C, H, W)
Encoder		Conv(32, 6, 2, 0), BN, Leaky ReLU	(1, 28, 28)	(64, 12, 12)
		Conv(128, 6, 2, 0), BN, Leaky ReLU	(64, 12, 12)	(128, 4, 4)
		Conv(256, 4, 2, 0), BN, Leaky ReLU	(128, 4, 4)	(256, 1, 1)
Decoder		DeConv(128, 4, 2, 0), BN, ReLU	**(256*2, 1, 1)**	(128, 5, 5)
		DeConv(32, 6, 2, 0), BN, ReLU	(128, 5, 5)	(32, 12, 12)
		DeConv(1, 6, 2, 0), Tanh	(32, 12, 12)	(1, 28, 28)
Discriminator	**Closed-Set Classifier**	Conv(32, 6, 2, 0), BN, Leaky ReLU	(1, 28, 28)	(64, 12, 12)
		Conv(128, 6, 2, 0), BN, Leaky ReLU	(64, 12, 12)	(128, 4, 4)
		Conv(256, 4, 2, 0), BN, Leaky ReLU	(128, 4, 4)	(256, 1, 1)
		Conv(10, 1, 1, 0), (Softmax)	(256, 1, 1)	10
	Activation	Activation function	10	1

(a) Network architecture of Combined-GAN for 28×28 grayscale images

Network		Layers	Input(C, H, W)	Output(C, H, W)
Encoder		Conv(64, 4, 2, 1), BN, Leaky ReLU	(3, 32, 32)	(64, 16, 16)
		Conv(128, 4, 2, 1), BN, Leaky ReLU	(64, 16, 16)	(128, 8, 8)
		Conv(256, 4, 2, 1), BN, Leaky ReLU	(128, 8, 8)	(256, 4, 4)
		Conv(512, 4, 2, 0), BN, Leaky ReLU	(256, 4, 4)	(512, 1, 1)
Decoder		DeConv(256, 4, 2, 0), BN, ReLU	**(512*2, 1, 1)**	(256, 4, 4)
		DeConv(128, 4, 2, 1), BN, ReLU	(256, 4, 4)	(128, 8, 8)
		DeConv(64, 4, 2, 1), BN, ReLU	(128, 8, 8)	(64, 16, 16)
		DeConv(3, 4, 2, 1), Tanh	(64, 16, 16)	(3, 32, 32)
Discriminator	**Closed-Set Classifier**	Conv(64, 4, 2, 1), BN, Leaky ReLU	(3, 32, 32)	(64, 16, 16)
		Conv(128, 4, 2, 1), BN, Leaky ReLU	(64, 16, 16)	(128, 8, 8)
		Conv(256, 4, 2, 1), BN, Leaky ReLU	(128, 8, 8)	(256, 4, 4)
		Conv(512, 4, 2, 0), BN, Leaky ReLU	(256, 4, 4)	(512, 1, 1)
		Conv(10, 1, 1, 0), (Softmax)	(512, 1, 1)	10
	Activation	Activation function	10	1

(b) Network architecture of Combined-GAN for 32×32 color images

Table 4.2: NETWORK ARCHITECTURE OF COMBINED-GAN. Conv(d,k,s,p) and DeConv(d,k,s,p) denote the convolutional layer with d as dimension, k as kernel size, s as stride and p as padding. BN is batch normalization. C, H, W denotes the the number of channels, height and weight of the input and output for each layer. After obtaining the latent representations from encoder, we concatenate the latent representations (the concatenation is labeled in bold font) for generating new images. (softmax) means that, if we use closed-set classifier to predict the class labels, we apply the softmax activation function on the output of last convolutional layer, while if we use discriminator to predict real or fake, the softmax layer is removed and the activation function is directly applied on the output of last convolutional layer.

4.2.1 Generative Network Architecture

Our Combined-GAN is made up of three components: encoder G_{enc}, decoder G_{dec} and discriminator D (including a closed-set classifier C_{cs} and an activation function A). Table 4.2 shows two versions of detailed generative network architectures for different input image sizes. Our Combined-GAN network architecture heavily borrows from DCGAN (Radford et al., 2016) and AttGAN (He et al., 2019). We adopt most of the architecture guidelines for training GAN in Radford et al. (2016). G_{enc} is a stack of convolutional layers, for every convolutional layer, batch normalization and Leaky ReLU are applied to the output. G_{dec} is a stack of transposed convolutional layers, we use ReLU activation for all layers except for the output of last layer, which uses Tanh. D shares a similar architecture with G_{enc}, and they both contain a stack of convolutional layers. The difference is that D has two more layers to do two different supervised learning tasks. The penultimate layer of D is used as C_{cs} to predict class labels using softmax loss. Additionally an activation function mentioned in Equation (2.7) is applied on the logits of C_{cs} to predict real and fake.

For the small version generative model (Table 4.2a), a series of four fractionally-strided convolutions convert 28×28 pixel images into latent representations, and then the latent representations are fed into a series of four strided convolutions and decoded to 28×28 pixel images again.

4.2.2 Training details

All models are trained with a mini-batch size of 64. For convolutional layers, all weights are initialized from a zero-centered normal distribution with standard deviation 0.02. In the LeakyReLU, the slope of the leak is set to 0.2 in all models.

For Combined-GAN model, we use Adam optimizer (Kingma and Ba, 2014) (β_1 = 0.5, β_2 = 0.999) for tunning hyperparameters, and the learning rate is set as 0.0002. The coefficients for the losses in Equation (3.8) and Equation (3.9) are set as $\lambda_1 = 50$, $\lambda_2 = 1$, $\lambda_3 = 100$.

For open-set classifier, we use stochastic gradient descent algorithm for optimization. We use different learning rate for different models (0.01 for LeNet++ and 0.005 for ResNet-18++).

4.3 Evaluation Metrics

We borrow some commonly used evaluation metrics like confidence, AUC and adapt them for OSR. Besides, we also use Open-Set Classification Rate (OSCR) to evaluate models. Based on OSCR, we also provides AUOC score to evaluate the overall performance of the open-set model.

4.3.1 Confidence

In CSR, confidence is used to measure the possibility of a known sample to be classified as the correct class. For OSR, the unknown sample does not have correct class, the unknown sample is expected to be classified as any known class with low probability. Thus, to measure how confident the open-set model classify the $\mathcal{KK}$ and reject $\mathcal{UU}$, confidece is defined in the following way:

$$confidence(x) = \begin{cases} S_k(x) & \text{if } x \in \mathcal{D}_{\mathcal{KK}} \text{ is from class } k \\ 1 + \frac{1}{K} - \max_k(S_k(x)) & \text{if } x \in \mathcal{D}_{\mathcal{UU}} \end{cases} \tag{4.1}$$

For known samples, confidence is measured by computing the softmax score of the correct class. For unknown samples, since they do not belong to any known class, we want to minimize the softmax scores $S_k(x)$ for all known classes. In other words, we want the maximum entropy

distribution of uniform probabilities over all known classes. This is achieved when $S_k(x) = \frac{1}{K}, \forall k \in K$. In this case, minimum of $\max_k(S_k(x))$ is achieved and thus the open-set model has 100% confident in rejecting the unknown samples.

4.3.2 Area Under the Curve (AUC)

AUC is an evaluation metric commonly used in research. First, ROC Curve is created by plotting the True Positive Rate (TPR) verse FPR. Then AUC score is computed based on ROC Curve. Since in our case, it is a multi-class OSR problem, we regard all samples from known classes as positive samples, the remaining ones as negative samples.

For evaluation, we split the testing samples into $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$. Let θ be a threshold value. For $\mathcal{D}_{\mathcal{KK}}$, TPR is calculated as the fraction of known samples where the correct class $\hat{k} \in K$ has a probability greater than θ, as shown in Equation 4.2.

$$\mathrm{TPR}(\theta) = \frac{\left|\left\{x | x \in \mathcal{D}_{\mathcal{KK}} \wedge P(\hat{k}|x) > \theta\right\}\right|}{|\mathcal{D}_{\mathcal{KK}}|} \tag{4.2}$$

Since in the OSR scenario, we do not have explicit probability for unknown class. For the unknown samples, a low probability of being classified as any known classes is expected. Thus, FPR is computed as the fraction of samples from $\mathcal{D}_{\mathcal{UU}}$ that are classified as any known class k with a probability greater than θ:

$$\mathrm{FPR}(\theta) = \frac{|\{x | x \in \mathcal{D}_{\mathcal{UU}} \wedge \max_k P(k|x) \geq \theta\}|}{|\mathcal{D}_{\mathcal{UU}}|} \tag{4.3}$$

After obtaining TPR and FPR, we can calculate the AUC score. Higher AUC score indicates that the open-set model is better at separating the known data from unknown data in a better way.

4.3.3 OSCR

OSCR is introduced in Dhamija et al. (2018) to assess open-set model. Let θ be a threshold, FPR is defined in Equation (4.3). For the $\mathcal{D}_{\mathcal{KK}}$, CCR is calculated as the fraction of the samples where the correct class $\hat{k} \in K$ has the maximum probability and the probability is greater than θ.

$$\mathrm{CCR}(\theta) = \frac{\left|\left\{x | x \in \mathcal{D}_{\mathcal{KK}} \wedge \arg\max_k P(k|x) = \hat{k} \wedge P(\hat{k}|x) > \theta\right\}\right|}{|\mathcal{D}_{\mathcal{KK}}|} \tag{4.4}$$

As we can see from Equation (4.4) and Equation (4.2), the difference between OSCR and ROC mentioned in Section 4.3.2 is that, for the known classes, OSCR not only calculates the probability of the known classes, but also ensures that the sample is classified as the correct class. Thus, OSCR is more accurate for the known classes recognition while separating the known classes and unknown classes. OSCR curve displays corresponding CCR values regarding different TPR values. Since we are more interested in lower FPR, we also compute CCR values at different FPR of interest to evaluate the open-set model.

Similar to the concept of AUC, we compute the Area Under the OSCR curve (AUOC). In this case, AUOC represents the area under the OSCR curve. Higher AUOC score means the model can better correctly classify $\mathcal{D}_{\mathcal{KK}}$ and reject $\mathcal{D}_{\mathcal{UU}}$ simultaneously. In our experiments, we use AUOC score to select best models during validation process.

Specifically, from $\sum_{k=1}^{K} P(k|x) <= K \cdot \max_k P(k|x)$, we can derive $\forall k \in K : \max_k P(k|x) >= \frac{1}{K}$, from Equation (4.3), we can see the the smallest value of θ is $\frac{1}{K}$. In this case, FPR equals 1, and the corresponding CCR value is identical to the closed-set classification accuracy on $\mathcal{D}_{\mathcal{KK}}$.

Chapter 5

Results

In this chapter, we assess and compare our approach with other approaches on various benchmark datasets mentioned in Section 4.1 for two different types of network architectures.

First, LeNet++ network is trained and tested on grayscale image datasets (Set 1-4), where MNIST digits, KMNIST characters and Fashion-MNIST clothes images are used as $\mathcal{D}_{\mathcal{K}\mathcal{K}}$, the data splitting detail can be seen in Table 4.1. Besides, we also evaluate 2-D adaptive ResNet-18++ (Section 3.3) on color image datasets (Set 5 and Set 6), where CIFAR-10 and SVHN are used as $\mathcal{D}_{\mathcal{K}\mathcal{K}}$.

For every experiment, we select $\mathcal{D}_{\mathcal{U}\mathcal{U}}$ which is totally different from $\mathcal{D}_{\mathcal{K}\mathcal{K}}$ to test the effectiveness of our Combined-GAN model. We assess and compare the performance of different approaches by using evaluation metrics mentioned in Section 4.3, and provide some quantitative results. In addition, we also provide visual results to show how open-set models response to the known and unknown samples in 2-D space.

5.1 Approaches

All approaches can be classified as the ones using softmax threshold (SM) and the ones using EOS loss mentioned in Section 2.1.2. For all approaches except SM, some unknown samples are labeled as $\mathcal{D}_{\mathcal{K}\mathcal{U}}$ during training.

Our approach: EOS with generated samples using Combined-GAN (EOS+Combined-GAN) In our approach, we generate unknown samples from known dataset $\mathcal{D}_{\mathcal{K}\mathcal{K}}$ on the fly by Combined-GAN. All the generated images are labeled as $\mathcal{D}_{\mathcal{K}\mathcal{U}}$, the original dataset is augmented with all generated samples for training an open-set model.

Plain Softmax (SM) We compare our approach to a standard confidence-based method using softmax threshold. In this approach, a closed-set classifier is trained only on known classes and provides class prediction on known classes. For the purpose of open-set detection, unknown samples are detected by thresholding the max softmax score of all known classes.

Plain Softmax with Background (SM+BG) In this background approach, we label samples from the background class (not $\mathcal{K}\mathcal{K}$) as $\mathcal{K}\mathcal{U}$, then a closed-set classifier is trained on $\mathcal{D}_{\mathcal{K}\mathcal{K}}$ and $\mathcal{D}_{\mathcal{K}\mathcal{U}}$ using softmax loss. Different from SM approach, the dimension of the logit layer of the closed-set classifier increases by 1, the additional output is directly used to predict $\mathcal{D}_{\mathcal{K}\mathcal{U}}$ as another class.

EOS with Background (EOS+BG) In this approach, similar with SM+BG, we also label the background class as $\mathcal{K}\mathcal{U}$. Then an open-set model is trained on $\mathcal{D}_{\mathcal{K}\mathcal{U}}$ and $\mathcal{D}_{\mathcal{K}\mathcal{U}}$ using EOS loss.

EOS with Noise (EOS+Noise) In this approach, we generate noise images which have same image size as input known images and each pixel of noise images is randomly chosen from 0 to 1, then the noise images are labeled as $\mathcal{D}_{\mathcal{K}\mathcal{U}}$. Afterwards, an open-set model is trained on the noise samples and $\mathcal{D}_{\mathcal{K}\mathcal{K}}$ using EOS loss.

EOS with adversarial samples using FGSM (EOS+FGSM) For comparison, we use FGSM method (Goodfellow et al., 2015) to generate adversarial samples that look similar to original samples but can be easily classified as wrong classes. The FGSM works by using the gradients of the neural network to create an adversarial example. For an input image x, the method uses the gradients of the loss J with respect to the input image to create a new image x_adv that maximises the loss. This new image is called the adversarial sample. FGSM can be summarised using the following expression:

$$x\_adv = x + \epsilon * \text{sign}\left(\nabla_x J(\theta, x, y)\right) \tag{5.1}$$

where θ is model parameters, y is original input label, ϵ is multiplier on perturbations. In our experiments, we simply set ϵ as 0.1 (optimizing parameters in FGSM is out of the scope for this thesis).

After the adversarial samples are generated by FGSM, they are labeled as $\mathcal{D}_{\mathcal{KU}}$. Then, an open-set model is trained on $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{KU}}$ using EOS loss.

5.2 LeNet++ for OSR

Several experiments are done to train LeNet++ on Set 1-4 (Section 4.1.2) for OSR. From these experiments, some research questions need to be validated:

- Compared with other apporaches, can our Combined-GAN model generate effective open-set samples for different OSR secenarios (testing on different unknown samples)?
- Can LeNet++ be used to provide good 2-D deep feature representations for different datasets and how does LeNet++ response to known and unknown samples for different approaches?

5.2.1 Experiment 1: Training on MNIST

Quantitative Analysis

In this experiment, we train LeNet++ to classify MNIST digits and reject samples from unknown classes. For the background approach using SM and EOS, we use a subset of EMNIST letters (A-M) (more detail in Section 4.1.2) as the background class. We also use the subset of EMNIST letters (A-M) as $\mathcal{D}_{\mathcal{KU}}$ in the validation process. Then, another subset of EMNIST letters (N-Z) is used as $\mathcal{D}_{\mathcal{UU}}$ for testing. In addition, we also test on KMNIST and FMNIST datasets as $\mathcal{D}_{\mathcal{UU}}$. Thus, we evaluate the performance of different approaches by testing on the following three different datasets:

- MNIST ($\mathcal{D}_{\mathcal{KK}}$) + A subset of EMNIST letters (N-Z) ($\mathcal{D}_{\mathcal{UU}}$)
- MNIST ($\mathcal{D}_{\mathcal{KK}}$) + KMNIST ($\mathcal{D}_{\mathcal{UU}}$)
- MNIST ($\mathcal{D}_{\mathcal{KK}}$) + FMNIST ($\mathcal{D}_{\mathcal{UU}}$)

Figure 5.1 shows the OSCR results when we test LeNet++ on EMNIST letters (N-Z) and KMNIST, the result of testing on FMNIST can be found in Figure A.2.

From the OSCR curves, our approach gets relatively higher OSCR curves compared with other approaches for both cases. Correspondingly, the bar charts shows our approach almost achieves higher CCR values regarding all FPR values of interest. Usually we focus on the CCR values with small FPR values, but we also provide CCR value when FPR equals 1, in this case, CCR is the closed-set accurancy on $\mathcal{D}_{\mathcal{KK}}$. As a supplement, we also provide other evaluation results in Figure 5.2 regarding the evaluation metrics mentioned in Section 4.3.

When testing on subset of EMNIST letters (N-Z) as $\mathcal{D}_{\mathcal{UU}}$, since EMNIST letters can share some similar structure of different letters(e.g., "D" ($\mathcal{KU}$) and "O" ($\mathcal{UU}$), "M" ($\mathcal{KU}$) and "N" ($\mathcal{UU}$)), EOS+BG approach gets good results among all approaches, and EOS+BG can also classifiy the known and unknown samles with highest confidence (Figure 5.2(a)). The OSCR results of our approach is comparable with EOS+BG approach, as shown in Figure 5.1(a). The possible reason is that the shape of digits and letters can be similar, e.g., "9" and "g", "0" and "o", "1" and "j", etc. The generate unknown samples from MNIST digits can effectively capture the feature related to EMNIST letters. Thus, the open-set classifier can easily recognize and reject the letters. Correspondingly, our approach gets better performance regarding AUC and AOUC scores, as shown in Figure 5.2(a).

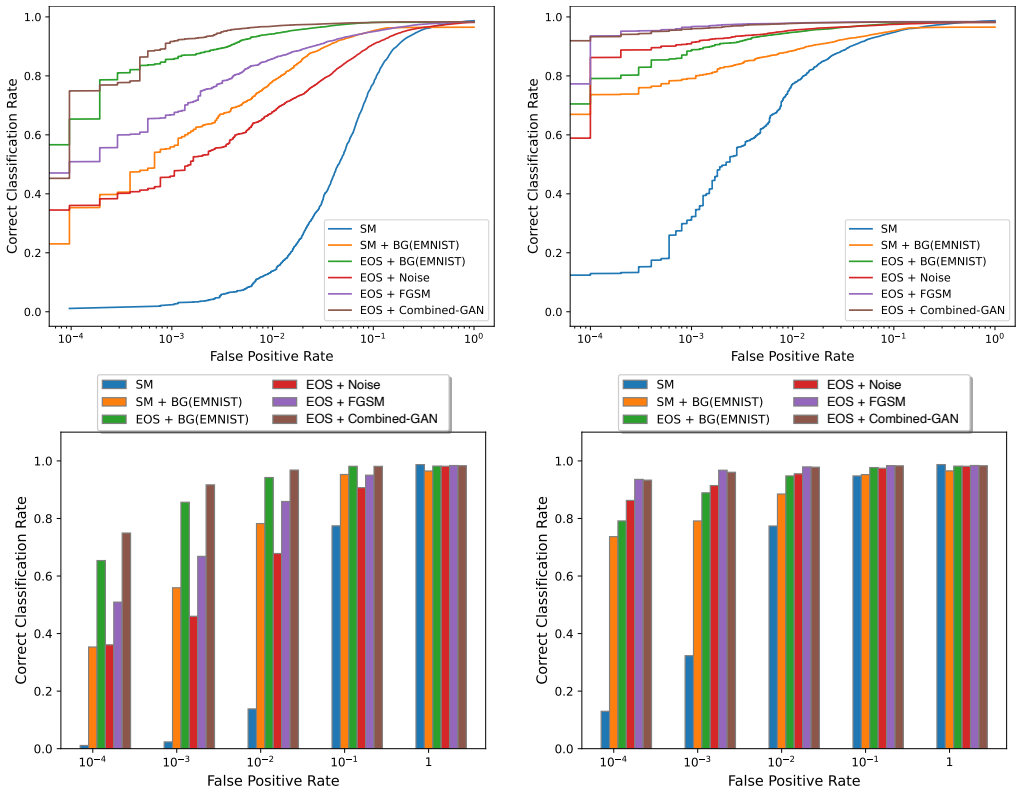
When testing KMNIST as $\mathcal{D}_{\mathcal{UU}}$, since KMNIST is far different from MNIST, it can be easily detected with high confidence (Figure 5.2(b)) by other methods, like EOS+Noise and EOS+FGSM. Compared with other approaches, our approach almost achieve best results on OSCR (Figure 5.1(b)) and other evaluation metrics(Figure 5.2(b)).

From the OSCR results, we can find that for the two unknown testing datasets, all approaches can almost achieve 100% closed-set accuracy. However, when FPR values become smaller, the CCR values of SM drop dramatically. Comparing with BG approaches, EOS+BG is much better than SM+BG. This indicates that EOS loss is more helpful than softmax loss when training with same unknown samples .

Visual Analysis

2-D deep feature visualization of MNIST ($\mathcal{D}_{\mathcal{KK}}$) is already displayed in Figure 3.2(a) when LeNet++ is trained as a closed-set classifier using SM, LeNet++ works well on provide 2-D features for MNIST samples, 10 classes scatter evenly and separately in the 2-D plot. Figure 5.3 shows deep feature representations of MNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$) using different approaches. The visualization results of testing on EMNIST letters(N-Z) and FMNIST as $\mathcal{D}_{\mathcal{UU}}$ can be found in Figure A.3 and Figure A.4 respectively.

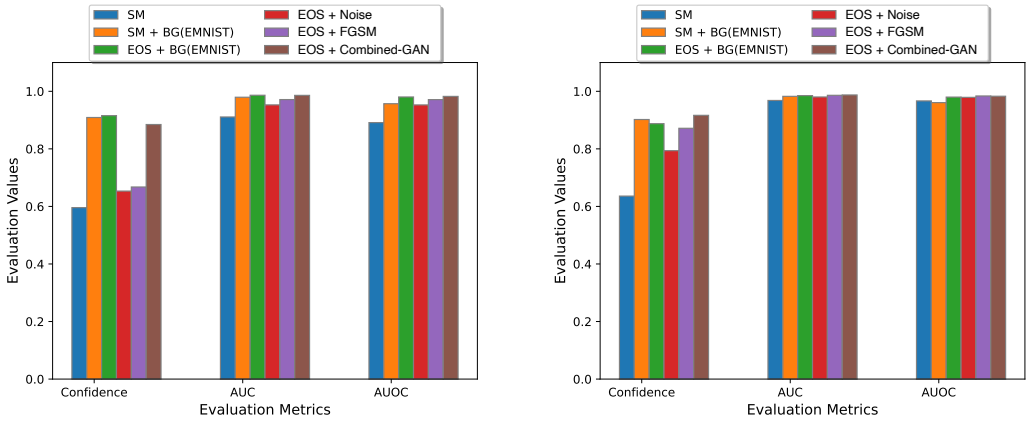
For SM approach, the unknown samples scatter without patterns on the 2-D plot, the histogram also shows the unknown samples share large overlap with known samples, as shown in Figure 5.3(a). Since SM+BG approach directly classifies the unknown samples as another class by adding one more dimension on the logit layer, we can see 11 patterns in Figure 5.3(b), but the background class still share large overlap with known classes, especially around the origin. Figure 5.3(c)-Figure 5.3(f) show approaches using EOS, the unknown samples are driven close to the origin. Using our approach, LeNet++ can separate $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$ more effectively and force $\mathcal{D}_{\mathcal{UU}}$ to have smaller magnitude.



(a) Testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and subset of EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$)

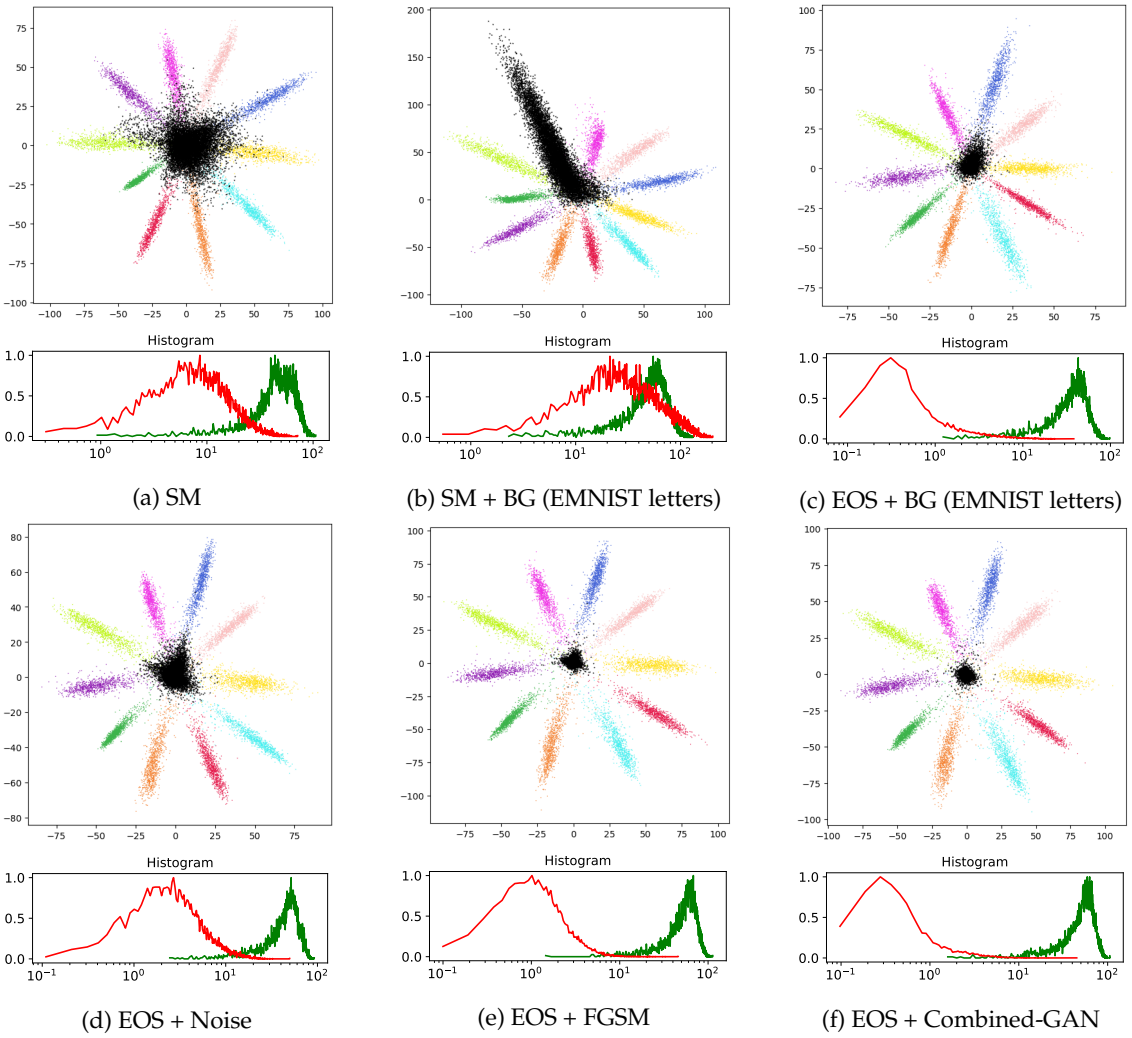
Figure 5.1: OSCR RESULTS OF TRAINING ON MNIST ($\mathcal{D}_{\mathcal{KK}}$). The plots on the top illustrate OSCR curve (Section 4.3.3) of multiple approaches testing on different datasets. The plots in the bottom shows CCR values at different FPR .



(a) Testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and subset of EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$)

Figure 5.2: EVALUATION RESULTS OF TRAINING ON MNIST ($\mathcal{D}_{\mathcal{KK}}$). The figure shows the comparision of multiple approaches testing on different datasets.



(a) SM (b) SM + BG (EMNIST letters) (c) EOS + BG (EMNIST letters)

(d) EOS + Noise (e) EOS + FGSM (f) EOS + Combined-GAN

Figure 5.3: LeNet++ responses to MNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$). LeNet++ is trained on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters (A-M) ($\mathcal{D}_{\mathcal{KU}}$), and tested on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$). 2-D feature visualization plots are displayed on the top, colored dots represent samples from MNIST, while the black dots represent samples from KMNIST. The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$.

5.2.2 Experiment 2: Training on KMNIST

Quantitative Analysis

In this experiment, we train LeNet++ to recognize KMNIST characters and reject samples from unknown samples. For the background approach, we use MNIST digits as $\mathcal{D}_{\mathcal{KU}}$. In the validation process, we also use MNIST digits as $\mathcal{D}_{\mathcal{KU}}$ to select the best models. Then, we evaluate the performance of different approaches by testing on the following two different datasets:

- KMNIST ($\mathcal{D}_{\mathcal{KK}}$) + EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)
- KMNIST ($\mathcal{D}_{\mathcal{KK}}$) + FMNIST ($\mathcal{D}_{\mathcal{UU}}$)

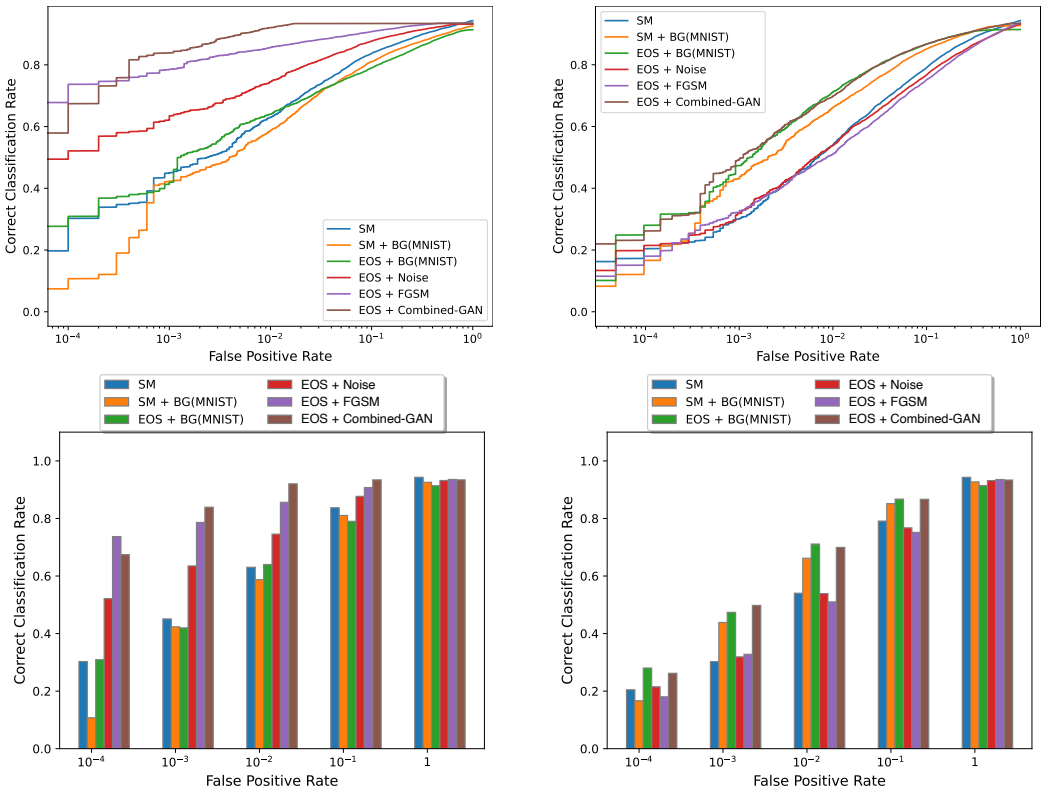
Figure 5.4 illustrates the OSCR curves and results of CCR at different FPR when testing on two different datasets using multiple approaches. From the OSCR curves, we can see for both cases, our approach gets relatively higher OSCR curves compared with other approaches. Correspondingly, the bar charts shows our approach almost achieves highest CCR values regarding the FPR of interest. Besides, our approach also achieves better performance regarding to evaluation metrics, as shown in Figure 5.5. The evaluation results indicate our generated samples can be used to capture feature in the open space, at least for classes in FMNIST and EMNIST letters.

From Figure 5.4, it is obvious to see that FMNIST is much easier to be detected for LeNet++ than EMNIST letters, resulting in relatively higher OSCR curve for most of the approaches. This is because FMNIST is far more different from KMNIST characters, while EMNIST letters share some similarities with KMNIST characters.

Visual Analysis

Figure 5.6 shows 2-D deep feature visualization of KMNIST ($\mathcal{D}_{\mathcal{KK}}$) when LeNet++ is trained as a closed-set classifier using SM. Similar with MNIST, LeNet++ works well on providing 2-D deep representation for KMNIST. 10 clear patterns can be seen in the plot, each pattern represents one digit class. Since KMNIST is more complex than MNIST, the decision boundaries are blurrier, which indicates relatively lower confidence. Figure 5.3 shows deep feature representations of KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST ($\mathcal{D}_{\mathcal{UU}}$) using different approaches. The visualization results of testing on EMNIST letters as $\mathcal{D}_{\mathcal{UU}}$ can be found in Figure A.7.

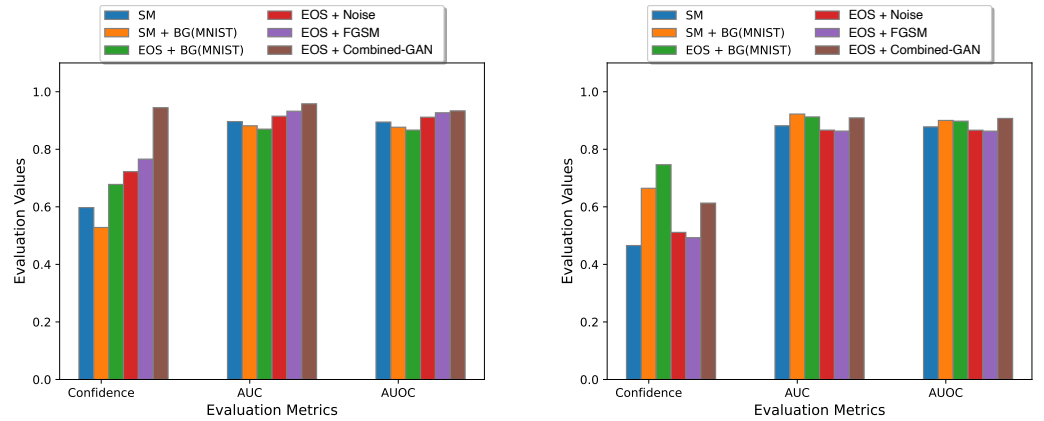
For SM approach, the unknown samples scatter without patterns on the plot, the histogram also shows the unknown samples share large overlap with known samples, as shown in Figure 5.7(a). For the approaches using the background class (MNIST), the unknown samples are fail to separated from known samples, they share large overlap with known classes (Figure 5.7(b) and Figure 5.7(c)). The reason is LeNet++ is trained on MNIST as the background class, since FMNIST is far different, when we test on FMNIST, it does not help us to recognize FMNIST effectively. In Figure 5.7(d)-Figure 5.7(f), the unknown samples are driven close to the origin. Using our approach, LeNet++ can separate $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$ more effectively and force $\mathcal{D}_{\mathcal{UU}}$ to have smaller magnitude, as shown in Figure 5.7(f).



(a) Testing on KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)

Figure 5.4: OSCR RESULTS OF TRAINING ON KMNIST ($\mathcal{D}_{\mathcal{KK}}$) . The plots on the top illustrate OSCR curve (Section 4.3.3) of multiple approaches testing on different datasets. The plots in the bottom shows CCR values at different FPR.



(a) Testing on KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)

Figure 5.5: EVALUATION RESULTS OF TRAINING ON KMNIST ($\mathcal{D}_{\mathcal{KK}}$). The figure shows the comparision of multiple approaches testing on different datasets.

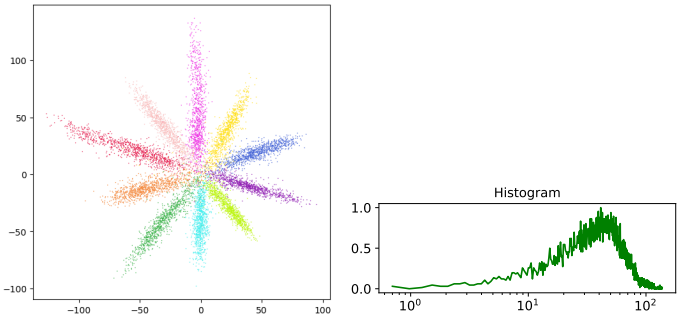
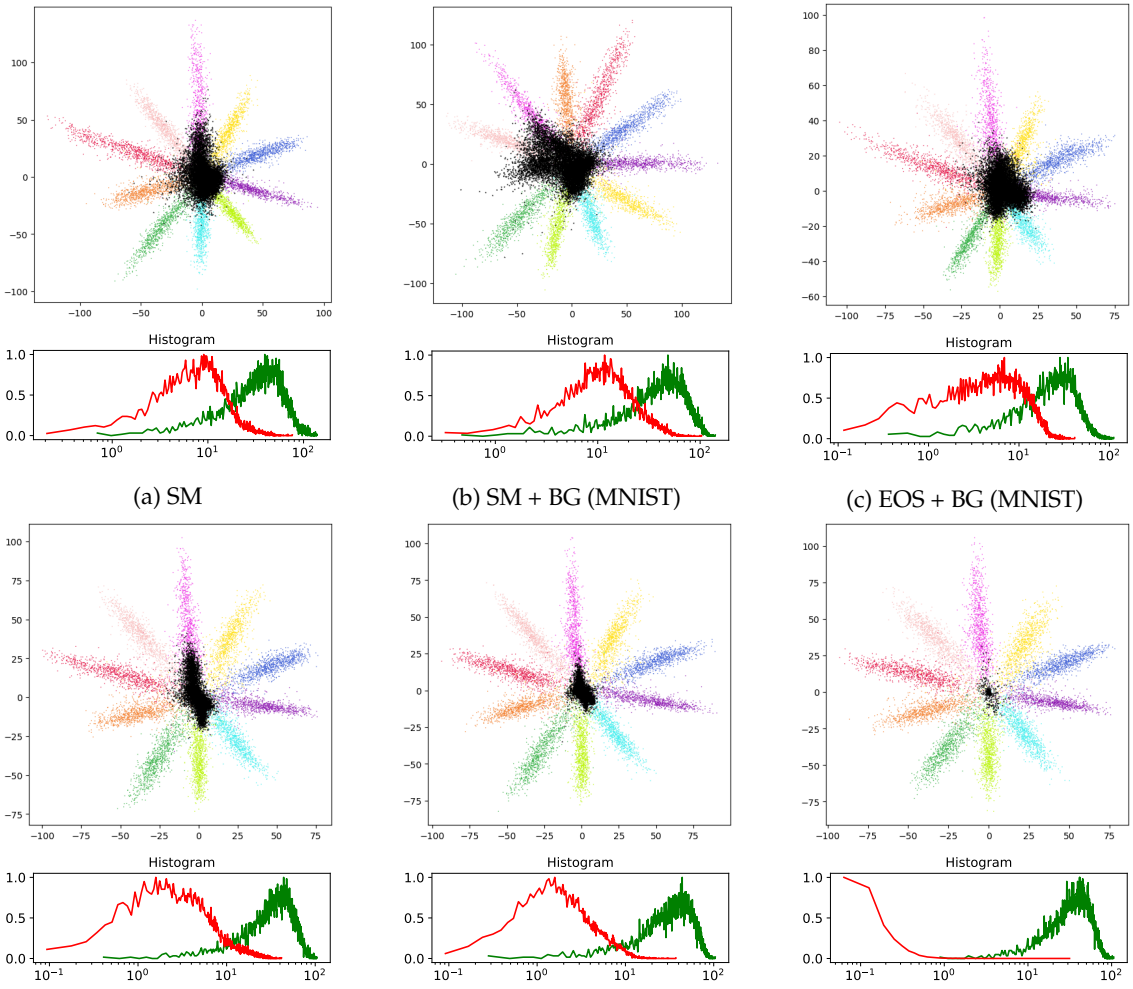


Figure 5.6: LeNet++ responses to KMNIST ($\mathcal{D}_{\mathcal{KK}}$). LeNet++ is trained and tested on KMNIST. 2-D feature visualization plots are displayed on the left, and each color represents one class in KMNIST. The figures on the right are the feature magnitude histograms of KMNIST samples.



(a) SM (b) SM + BG (MNIST) (c) EOS + BG (MNIST)

(d) EOS + Noise (e) EOS + FGSM (f) EOS + Combined-GAN

Figure 5.7: LeNet++ responses to KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST ($\mathcal{D}_{\mathcal{UU}}$). LeNet++ is trained on KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and different $\mathcal{D}_{\mathcal{KU}}$, and tested on KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST ($\mathcal{D}_{\mathcal{UU}}$). 2-D feature visualization plots are displayed on the top, colored dots represent samples from KMNIST, while the black dots represent samples from FMNIST. The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$.

5.2.3 Experiment 3: Training on FMNIST

Quantitative Analysis

In this experiment, we train LeNet++ to classify FMNIST and reject samples from unknown samples. For the background approaches, we use MNIST digits as $\mathcal{D}_{\mathcal{KU}}$. In the validation process, we also use MNIST digits as $\mathcal{D}_{\mathcal{KU}}$ to select the best models in the validation process. Then, we evaluate the performance of different approaches by testing on the two following different datasets:

- FMNIST ($\mathcal{D}_{\mathcal{KK}}$) + EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)
- FMNIST ($\mathcal{D}_{\mathcal{KK}}$) + KMNIST ($\mathcal{D}_{\mathcal{UU}}$)

Figure 5.8 shows the OSCR results when we test LeNet++ on EMNIST letters and KMNIST as $\mathcal{D}_{\mathcal{UU}}$. Our approach achieves comparable results among all approaches when testing on EMNIST letters, while it does not have good performance when testing on KMNIST.

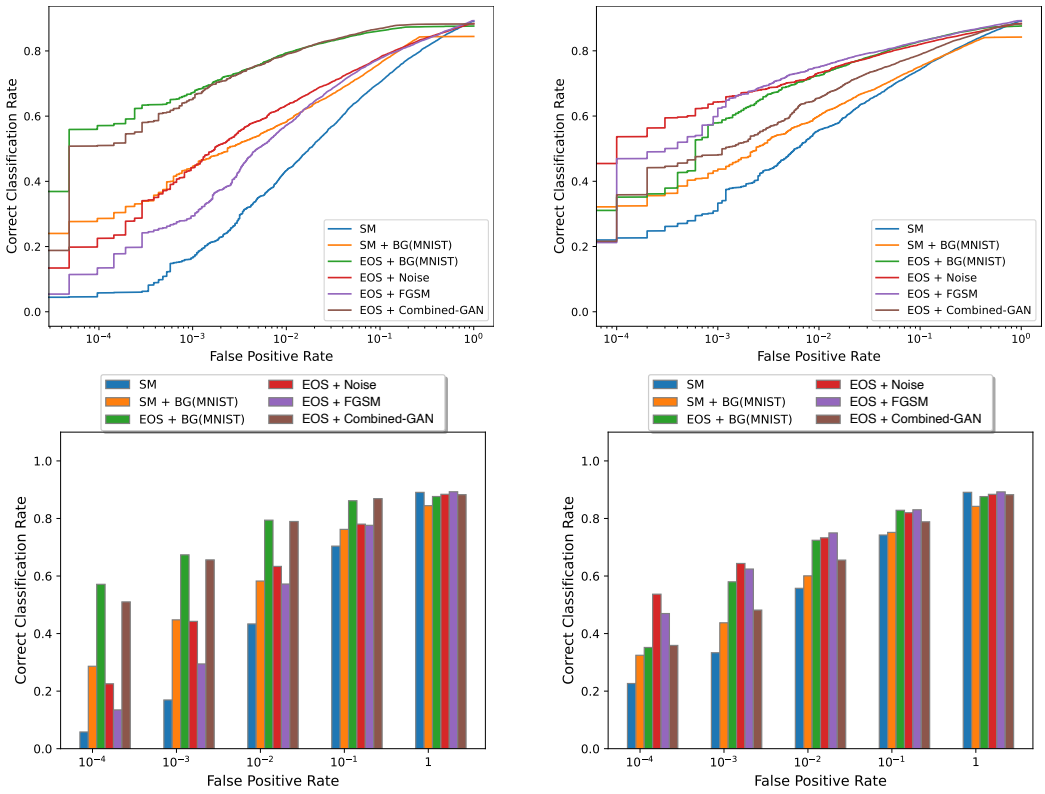
When testing on EMNIST lettes, EOS+BG approach achieves best results regarding OSCR results, as shown in Figure 5.8(a). As we discussed in Section 5.2.1, the EMNIST letters and MNIST digits share some similiarities. Thus, when LeNet++ is trained on MNIST as the background class, it will easily detect EMNIST letters during testing. Even we use generated samples from FMNIST, which is far different from EMNIST letters, our approach still can achieves better general results regarding the evaluation metrics in Figure 5.9(a). However, our approach do not achieve best CCR values at lower FPR values compared with EOS+BG (Figure 5.8(a)).

When testing on the KMNIST characters, our approach fails to compete with other approaches regarding the OSCR curve and corresponding evaluation metrics. The results indicates our generative model are sensitive for the dataset, when the $\mathcal{D}_{\mathcal{KK}}$ is far from $\mathcal{D}_{\mathcal{UU}}$, the generated samples from known samples may have limited representation ability, in our case, the generative samples from FMNIST are hard to capture the related features from unknow samples from KMNIST.

Visual Analysis

Figure 5.10 shows 2-D deep feature visualization of FMNIST ($\mathcal{D}_{\mathcal{KK}}$) when LeNet++ is trained as a closed-set classifier using SM. Different from MNIST and KMNIST, it is more difficult for LeNet++ to provide 2D deep representation for FMNIST. Although we can see 10 patterns in the plot, the patterns are not clearly and evenly separated like MNIST, some classes are even mixed with each other, which makes LeNet++ hards to recognize. From Figure 5.8), The classification accuracy on the FMNIST is 89% (MNIST: 99%, KMNIST: 95%).

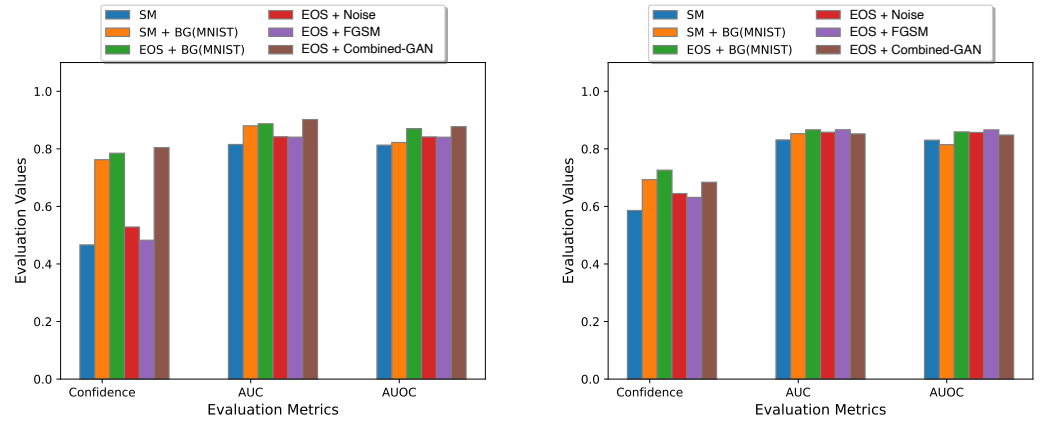
Figure 5.11 shows deep feature representations using LeNet++ when testing on FMNIST as $\mathcal{D}_{\mathcal{KK}}$ and EMNIST letters as $\mathcal{D}_{\mathcal{UU}}$. The visualization results of testing on KMNIST as $\mathcal{D}_{\mathcal{UU}}$ can be found in Figure A.8. Since SM+BG approach directly classifies the unknown samples as another class by adding one more dimension on the logit layer, we can see 11 patterns in Figure 5.7(b), but the unknown samples still share large overlap with the known samples around the origin. Figure 5.3(c)-Figure 5.3(f) show approaches using EOS, the unknown samples are driven close to the origin. Using our approach, LeNet++ can separate $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$ more effectively and force $\mathcal{D}_{\mathcal{UU}}$ to have smaller magnitude, as shown in Figure 5.3(f).



(a) Testing on FMNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on FMNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$)

Figure 5.8: OSCR RESULTS OF TRAINING ON FMNIST ($\mathcal{D}_{\mathcal{KK}}$) . The plots on the top illustrate OSCR curve (Section 4.3.3) of multiple approaches testing on different datasets. The plots in the bottom shows CCR values at different FPR.



(a) Testing on FMNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on FMNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$)

Figure 5.9: EVALUATION RESULTS OF TRAINING ON FMNIST ($\mathcal{D}_{\mathcal{KK}}$). The figure shows the comparision of multiple approaches testing on different datasets.

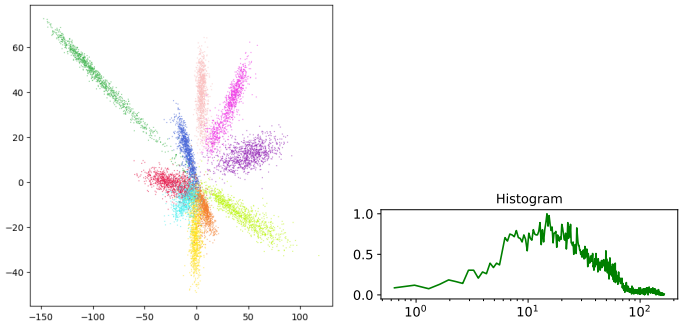
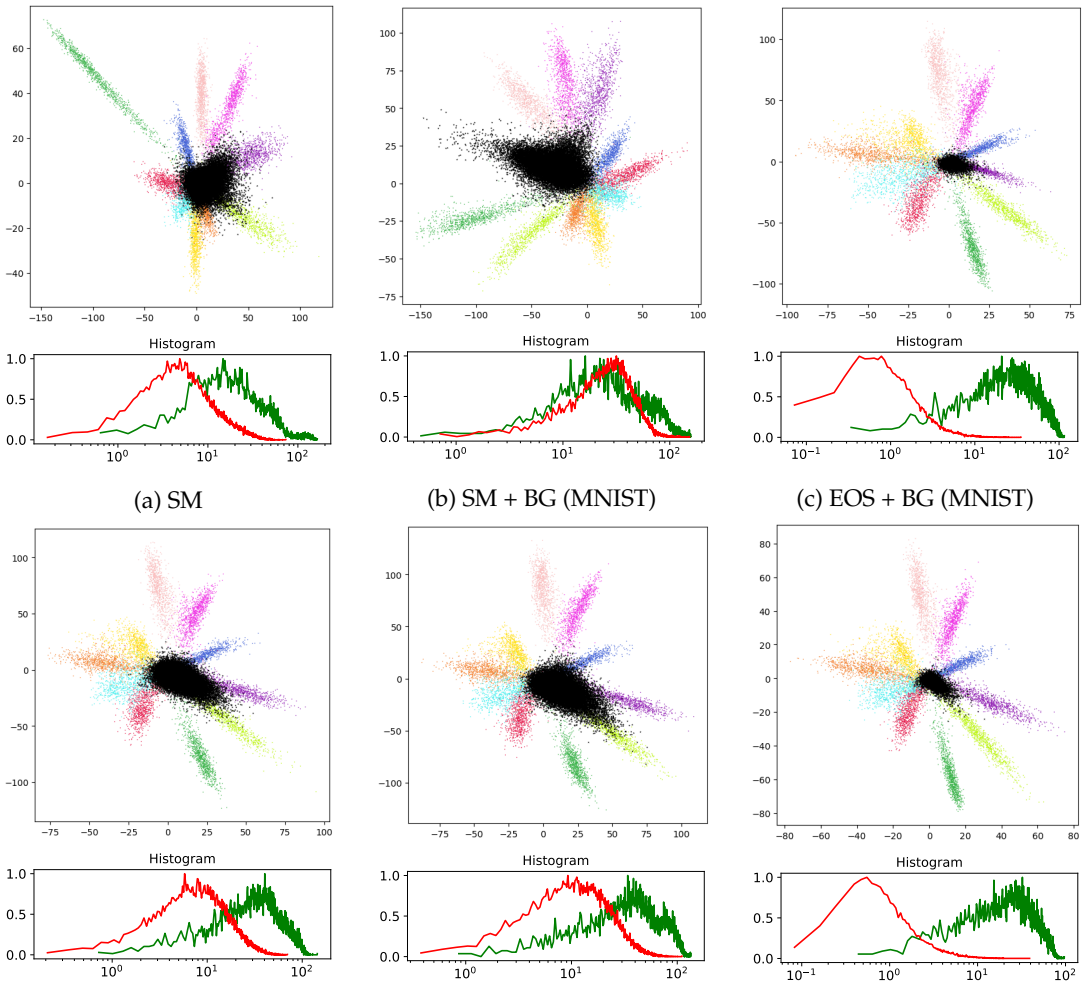


Figure 5.10: LENET++ RESPONSES TO FMNIST ($\mathcal{D}_{\mathcal{KK}}$). LeNet++ is trained and tested on KMNIST. 2-D feature visualization plots are displayed on the left, and each color represents one class in KMNIST. The figures on the right are the feature magnitude histograms of FMNIST samples.



(a) SM (b) SM + BG (MNIST) (c) EOS + BG (MNIST)

(d) EOS + Noise (e) EOS + FGSM (f) EOS + Combined-GAN

Figure 5.11: LENET++ RESPONSES TO FMNIST ($\mathcal{D}_{\mathcal{KK}}$) AND EMNIST LETTERS ($\mathcal{D}_{\mathcal{UU}}$). LeNet++ is trained on FMNIST ($\mathcal{D}_{\mathcal{KK}}$) and MNIST ($\mathcal{D}_{\mathcal{KU}}$), and tested on FMNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters($\mathcal{D}_{\mathcal{UU}}$). 2-D feature visualization plots are displayed on the top. Colored dots represent samples from FMNIST, while the black dots represent samples from EMNIST letters. The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$.

5.3 ResNet-18++ for OSR

Several experiments are done to train ResNet-18++ on Set 5 and Set 6 for OSR. Some research questions need to be validated from these experiments:

- Compared with other apporaches, can our Combined-GAN model generate effective open-set samples for different OSR secenarios (testing on different unknown samples)?
- Can ResNet-18++ be used to provide good 2-D deep feature representations for different datasets and how does ResNet-18++ response to known and unknown samples for different approaches?

5.3.1 Experiment 4: Training on CIFAR-10

Quantitative Analysis

In this experiment, ResNet-18++ is trained to classify CIFAR-10 images and reject samples from unknown classes. For the background approach using SM and EOS, we randomly select 45 subclasses from CIFAR-100 dataset as $\mathcal{D}_{\mathcal{KU}}$, and they are used as $\mathcal{D}_{\mathcal{KU}}$ to select the best models in the validation process. We select other 45 different subclasses (different from $\mathcal{KU}$) as $\mathcal{D}_{\mathcal{UU}}$ for testing. In addition, we also test on SVHN dataset as $\mathcal{D}_{\mathcal{UU}}$. We evaluate the performance of different approaches by testing on the following two different datasets:

- CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) + a subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$)
- CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) + SVHN ($\mathcal{D}_{\mathcal{UU}}$)

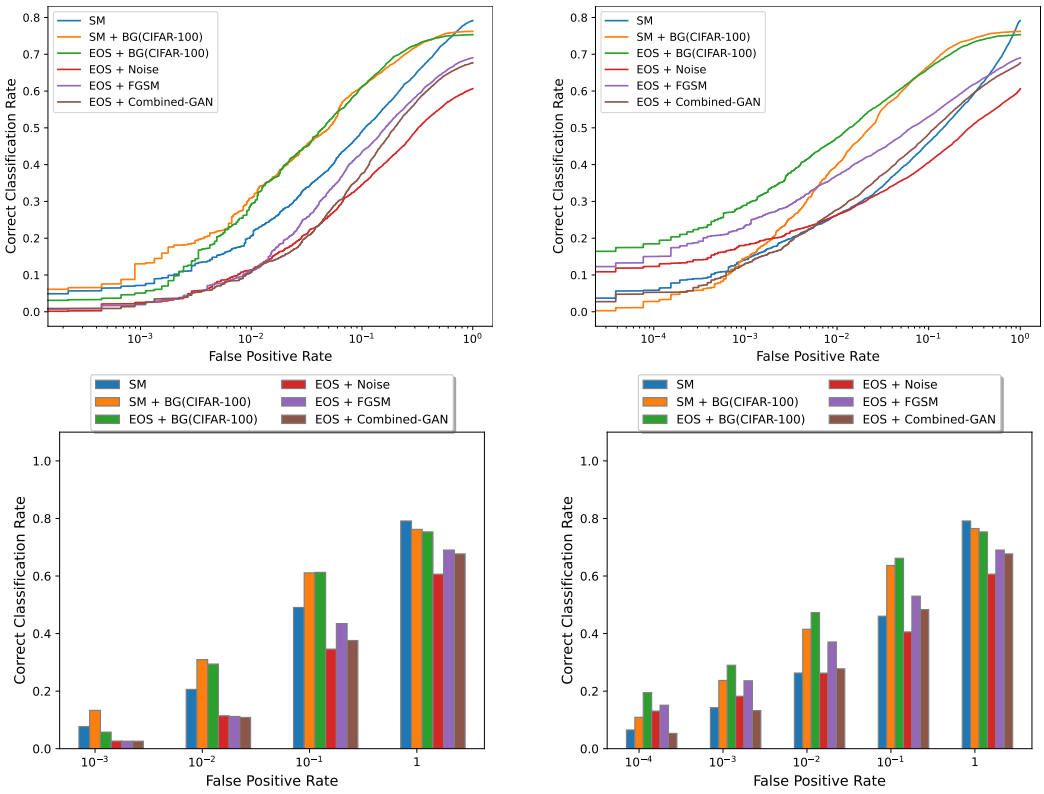
Figure 5.12 illustrates the OSCR curves results when we test ResNet-18++ on a subset of CIFAR-100 and SVHN as $\mathcal{D}_{\mathcal{UU}}$. As a supplement, we also provide other evaluation results in Figure 5.13 regarding the evaluation metrics like confidence, AUC and AUOC.

From the evaluation results, we can see the our approach fails to outperform all other approaches when detecting the CIFAR-100 samples as $\mathcal{D}_{\mathcal{UU}}$, especially compared with the approaches using the background class. Our geneated samples somehow mislead the open-set model in classifying known samples, as we can see in Figure 5.12, the closed-set accuracy (when FPR equals 1) is lower than other approaches. The possible reason is the generated images are not good enough to represent unknown classes from open space, at least for CIFAR-100 and SVHN.

Actually for most approaches, the performance degrades comparing with Experiment 1-3, as the FPR value becomes smaller, the CCR values drops quickly. Especially, when we testing on a subset of CIFAR-100 as $\mathcal{D}_{\mathcal{UU}}$, most CCR scores even stay below 10% when FPR equals 10^{-3}. There are possible two reasons for the degradation for CIFAR-10. One reason is that, the CIFAR-10 dataset is much more difficult to recognize than grayscale images. Even using SM, the closed-set accuracy (when FPR equals 1) is 79%, much lower than grayscale images (MNIST: 99%, KMNIST: 95%, FMNIST: 89%); Another reason is CIFAR-100 is much similar with CIFAR-10, this further makes the open-set classifier hard to recongize the CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) from CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$).

Among all approaches, the background approaches achieve relatively better results (Figure 5.12). During training, the open-set model learns how to reject the subset of CIFAR-100 ($\mathcal{D}_{\mathcal{KU}}$), when we testing on another subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$), which shares similarity with the subset in training, the unknown samples will be detected easily.

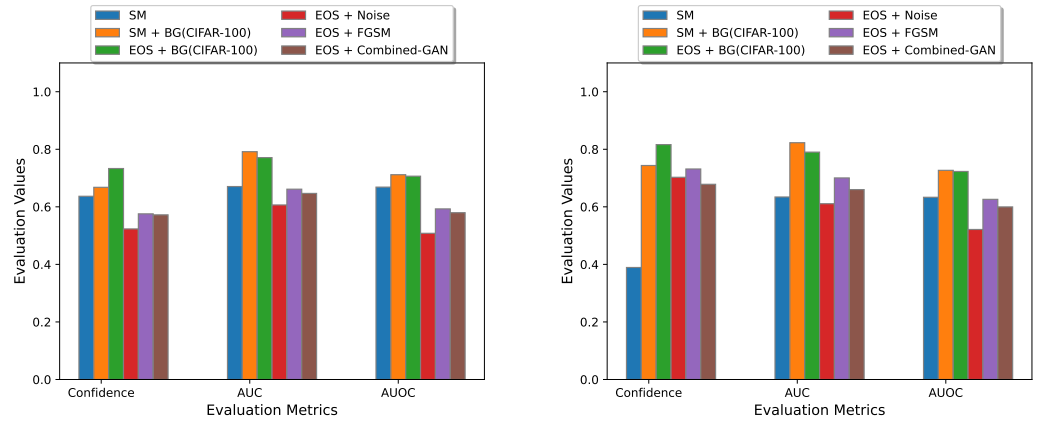
We also observe that when we test on SVHN ($\mathcal{KU}$ and $\mathcal{UU}$), the CCR values at smaller FPR values (10^{-3}) are larger than the ones of testing on a subset of CIFAR-100. This is because that SVHN is much different from CIFAR-10, thus ResNet-18++ can more likely to recognize the CIFAR-10 classes.



(a) Testing on CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) and a subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) and SVHN ($\mathcal{D}_{\mathcal{UU}}$)

Figure 5.12: OSCR RESULTS OF TRAINING ON CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) . The plots on the top illustrate OSCR curve (Section 4.3.3) of multiple approaches testing on different datasets. The plots in the bottom shows CCR values at different FPR.



(a) Testing on CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) and a subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) and SVHN ($\mathcal{D}_{\mathcal{UU}}$)

Figure 5.13: EVALUATION RESULTS OF TRAINING ON CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$). The figure shows the comparision of multiple approaches testing on different datasets.

Visual Analysis

Figure 5.14 shows 2-D deep feature visualization of CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) when ResNet-18++ is trained as a closed-set classifier using SM. As we can see in the 2-D plot, The 2-D adaptive ResNet-18++ works well on providing 2-D deep feature representations for CIFAR-10 dataset. The feature representations of 10 classes can be recognized clearly. For each class, the samples distributed almost evenly in the range of feature magnitude, as can be seen in the histogram. Since CIFAR-10 is much more complex and noisier images than grayscale images like MNIST and KMNIST, some samples are mixed in the area which is close to the origin, which results in relatively lower accuracy.

Figure 5.15 shows deep feature representations using ResNet-18++ network when testing on CIFAR-10 as $\mathcal{D}_{\mathcal{KK}}$ and CIFAR-100 dataset as $\mathcal{D}_{\mathcal{UU}}$. Actually from the 2-D plot and the histogram, we can see the unknown samples are almost overlapped with the known samples, which makes open-set model hard to detect the unknown from the known samples. The main reason is CIFAR-10 dataset and CIFAR-100 dataset share large similarity. even CIFAR-10 and CIFAR-100 almost have totally different classes, but most of the classes are animals and vehicles, and also they both have similar background, like blue sky, green grassland and etc. Since SM+BG approach directly classifies the unknown samples as another class by adding one more dimension on the logit layer, we can see 11 patterns in Figure 5.15(b), but the unknown samples still share large overlap with the known samples around the origin.

The generated samples from Combined-GAN also degrade the classification of known classes in the 2-D view, as we can see in Figure 5.15(b), the shape of patterns in 2-D plot is much wider than the approaches using softmax thresholding. Meanwhile, the feature magnitude of the known classes shrunken, which makes the feature magnitude distribution of known classes almost overlap with unknown classes. All makes the ResNet-18++ hard to classify the known classes and detect the unknown classes.

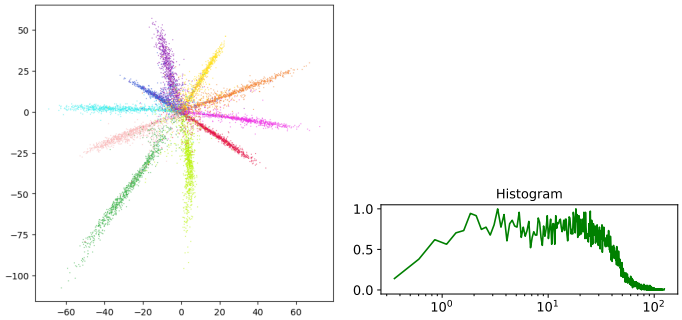
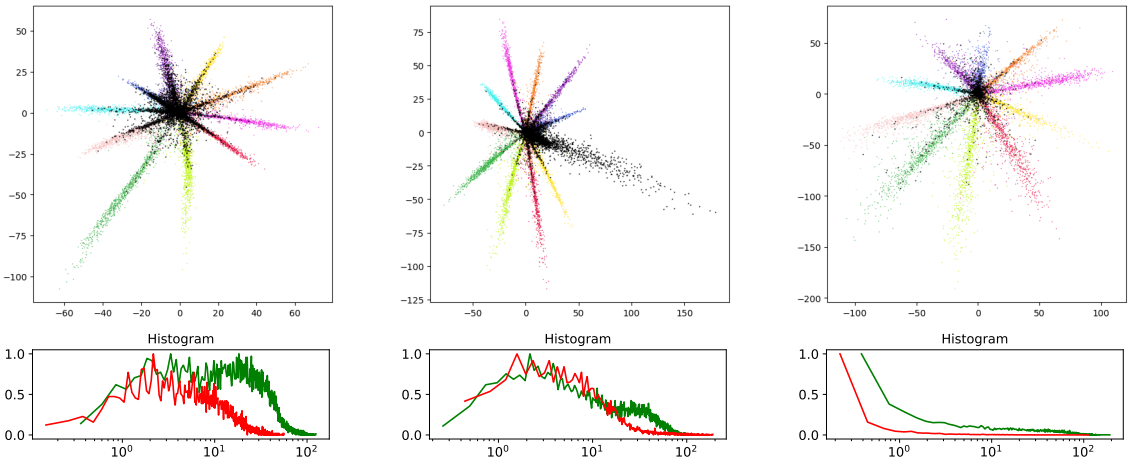
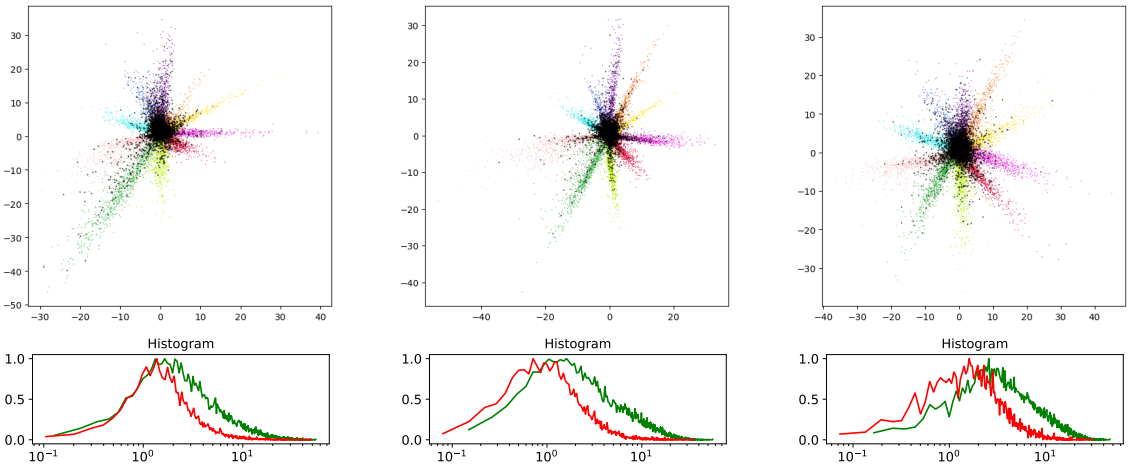


Figure 5.14: LeNet++ responses to CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$). ResNet-18++ is trained and tested on CIFAR-10. 2-D feature visualization plots are displayed on the left, and each color represents one class in CIFAR-10. The figures on the right are the feature magnitude histograms of CIFAR-10 samples.



(a) SM (b) SM + BG (CIFAR-100) (c) EOS + BG (CIFAR-100)



(d) EOS + Noise (e) EOS + FGSM (f) EOS + Combined-GAN

Figure 5.15: ResNet-18++ responses to CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) and a subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$). ResNet-18++ is trained on CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) and a subset of CIFAR-100 ($\mathcal{D}_{\mathcal{KU}}$), and tested on CIFAR-10 ($\mathcal{D}_{\mathcal{KK}}$) and another subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$). 2-D feature visualization plots are displayed on the top, colored dots represent samples from CIFAR-10, while the black dots represent samples from a subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$). The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$.

5.3.2 Experiment 5: Training on SVHN

Quantitative Analysis

In this experiment, we train ResNet-18++ to classify SVHN images and reject samples from unknown classes. For the background approach, similar with Experiment 4, we randomely select 45 subclasses from CIFAR-100 dataset as $\mathcal{D}_{\mathcal{KU}}$ for training and validation, and other 45 different subclasses (different from $\mathcal{KU}$) as $\mathcal{D}_{\mathcal{UU}}$ for testing. Hence, we evaluate the performance of different approaches by testing on: SVHN ($\mathcal{D}_{\mathcal{KK}}$) + a subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$)

From the evaluation results (Figure 5.16), similar with the results in Experiment 4, our approach does not achieve the best result in detecting CIFAR-100 samples ($\mathcal{D}_{\mathcal{UU}}$), especially compared with the approaches using the background class. Our geneated samples somehow mislead the open-set model in classifying known samples, as we can see in Figure 5.16(b), the closed-set accuracy (when FPR equals 1) is lower than other approaches. The possible reason is that SVHN dataset is very different from CIFAR-100 dataset, the generated images can not convey enough infomation about CIFAR-100 dataset.

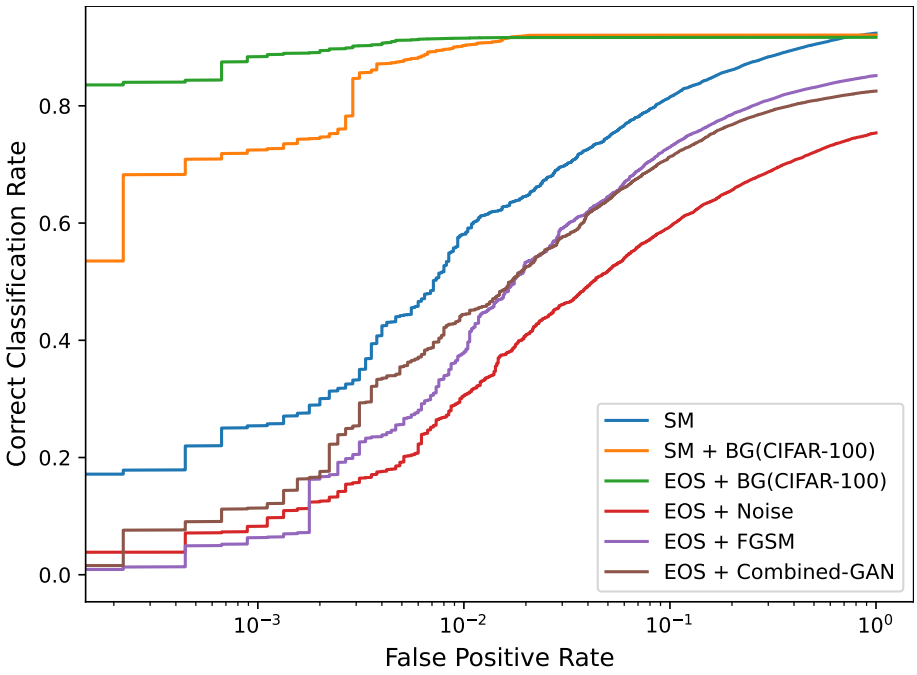
Among all the approaches, the background approaches achieve much better results, especially for EOS+BG. Even when FPR is very small (10^{-3}), the corresponding CCR is still above 80%. There are two main reasons. One is that, during training, the ResNet-18++ learns how to reject the subset of CIFAR-100 ($\mathcal{D}_{\mathcal{KU}}$), when we testing on another subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$), which shares similarity with the subset in training ($\mathcal{KU}$ and $\mathcal{KU}$ may from same super classes), the open-set model can separate the SVHN from CIFAR-100. Another reason is that, SVHN is easier to recognize, as we can see in Figure 5.16(b), the closed-set accuracy (when FPR equals 1) by using SM is 92%, better than the accuracy on CIFAR-10 (79%).

Visual Analysis

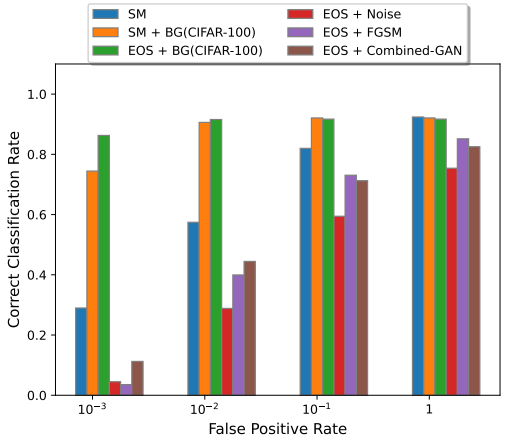
Figure 5.17 shows 2-D deep feature visualization of SVHN ($\mathcal{D}_{\mathcal{KK}}$) when ResNet-18++ is trained as a closed-set classifier using SM. Similar with the case of CIFAR-10, ResNet-18++ works well on classifying SVHN while providing good 2-D deep feature representations, 10 patterns can be seen clearly from the plot.

Figure 5.18 shows deep feature representations using ResNet-18++ network when testing on SVHN as $\mathcal{D}_{\mathcal{KK}}$ and the subset of CIFAR-100 as $\mathcal{D}_{\mathcal{UU}}$. Even they are totally different datasets, the samples still share a large overlap. But, this is better than the case in which ResNet-18++ responses to CIFAR-10 and CIFAR-100 samples (Figure 5.15(a)). Since CIFAR-100 dataset is much different from SVHN dataset. When ResNet-18++ is trained on the background class (CIFAR-100), ResNet-18++ can separate the unknown from known classes successfully using SM or EOS which can be seen in Figure 5.18(b) and Figure 5.18(c).

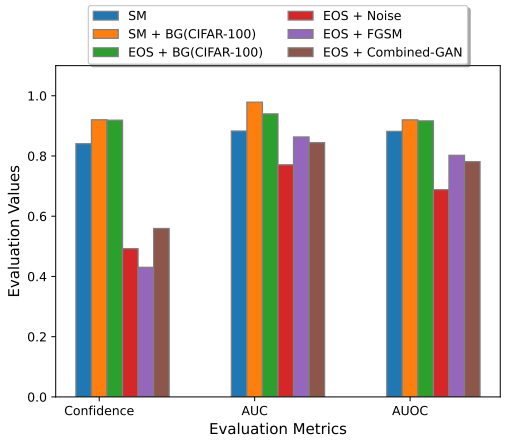
For approaches using EOS with generated samples from noise, FGSM and our Combined-GAN, the visual results are similar with the ones in the case of CIFAR-10. The feature magnitude of the known classes shrunken, which makes the feature magnitude distribution of known classes almost overlap with unknown classes. This makes ResNet++ hard to classify the known classes and detect the unknown classes.



(a) OSCR curve



(b) CCR values at different FPR values



(c) Evaluation results

Figure 5.16: Evaluation Results of testing on SVHN ($\mathcal{D}_{\mathcal{KK}}$) + a subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$).

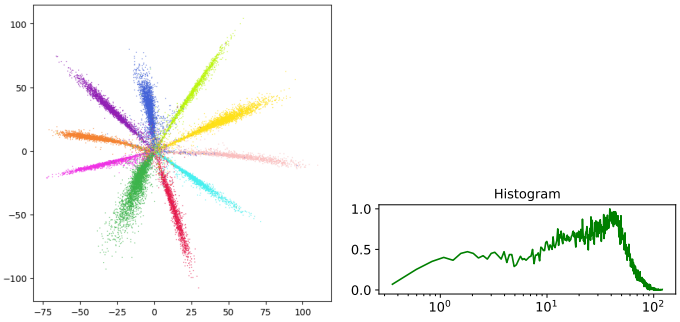
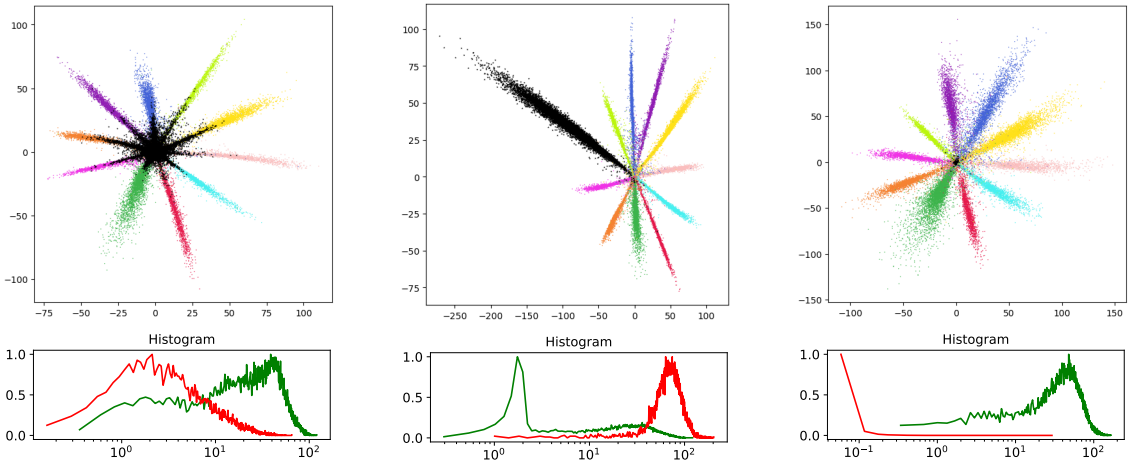


Figure 5.17: RESNET-18++ RESPONSES TO SVHN ($\mathcal{D}_{\mathcal{KK}}$). LeNet++ is trained and tested on SVHN. 2-D feature visualization plots are displayed on the left, and each color represents one class in SVHN. The figures on the right are the feature magnitude histograms of SVHN samples.



(a) SM (b) SM + BG (CIFAR-100) (c) EOS + BG (CIFAR-100)

(d) EOS + noise (e) EOS + FGSM (f) EOS + Combined-GAN

Figure 5.18: RESNET-18++ RESPONSES TO SVHN ($\mathcal{D}_{\mathcal{KK}}$) AND A SUBSET OF CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$). LeNet++ is trained on SVHN ($\mathcal{D}_{\mathcal{KK}}$) and a subset of CIFAR-100 ($\mathcal{D}_{\mathcal{KU}}$), and tested on SVHN ($\mathcal{D}_{\mathcal{KK}}$) and another subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$). 2-D feature visualization plots are displayed on the top, colored dots represent samples from SVHN, while the black dots represent samples from the subset of CIFAR-100 ($\mathcal{D}_{\mathcal{UU}}$). The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$.

Testing Dataset	Reconstruction Loss	Adversarial Loss	Mixed-class Loss	Accuracy	Confidence	AUC	AUOC
MNIST+EMNIST	✓	✓	✓	**0.9835**	0.7978	**0.9818**	**0.9799**
	✗	✓	✓	0.9817	0.6536	0.9356	0.9355
	✓	✗	✓	0.9816	**0.8114**	0.9808	0.9787
	✓	✓	✗	0.9823	0.7358	0.9762	0.9752
MNIST+KMNIST	✓	✓	✓	**0.9833**	0.9261	**0.9883**	**0.9831**
	✗	✓	✓	0.9812	0.7513	0.9757	0.9744
	✓	✗	✓	0.9816	0.9072	0.9855	0.9812
	✓	✓	✗	0.9824	**0.9323**	0.9871	0.9822
MNIST+FMNIST	✓	✓	✓	**0.9829**	**0.9793**	**0.9915**	**0.9829**
	✗	✓	✓	0.9813	0.7536	0.9768	0.9756
	✓	✗	✓	0.9819	0.978	0.9893	0.9818
	✓	✓	✗	0.9821	0.976	0.9889	0.9821

Table 5.1: EFFECT OF DIFFERENT LOSS FUNCTIONS FOR LENET++. LeNet++ is trained on the MNIST and generated samples from MNIST using generative models with different loss functions. Then, LeNet++ is tested on $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$.

5.4 Ablation Study

In this section, we evaluate the effect of three kinds of loss functions in the generative model: reconstruction loss, adversarial loss and mixed-class loss.

5.4.1 Quantitative Analysis

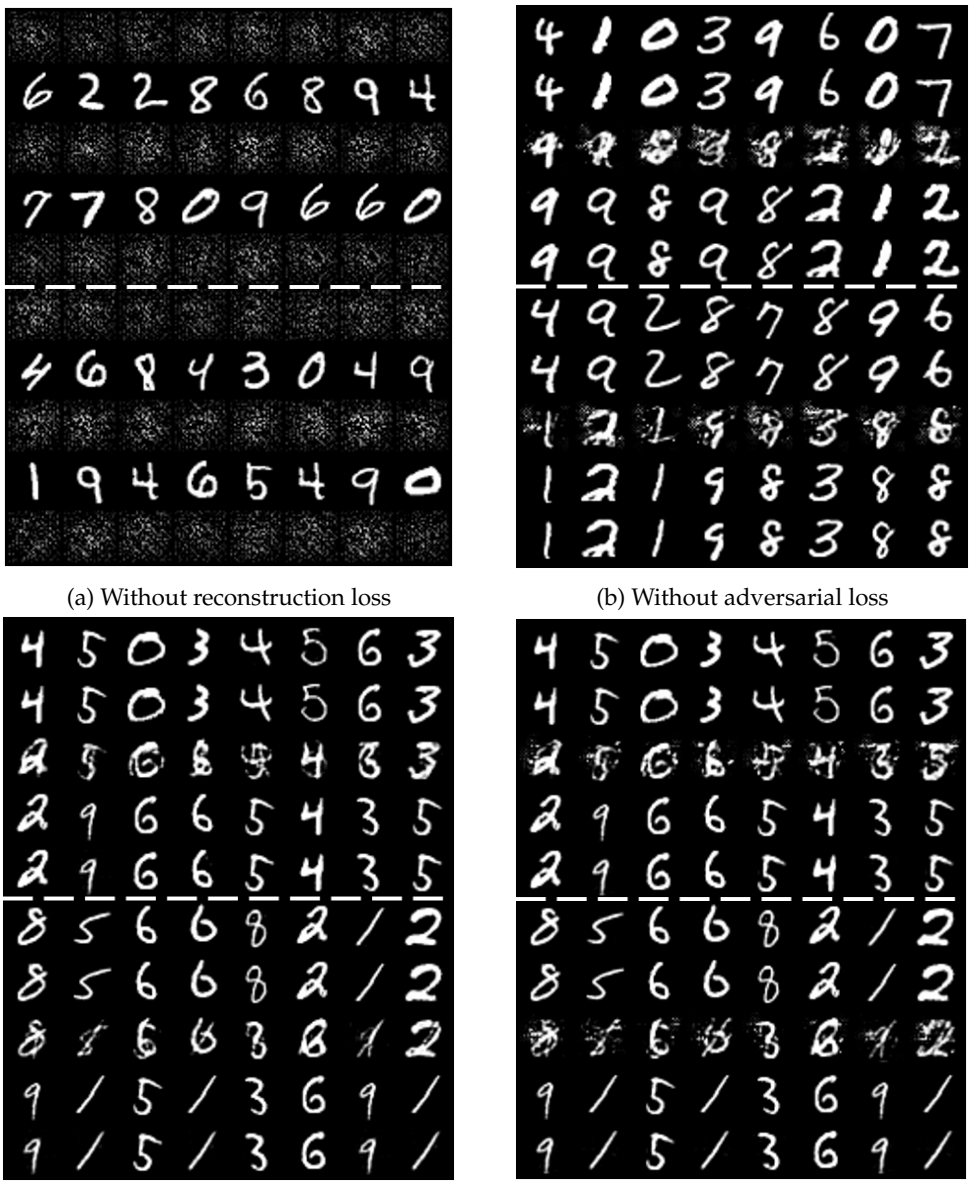
Since our goal is to generate effective open-set samples for an open-set model, we evaluate the performance of LeNet++ using the metrics mentioned in Section 4.3 to assess the impact of different loss settings on the open-set model. Here, we use LeNet++ as the open-set model to do the quantitative analysis.

As we can see in Table 5.1. In most of the cases, our original loss for generative model achieves the best results, especially for AUC and AUOC. Without reconstruction loss, the corresponding performance of LeNet++ falls with a large margin. The effect of adversarial loss and mixed-class loss is limited but they still contribute to better results.

5.4.2 Visual Analysis

Figure 5.19 illustrates the results of different combinations of three loss functions. Without reconstruction loss, the network only can output noisy images (Figure 5.19(a)), it seems that the combination of adversarial loss and mixed-class loss does not work. The result is consistence with the quantitative analysis in Section 5.4.1, While Figure 5.19(b) shows the network can also produce plausible images without adversarial loss. Similarly in Figure 5.19(c), the network can generate clear reconstructed images. The generated images seem too clear. In this case, the clear images may be easily misclassified as one of the known images. Figure 5.19(d) shows the result of the original setting of our generative model, which shows effective open-set samples that are combined both features from different classes and hard to distinguish which class they belong to.

In conclusion, all the loss functions in the generative model are essential. Especially, the reconstruction loss is critical in generating effective open-set samples.



(a) Without reconstruction loss

(b) Without adversarial loss

(c) Without mixed-class loss

(d) Our original loss setting

Figure 5.19: DIFFERENT LOSS FUNCTIONS FOR OUR COMBINED-GAN. Each subfigure displays two sets of images (separated by a white dash line), which include the original images from one class (1st row), original images from another class (3rd row) and generated images from two different classes (2nd row).

Chapter 6

Discussion and Limitations

6.1 Discussion

6.1.1 Visualization of the Generated Samples

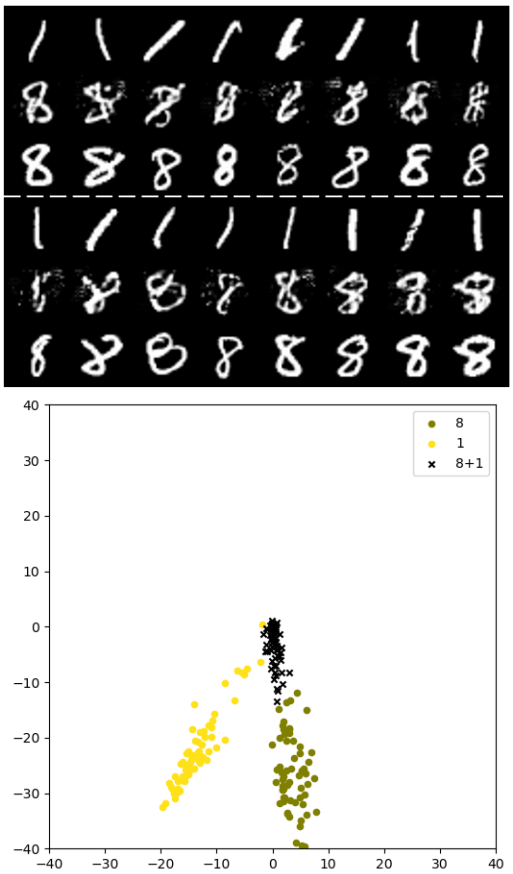
It is interesting to visualize the unknown samples generated by our Combined-GAN to get an insight about the unknown samples trained in OSR. We use the small version of the generative model (Table 4.2a) to generate open-set samples from MNIST, KMNIST and FMNIST. We also generate unknown samples from CIAFR-10 and SVHN dataset using a bigger generative model (Table 4.2b). In addition, 2-D deep feature visualization is provide to show how LeNet++ and ResNet-18++ reponse to the known and unknown samples. Here, we give examples from MNIST dataset and CIFAR-10 dataset, more generated samples from KMNIST, FMNIST and SVHN dataset can be seen in Figure A.1.

Generated samples from MNIST

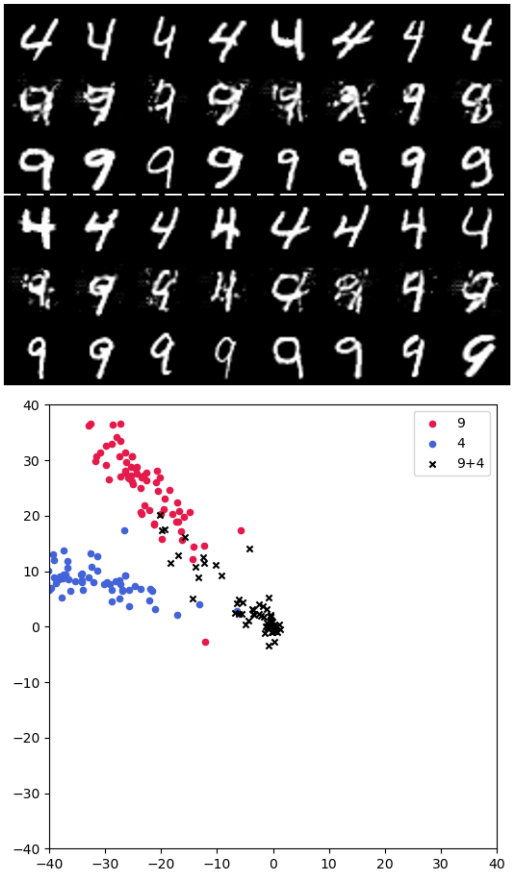
In Experiment 1 (Section 5.2.1), we save the best Combined-GAN model and LeNet++ model after validation process. Then, we use the trained generative model to generate samples from MNIST dataset in testing phase. In Figure 6.1, we provide some generated samples from two different classes of MNIST digits and the deep feature visualization of the samples in three scenarios. From the figures on the top, we can see the generated samples contained mixed features from two different classes, which is hard for us to recognise and distinguish. From the scatter plot in the bottom, we can see that samples from original digit classes show clear patterns as each class clusters in a group. While the generated samples are pushed close to the origin. Thus, LeNet++ successfully separates the unknown from known classes.

Generated samples from CIFAR-10

We also use the trained generative model in Experiment 4 (Section 5.3.1) to generate samples from CIFAR-10 dataset for testing. Figure 6.2 displays some generated samples from two different classes of the CIFAR-10 dataset and deep feature visualization of testing samples in three scenarios. From the figures on the top, we can see that some features of the generated samples come from original samples. However, the fake samples are hard to be understood from human perception. The scatter plots in the bottom show that both known and known samples are very close to the origin. It is also hard for ResNet-18++ to separate the known images and the synthetic images when the network is trained on CIFAR-10 dataset and generated samples by Combined-GAN.

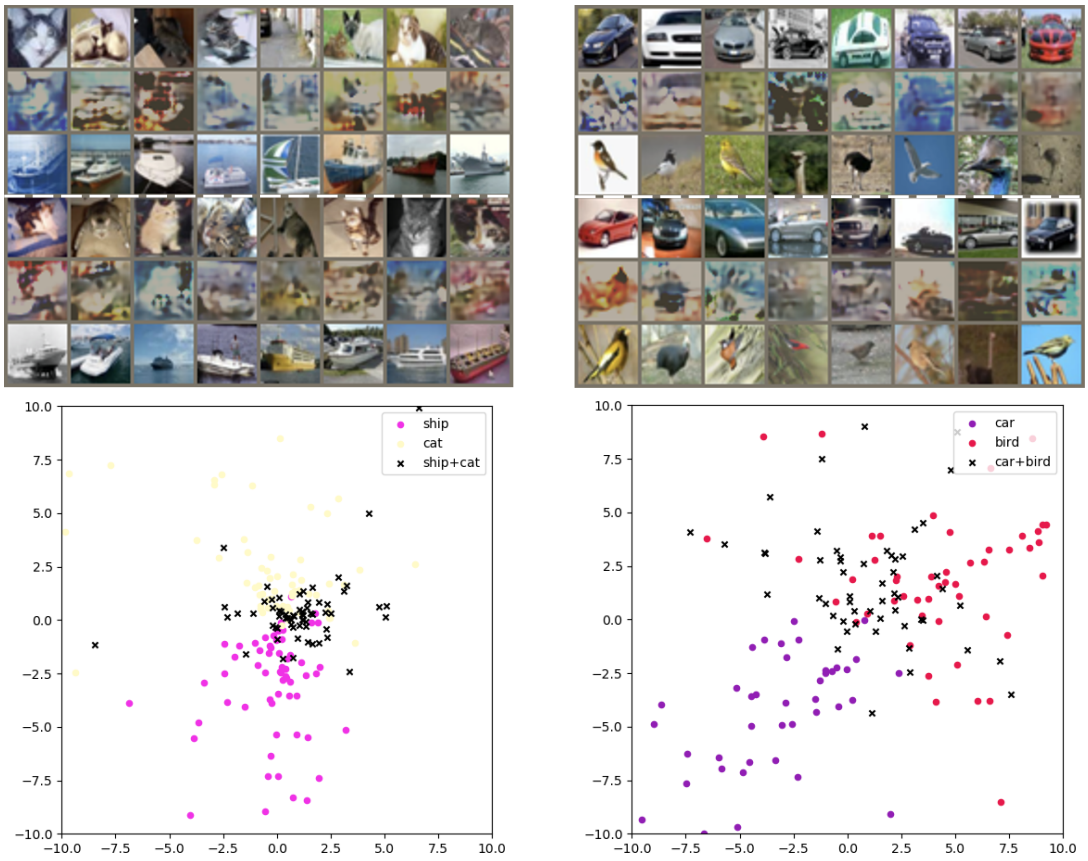


(a) Samples from digit "1" and digit "8".



(b) Samples from digit "4" and digit "9".

Figure 6.1: GENERATED SAMPLES AND FEATURE VISUALIZATION FOR MNIST. The figure illustrates two different scenarios. The figures on top display two sets of images (separated by a white dash line), which include the original images from one class (1st row), original images from another class (3rd row) and generated images from two different classes (2nd row). The scatter plots in the bottom displays the feature representations of original digit images and generated images. Colored dots represent known samples and different color represents different classes. The black dots represent generated unknown samples.



(a) Samples from "ship" and "cat"

(b) Samples from "car" and "bird"

Figure 6.2: GENERATED SAMPLES AND FEATURE VISUALIZATION FOR CIFAR-10. The figure illustrates two different scenarios. The figures on top display two sets of images (separated by a white dash line), which include the original images from one class (1st row), original images from another class (3rd row) and generated images from two different classes (2nd row). The scatter plots in the bottom displays the feature representations of original digit images and generated images. Colored dots represent known samples and different color represents different classes. The black dots represent generated unknown samples.

6.1.2 2-D Visualization

In our experiments, we found the bias neuron in the last fully connected layer of the open-set model is important for 2-D visualization. When we set the bias term as True, which means the bias neuron is active, the 2-D deep feature representations of unknown samples can sometimes shown in Figure 6.3(a), as we can see, there is one class clustering around the origin. However, when we use the model for OSR, the unknown samples also cluster around the 2-D plot, in this case, the known samples hide behind the unknown samples, as shown in Figure 6.3(b). When the bias term is set as False, all known classes cluster almost evenly and symmetrically around the origin, as can be seen in Figure 6.3(c). The unknown samples distribute around the origin. We also compared the OSCR results for both cases (Figure 6.4), The OSCR curve of the case without bias neuron is higher than the case with bias neuron, the result is consistent with the visualization analysis. The bias term seems does not affect the closed set accuracy ,as both cases achieve around 99% (when FPR equals 1). In our experiment, we also found the bias neuron in the last fully connected layer works the same way as for ResNet-18++.

it's worth noting that the impact of bias on OSR still needs further investigation, since the results discussed above are only about MNIST, and the difference is relatively small. We found LeNet++ with bias neuron can sometimes also provide similar visualization as Figure 6.3(c). Even if the LeNet++ does not provide good 2-D visualization, it does not mean it can not be used for OSR. The 2-D visualization of deep s is only provided to support OSR in a way which is easy to be understood for humans.

6.1.3 Loss Functions for the Generative Model

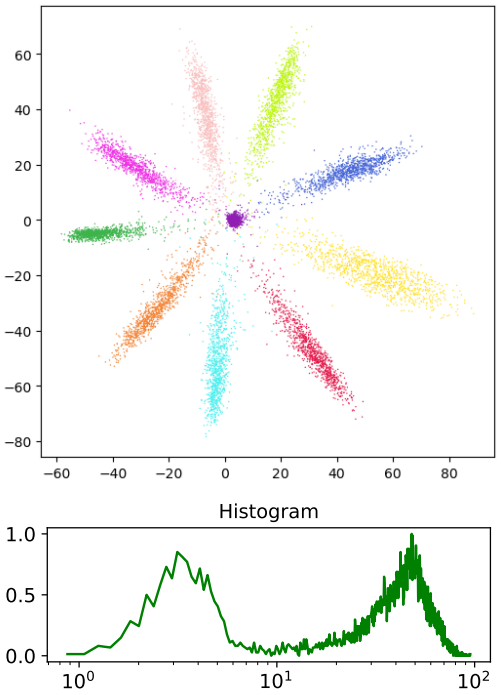
The goal of the generative model is to generate fake images similar to both classes. Hence, for Combined-GAN, We add mixed-class loss (Section 3.2) to try to drive the generated samples close to the boundary of two different known classes. Statistically, we expect that the generated samples have equal logits/softmax scores on two different classes.

At first we tried to evaluate the performance of the generative model directly by evaluating the probability of softmax score on two different known classes. Statically, assume we have a known image pair from two different classes, e.g., $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2)\}(\boldsymbol{x_1}, \boldsymbol{x_2} \in \mathcal{D}_{\mathcal{KK}}, y_1 \neq y_2)$. We can obtain $\log p_C(y_i|\boldsymbol{x_i})$ from the closed set classifier, and then we use the score $-\sum_{k=1}^{1} p_C(y_i|\boldsymbol{x_i}) \cdot \log(p_C(y_i|\boldsymbol{x_i}))$ to assess the generative model. In practice, out of our expectation, it did not work well. Instead, we evaluated the performance of open-set model to indirectly select the best generative model. Actually we found that most generated samples are classified as one of the known classes with high confidence. Since DNNs tend to be too confident in prediction, the estimated softmax score can not be assumed to represent the true probability. In our experiment, even if the generated samples are similar to the original known images and classified as one of the known classes with high probability, they can still be used as open-set samples for OSR.

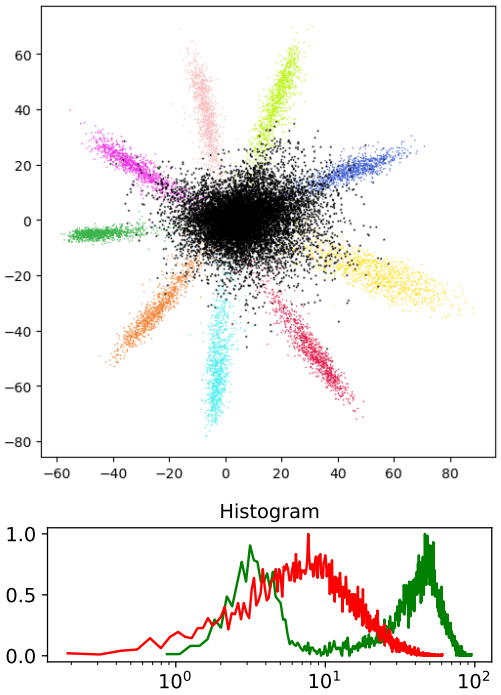
For the reconstruction loss function applied in the generative model, we use the L1 loss between reconstructed images and original images (Equation (3.7)). We also compared the results by using L2 loss and the combination of L1 and L2 loss for reconstruction loss. Compared with other loss settings, our case achieve the best results, the detail of comparison can be seen in Table A.1.

6.1.4 Model Selection During Training

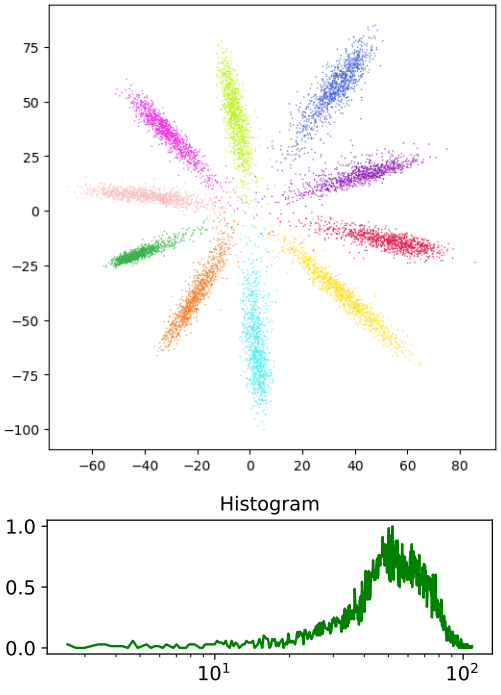
In our experiments, we train the generative model and open-set model alternatively and the generated samples are generated on the fly. We tried to use a trained generative model. After the generative model was fully trained, we used the trained generative model to generate fake samples as unknown samples for training open-set model. The trained generative model did not achieve



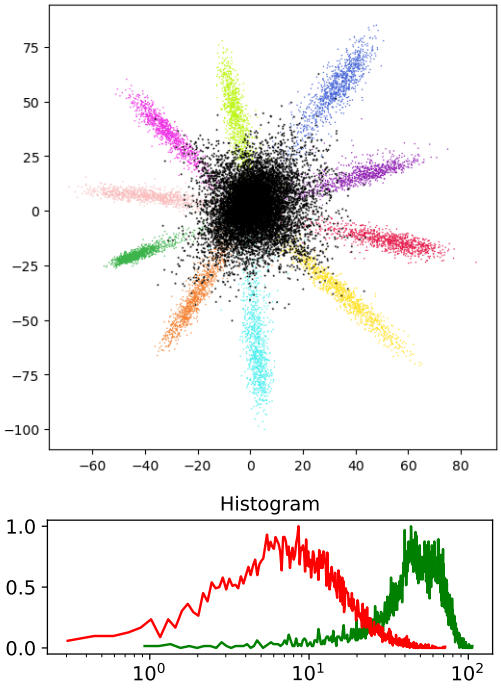
(a) Bias: True



(b) Samples from "car" and "bird"



(c) Samples from "car" and "bird"



(d) SM

Figure 6.3: LeNet++ response to MNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$). LeNet++ is trained on MNIST, and tested on MNIST and KMNIST. 2-D feature visualization plots are displayed on the top, colored dots represent samples from MNIST, while the black dots represent samples from KMNIST. The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$.

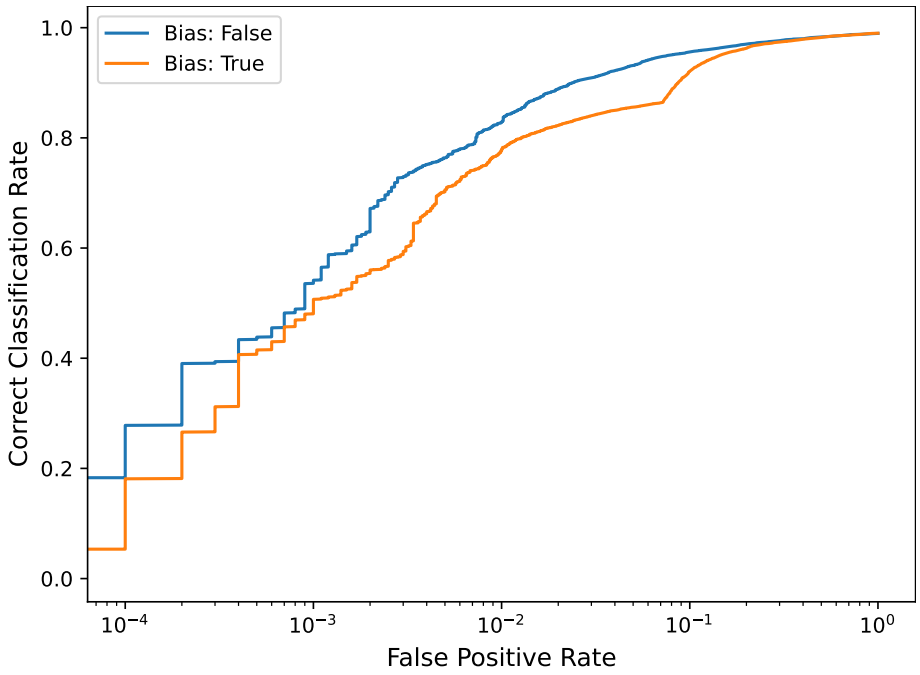


Figure 6.4: THE IMPACT OF BIAS TERM FOR LENET++. The blue line shows the results when the bias term in the last fully connected layer of LeNet++ is set to False, while the orange line shows the results when the bias term is set to True. LeNet++ is trained on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and tested on MNIST and KMNIST ($\mathcal{D}_{\mathcal{UU}}$)

better result as expected (can be seen in Table A.4) the possible reason is that noisy samples also help to improve the performance of open-set model in the early epochs.

We thought about simplifying the overall network architecture (Section 3.1) by sharing the network model of the open-set classifier and the discriminator in generative model, i.e, only one model is trained as 3 roles: C_{os}, C_{cs} and D, we attempted to adapt two different network architectures for training, but it seemed too hard for one model to tackle multi-objective task at the same time.

6.1.5 Filtering Impact on Open-Set Model

After generating fake samples from our generative model, we used all fake samples as unknown samples for training the open-set model. It should have the potential to improve the performance by processing and filtering the generated samples. For this purpose, we tried to apply filtering strategies after generating fake samples. We only kept the samples that was not classified as one of the known classes with high probability. After filtering, we added sample weight for filtered samples for balancing the filtered samples and unfiltered samples. It turned out that the filtering strategies did not improve the performance of the open-set model, as can be seen in Figure A.9. But from the attempts, it validated that even the generated samples which can be classified as one of the known classes can be useful for training open-set model.

6.1.6 Impact of the Background Class

In the experiments, we compared our approach with other approaches, including SM+BG and EOS+BG. Using the softmax function , the probability of the background class can be directly computed. In Experiment 4-5, SM+BG achieved relatively better performance, while it did not achieve a better result compared with other approaches in Experiment 1-3. The possible reason is that SM+BG is difficult to train and sometimes we need several trials until it has learned to separate all known classes well. For background approach using EOS, it usually performs better when the testing data is related to the background class during training, while when testing dataset is totally different from background class, the impact is limited, as we can see in Figure 5.1, the rank of EOS+BG falls behind.

6.2 Limitations

In this section, I will discuss some potential limitations in the thesis.

In the OSR scenarios, the open space should be infinitely large, i.e., the open-set classifier should be able to recognize unknown classes from any distributions. Since our open-set samples are generated from two different known classes, the generated images still belong to the subspace of the original space, which includes known classes. If the unknown images of the testing dataset are far from the original space of the known classes, the generated samples from our generative model can be hard to represent the unknown testing samples. Thus, the open-set classifier still has limited knowledge about the unknown testing samples and can fail to recognize and reject them, as shown in Figure 5.8(b) in Experiment 3. One promising way to solve the problem is constantly generating samples from known data and unknown data from different datasets. Since the unknown world is infinite, it is impossible to exhaust all the unknown area. However, if the open-set model is trained on more unknown data, it can recongize and reject the new unknown data with higher confidence.

The results of our approach in Experiment 4-5 (Section 5.3.1 and 5.3.2) are not as good as expected. There are some possible reasons. First, even ResNet-18++ can provide good 2 dimensional deep feature representations of CIFAR-10 and SVHN dataset, but it fails to separate the known and unknown classes for OSR, the ResNet-18++ network architecture setting maybe not good enough for OSR. Secondly, the EOS loss may be not good enough for the open-set model, even approaches using softmax thresholding achieve better results than our approaches in Experiment 4-5.

Our generative model is sometimes sensitive to datasets for OSR. The generated samples can help the open-set model achieve better results in Experiment 1-2 when detecting different unknown data. However, in Experiment 3, when LeNet++ is trained on the generated samples from FMNIST during training and used to detect unknown samples from KMNIST in testing, our approach fails to achieve comparable result (Figure 5.8(b)).

Since there are too many ways to optimize the generative model, like hyperparameter selection, loss function selection, etc. We only select part of them to evaluate the model, these can be found in Section A.3. Due to the time limit, all experiments are conducted once.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Throughout this thesis, we explored the possibility of utilizing GAN to generate effective unknown samples for tackling the OSR problem.

Based on supervised GAN, we proposed a new encoder-decoder generative model (Combined-GAN) to generate unknown samples from two different classes by combining their latent representations. By using effective combination of reconstruction loss, adversarial loss and mixed-class loss, Combined-GAN can generate fake samples similar to both classes and they can be used as open-set samples to provide information about the unknown classes. Then appropriate open-set models are trained on the generated samples and known samples to improve the classification accuracy of the known classes and detection of the known classes simultaneously. Then we conducted several experiments on multiple data sets. For each set, we selected different benchmark datasets as known classes and unknown classes. We used different generative network architectures for datasets with different image sizes. Correspondingly, we apply LeNet++ and a 2-D adaptive ResNet-18++ as open-set model for OSR. In the experiments, we used different evaluation metrics (including OSCR, AUC, confidence, etc.) to evaluate the performance of different approaches. In addition, we also provided 2-D visualization to display deep features of known and unknown samples extracted from the open-set model.

From quantitative and qualitative analysis, most results from the experiments indicate that the unknown samples generated by Combined-GAN can effectively help open-set model identify and reject new unknown samples in different OSR scenarios.

7.2 Future Work

This thesis provides a GAN-based generative model for OSR. However, there is still much potential to improve the effectiveness of the generative model for dealing with the OSR problem.

7.2.1 Latent Space Exploration

In our encoder-decoder generative model, we simply concatenated the latent representations from two different known classes. As discussed in Section 6.2, the generated samples can hardly get equal probability on both classes. It is a promising research direction to further explore the concatenated latent representations using the method in Neal et al. (2018). They used gradient descent to search in latent space in order to find the latent vector which can represent the deep

feature of open set samples. Fully exploring the latent space of known classes can possibly enable the decoder to obtain a better starting point in generating more effective fake samples for open-set classifier.

As discussed in Section 6.2, the open space is infinitely large, in order to detect the unknown space, the open-set classifier should acquire as much information of unknown space as possible during training. One possible strategy is generating unknown samples from more than 2 classes. Although different classes from the same dataset can share some similarities, especially for simple datasets like MNIST, the combined latent space can still be promising in acquiring useful information of the unknown data. Another way is to combine the latent space of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{KU}}$ that are from different datasets to generate samples similar with $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{KU}}$. We assume the more open space information the open-set classifier obtains, the better it can detect open-set samples. However, since open space is infinite, it is a tradeoff between computation and efficiency. It is possible to build an open-set system, in which we constantly classify $\mathcal{D}_{\mathcal{KK}}$ and detect $\mathcal{D}_{\mathcal{KU}}$. Meanwhile, open-set samples are generated from $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{KU}}$ to allow open-set system to gain more information about the open space.

7.2.2 Network Architecture

In this work, we only use the basic CNN models (LeNet++ and ResNet-18++) as open-set classifier. For ResNet-18++, it can provide good 2-D deep feature representation for the known classesin our experiments, however, when it is exposed with unknown classes, it fails to separate the known from unknown samples, at least in 2-D space, it is possible that ResNet-18 can work on separate the known from the unknown classes in high dimension. More variants of ResNet++ can be tested in the future.

It is also possible to try out more complicated network architectures. For the generative model, we also mainly use convolutional layers in encoder, decoder and discriminator, it would be interesting to use attention mechanism (Vaswani et al., 2017)-based GAN model to generate open-set samples.

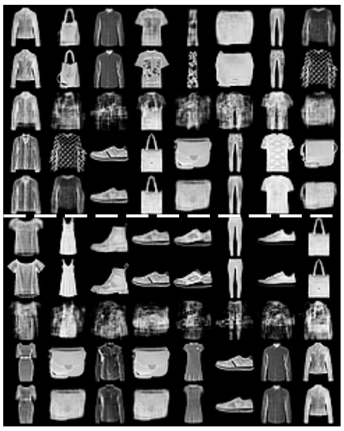
Appendix A

Attachements

A.1 Generated Samples

(a) MNIST samples

(b) KMNIST samples



(c) FMNIST samples

(d) SVHN samples

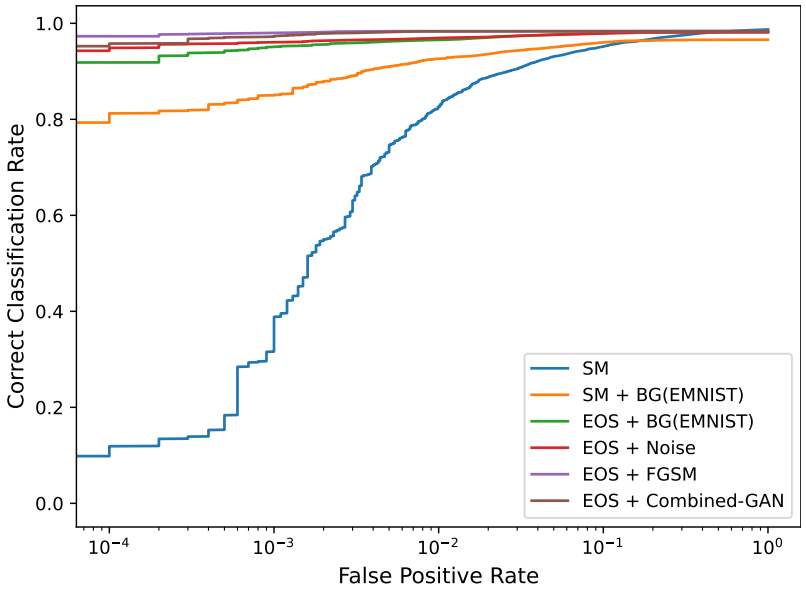
(e) CIFAR-10 samples

Figure A.1: SAMPLES GENERATED BY COMBINED-GAN. We provide some generated samples from two different classes for different benchmark datasets. Each subfigure displays two sets of images (separated by a white dash line), which include the reconstructed and the original images from one class (1st-2nd row), the generated images (3rd row), the original images and the reconstructed images from another class (4th-5th row).

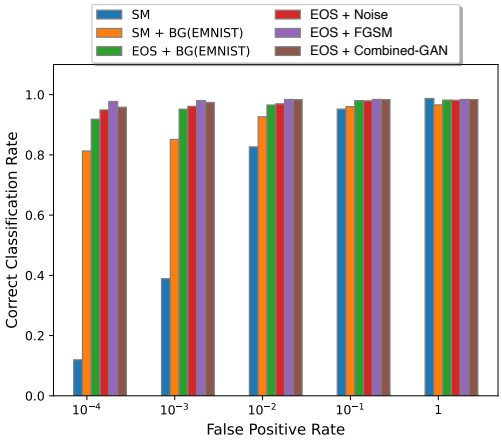
A.2 Supplement of Experiments

A.2.1 Supplement of Experiment 1

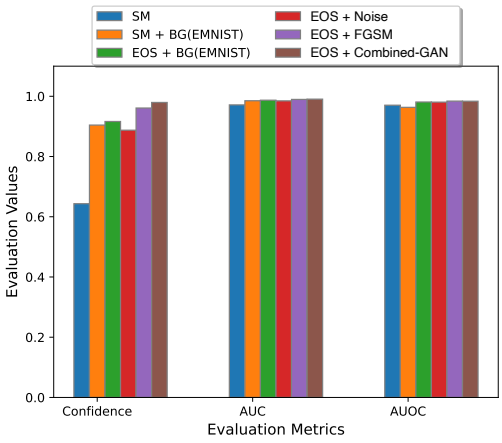
Training on MNIST and EMNIST Letters (A-M)(Background Class)



(a) OSCR curve

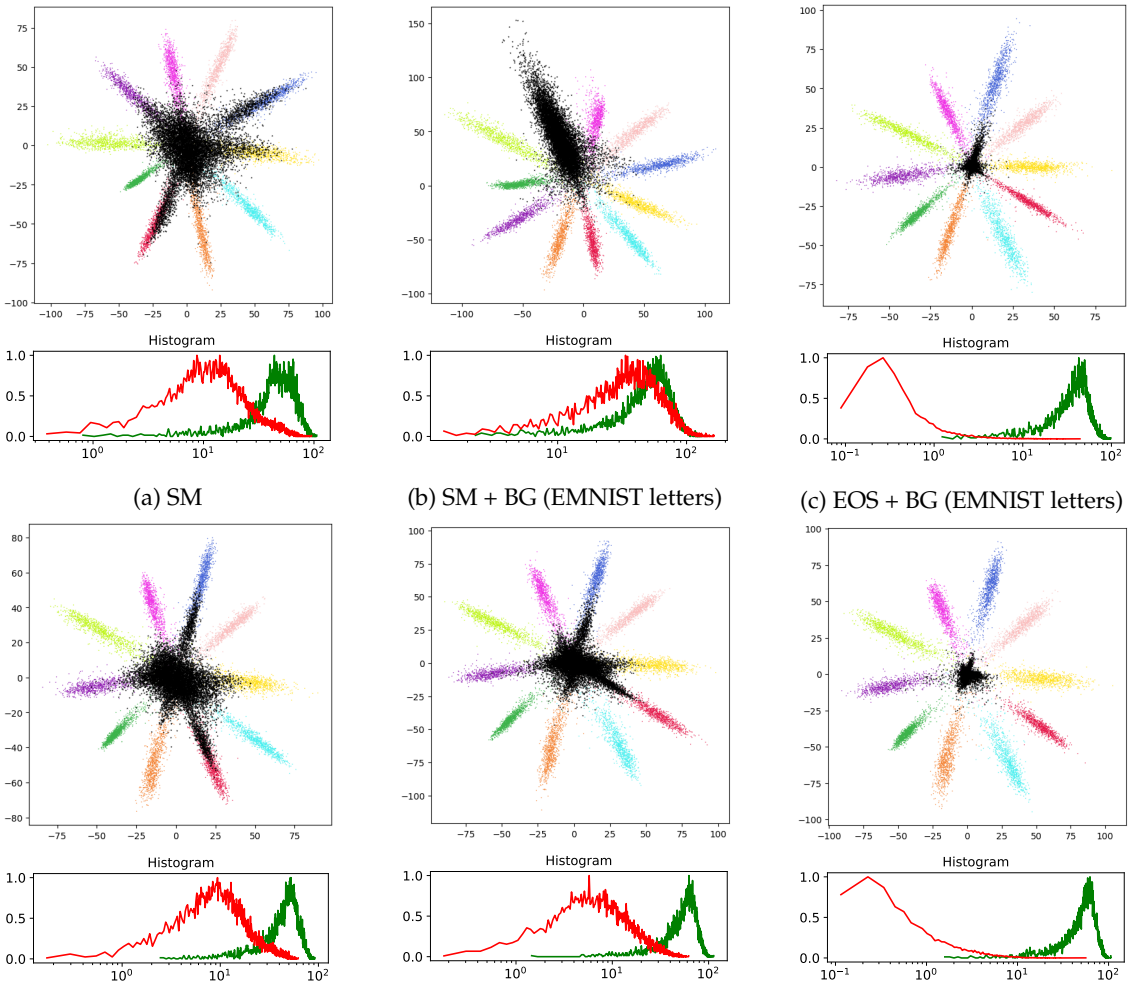


(b) CCR values at different FPR values



(c) Evaluation results

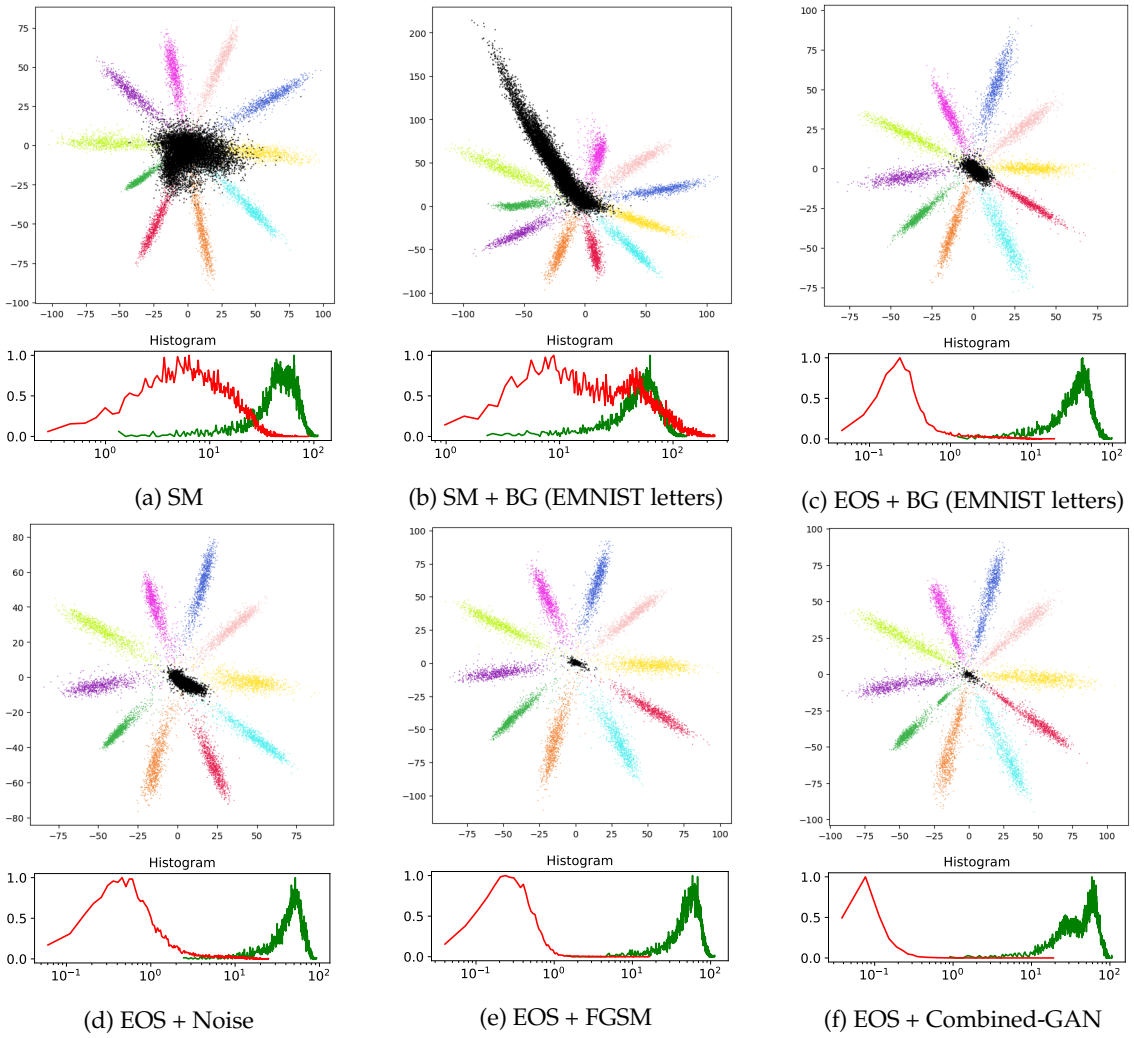
Figure A.2: Evaluation Results on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST ($\mathcal{D}_{\mathcal{UU}}$). Since FMNIST is far different from MNIST, it can be easily detected. As we can see from Figure A.2(a), most approaches get high OSCR curve and good evaluation scores (Figure A.2(c)) . For approaches using EOS, the CCR values even greater than 0.9 when the FPR is around 10^{-4} (Figure A.2(b)). EOS+FGSM approach gets slightly better results than our approach regarding OSCR results. However, all approaches have similar results regarding AUC and AUOC score (Figure A.2(c)), while our approach achieves slightly better result regarding confidence on detecting the known and unknown samples.



(a) SM (b) SM + BG (EMNIST letters) (c) EOS + BG (EMNIST letters)

(d) EOS + Noise (e) EOS + FGSM (f) EOS + Combined-GAN

Figure A.3: LeNet++ responses to MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters (N-Z) ($\mathcal{D}_{\mathcal{UU}}$). LeNet++ is trained on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters (A-M) ($\mathcal{D}_{\mathcal{KU}}$), and tested on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters (N-Z) ($\mathcal{D}_{\mathcal{UU}}$). 2-D feature visualization plots are displayed on the top, colored dots represent samples from MNIST, while the black dots represent samples from EMNIST letters (N-Z). The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$. For SM approach, the unknown samples scatter without patterns on the plot, the histogram also shows the unknown samples share large overlap with known samples, as shown in Figure A.3(a). Since SM+BG approach directly classifies the unknown samples as another class by adding one more dimension on the logit layer, we can see 11 patterns in Figure A.3(b), but the unknown samples still share large overlap with the known samples around the origin. Figure A.3(c)-Figure A.3(f) show approaches using EOS, the unknown samples are driven close to the origin. Using our approach, LeNet++ can separate $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$ more effectively and force $\mathcal{D}_{\mathcal{UU}}$ to have smaller magnitude, as shown in Figure A.3(f).



(a) SM (b) SM + BG (EMNIST letters) (c) EOS + BG (EMNIST letters)

(d) EOS + Noise (e) EOS + FGSM (f) EOS + Combined-GAN

Figure A.4: LeNet++ responses to MNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST ($\mathcal{D}_{\mathcal{UU}}$). LeNet++ is trained on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters (A-M) ($\mathcal{D}_{\mathcal{KU}}$), and tested on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST ($\mathcal{D}_{\mathcal{UU}}$). 2-D feature visualization plots are displayed on the top, colored dots represent samples from MNIST, while the black dots represent samples from FMNIST. The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$. For SM approach, the unknown samples scatter without patterns on the plot, the histogram also shows the unknown samples share large overlap with known samples, as shown in Figure A.4(a). Since Since SM+BG approach directly classifies the unknown samples as another class by adding one more dimension on the logit layer, we can see 11 patterns in Figure A.4(b), but the unknown samples still share large overlap with the known samples around the origin. Figure A.4(c)-Figure A.4(f) show approaches using EOS, the unknown samples are driven close to the origin. Using our approach, LeNet++ can separate $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$ more effectively and force $\mathcal{D}_{\mathcal{UU}}$ to have smaller magnitude, as shown in Figure A.4(f).

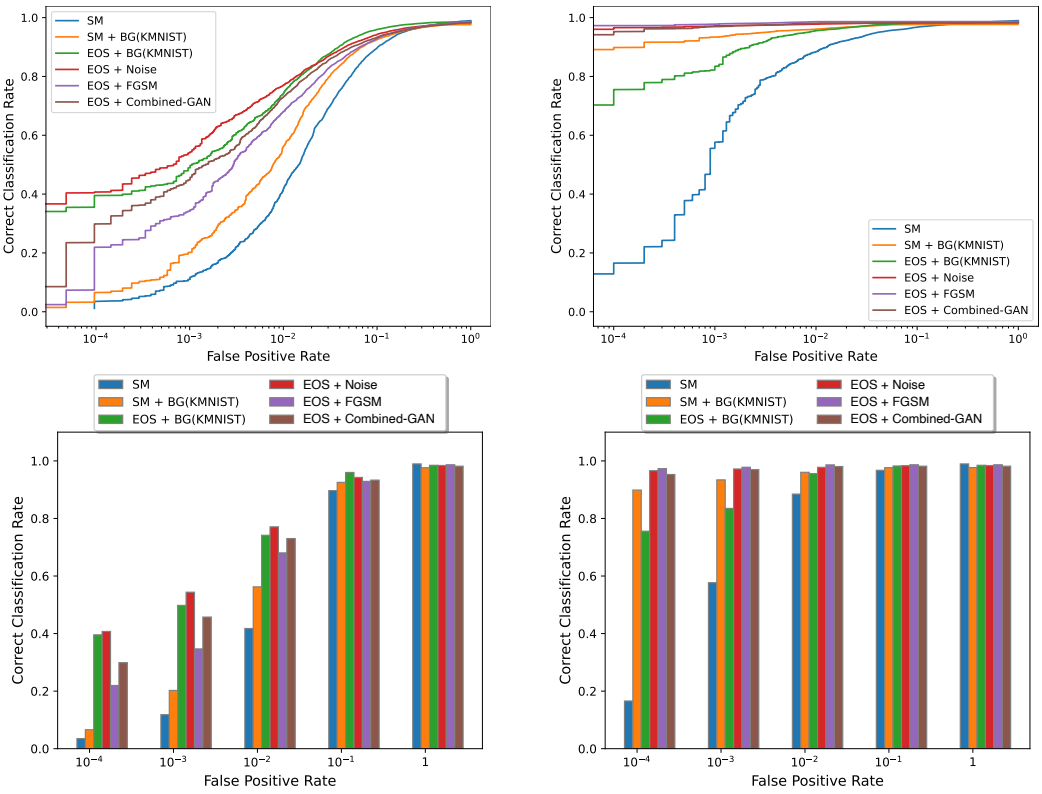
Training on MNIST and KMNIST (Background Class)

As a supplement of Experiment 1, we also use KMNIST (very different from MNIST) as the background class during training and validation. We evaluate the performance of different approaches by testing on two different datasets:

- MNIST ($\mathcal{D}_{\mathcal{K}\mathcal{K}}$) + EMNIST letters ($\mathcal{D}_{\mathcal{U}\mathcal{U}}$)
- MNIST ($\mathcal{D}_{\mathcal{K}\mathcal{K}}$) + FMNIST ($\mathcal{D}_{\mathcal{U}\mathcal{U}}$)

Figure A.5 illustrates the OSCR results when we test LeNet++ on EMNIST letters and FMNIST as $\mathcal{D}_{\mathcal{U}\mathcal{U}}$. Our approach does not have best performance compared with other approaches. Compared with the results in Experiment 1 (Section 5.2.1), in which we use EMNIST letters (A-M) as the background class, LeNet++ is more effective on detecting the unknown classes when trained on EMNIST letters than trained on KMNIST characters.

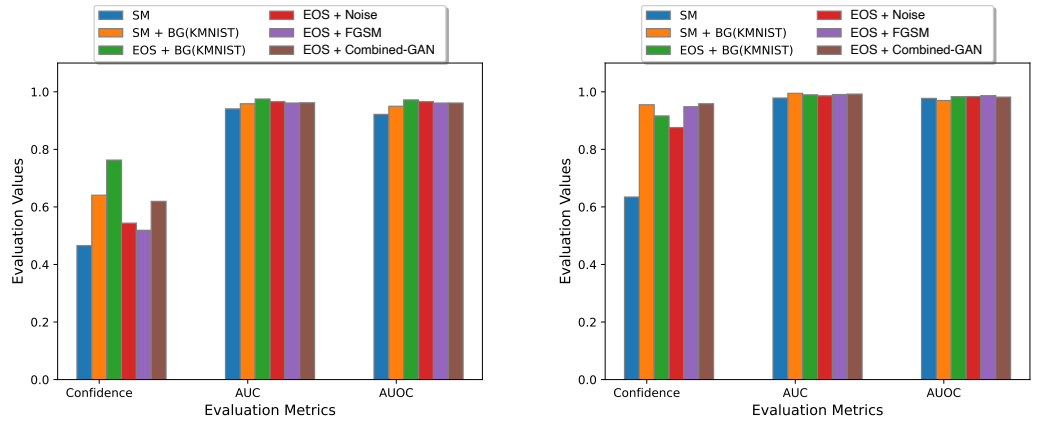
When testing on EMNIST letters as $\mathcal{D}_{\mathcal{U}\mathcal{U}}$, EOS+Noise and EOS+BG are better than other approaches, as shown in Figure A.5. For EOS+BG, the possible reason is that KMNIST characters and EMNSIT letters share some similarities. Surprisingly, when using EOS loss with random noise, it also achieves better OSCR results, but EOS+Noise can not provide good confidence in recongnizing the known and unknown samples (Figure A.6).



(a) Testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)

Figure A.5: OSCR RESULTS OF TRAINING ON MNIST ($\mathcal{D}_{\mathcal{KK}}$) (KMNIST AS BG). The plots on the top illustrate OSCR curve (Section 4.3.3) of multiple approaches testing on different datasets. The plots in the bottom shows CCR values at different FPR.

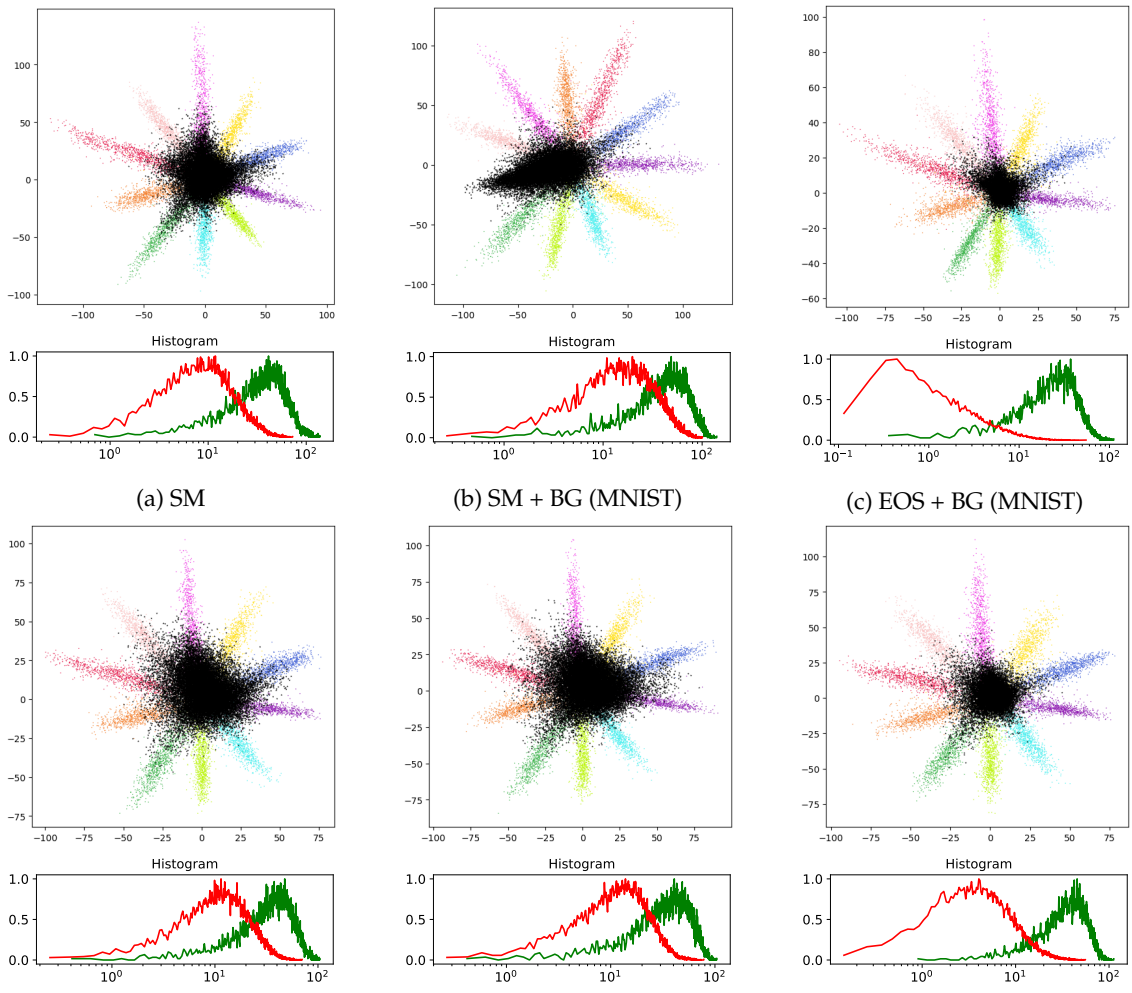


(a) Testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$)

(b) Testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and FMNIST ($\mathcal{D}_{\mathcal{UU}}$)

Figure A.6: EVALUATION RESULTS OF TRAINING ON MNIST ($\mathcal{D}_{\mathcal{KK}}$) (KMNIST AS BG). The figure shows the comparision of multiple approaches testing on different datasets.

A.2.2 Supplement of Experiment 2

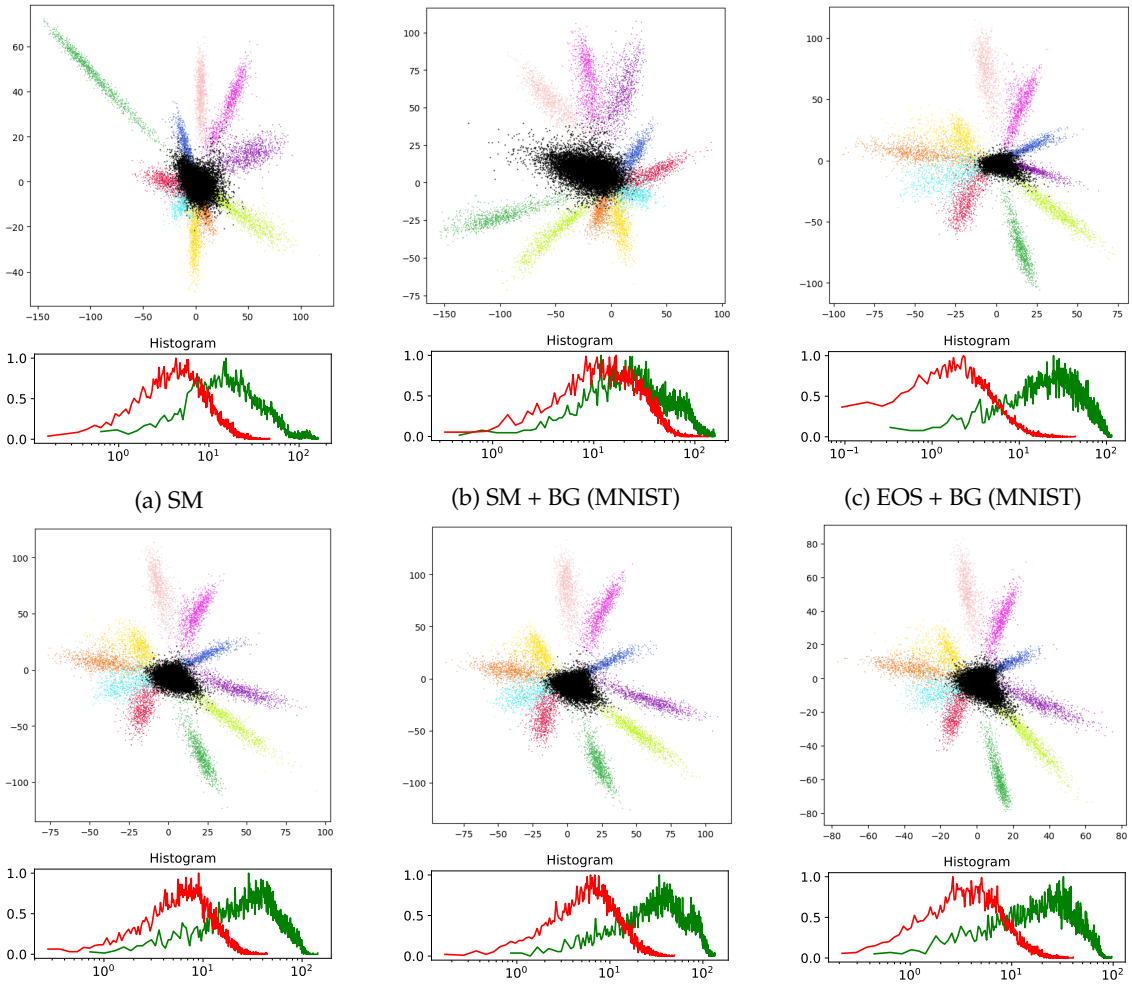


(a) SM (b) SM + BG (MNIST) (c) EOS + BG (MNIST)

(d) EOS + Noise (e) EOS + FGSM (f) EOS + Combined-GAN

Figure A.7: LeNet++ responses to KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$). LeNet++ is trained on KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and MNIST ($\mathcal{D}_{\mathcal{KU}}$), and tested on KMNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$). 2-D feature visualization plots are displayed on the top, colored dots represent samples from KMNIST, while the black dots represent samples from EMNIST letters. The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$. For SM approach, the unknown samples scatter without patterns on the plot, the histogram also shows the unknown samples share large overlap with known samples, as shown in Figure A.7(a). Since SM+BG approach directly classifies the unknown samples as another class by adding one more dimension on the logit layer, we can see 11 patterns in Figure A.7(b), but the unknown samples still share large overlap with the known samples around the origin. Figure A.7(c)-Figure A.7(f) show approaches using EOS, the unknown samples are driven close to the origin. Since EMNIST letters share some similarities with MNIST digits, after LeNet++ is trained on MNIST as the background class, when it is tested on EMNSIT letters, it can separate unknown samples from known samples more effectively than other approaches, while our approach is visually better than other approaches except for EOS+BG.

A.2.3 Supplement of Experiment 3



(a) SM (b) SM + BG (MNIST) (c) EOS + BG (MNIST)

(d) EOS + Noise (e) EOS + FGSM (f) EOS + Combined-GAN

Figure A.8: LeNet++ responses to FMNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$). LeNet++ is trained on FMNIST ($\mathcal{D}_{\mathcal{KK}}$) and MNIST ($\mathcal{D}_{\mathcal{KU}}$), and tested on FMNIST ($\mathcal{D}_{\mathcal{KK}}$) and KMNIST ($\mathcal{D}_{\mathcal{UU}}$). 2-D feature visualization plots are displayed on the top, colored dots represent samples from FMNIST, while the black dots represent samples from KMNIST. The figures in the bottom are the feature magnitude histograms for testing samples of $\mathcal{D}_{\mathcal{KK}}$ and $\mathcal{D}_{\mathcal{UU}}$. Since SM+BG approach directly classifies the unknown samples as another class by adding one more dimension on the logit layer, we can see 11 patterns in Figure A.8(b), but the unknown samples still share large overlap with the known samples around the origin. Figure A.8(c)-Figure A.8(f) show approaches using EOS, training on different $\mathcal{D}_{\mathcal{KU}}$ seems do not have any positive impact on detecting the unknown samples in testing time, since the 2-D plots are almost the same.

A.3 Model Optimization

Name	Reconstruction Loss	Accuracy	Confidence	AUC
R1	L1(img1_rec, img1) + L1(img2_rec, img2)	0.9669	**0.7705**	**0.9674**
R2	L2(img1_rec, img1) + L2(img2_rec, img2)	0.9638	0.7182	0.9613
R3	L1(img_mixed, img1) + L1(img_mixed, img2)	0.9662	0.5776	0.9366
R4	R1+0.1*R3	0.9670	0.7432	0.9643

Table A.1: COMPARISON OF DIFFERENT RECONSTRUCTION LOSSES. R1 is the reconstruction loss setting in our generative model: L1 loss between reconstructed images and original images. In addition, we used R2, R3 and R4 loss for comparison. R2 is used to compute the L2 loss between the reconstructed images and original images. R3 is used to compute the L1 loss between the generated mixed-class images and original images, because we want the generated images to look similar with original images. R4 is the combination of R1 and R3. For this comparison experiment, LeNet++ is trained on the MNIST ($\mathcal{D}_{\mathcal{KK}}$) and generated samples by our Combined-GAN with different loss functions, and then LeNet++ is evaluated on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$). The evaluation results (best results presented in bold font) show that L1 loss for the reconstruction loss funtion between reconstructed images and original images provides the best results, especially for confidence and AUC socre.

Testing Dataset	Approach	Confidence	AUC	AUOC
MNIST+EMNIST	SM	0.4595	0.924	0.9102
	EOS + Combined-GAN (DCGAN)	0.7342	0.9831	0.9818
	EOS + Combined-GAN (WGAN-GP)	**0.7724**	**0.9837**	**0.9818**
MNIST+FMNIST	SM	0.6055	0.9685	0.9676
	EOS + Combined-GAN (DCGAN)	**0.979**	**0.9901**	**0.9842**
	EOS + Combined-GAN (WGAN-GP)	0.9769	0.9897	0.9835

Table A.2: COMPARISON BETWEEN DCGAN AND WGAN-GP. In DCGAN (Radford et al., 2016), binary cross entropy loss is during training, while WGAN-GP (Gulrajani et al., 2017) measures the distance between the generated data distribution and original data distribution. We compare the results of using the loss function in DCGAN and WGAN-GP in our Combined-GAN. For this comparison experiment, LeNet++ is trained on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and generated samples from MNIST, and then we test on different datasets. The evaluation results (best results presented in bold font) show the evaluation performance of LeNet++ model using generated samples from our Combined-GAN. From the results, when testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$), WGAN-GP performs better, while when testing on FMNIST as unknown samples, DCGAN achieves better results.

λ_1	Accuracy	Confidence	AUC
1	0.9540	0.6947	0.9449
10	0.9529	0.6932	0.9421
50	**0.9669**	**0.7705**	**0.9674**
100	0.9528	0.7252	0.9492

(a) λ_1 selection

λ_2	Accuracy	Confidence	AUC
1	**0.9638**	**0.7349**	**0.9634**
10	0.9629	0.6095	0.9470
50	0.9628	0.5668	0.9317
100	0.9645	0.5794	0.9349

(b) λ_2 selection

λ_3	Accuracy	Confidence	AUC
1	0.9647	0.6161	0.9485
10	0.9643	0.7447	0.9642
50	0.9687	0.7390	0.9673
100	**0.9649**	**0.8019**	**0.9692**

(c) λ_3 selection

Table A.3: HYPERPARAMETER SELECTION FOR COMBINED-GAN MODEL. We compare several values when selecting hyperparameters for our Combined-GAN model (Section 3.2). For this comparison experiment, LeNet++ is trained on the MNIST ($\mathcal{D}_{\mathcal{KK}}$) and generated samples by our Combined-GAN with different hyperparameters, and then LeNet++ is evaluated on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$). From the evaluation results (best results presented in bold font), we select $\lambda_1 = 50$, $\lambda_1 = 1$ and $\lambda_1 = 100$ for our all experiments.

Testing Dataset	Approach	Confidence	AUC	AUOC
MNIST+EMNIST	SM	0.5856	0.8930	0.8779
	EOS + Combined-GAN	**0.7482**	**0.9782**	**0.9773**
	EOS + Combined-GAN (Trained)	0.7169	0.9745	0.9738
MNIST+KMNIST	SM	0.6173	0.9604	0.9591
	EOS + Combined-GAN	**0.9071**	**0.9864**	**0.9829**
	EOS + Combined-GAN (Trained)	0.8558	0.9844	0.9813
MNIST+FMNIST	SM	0.6144	0.9569	0.9557
	EOS + Combined-GAN	**0.9784**	**0.9894**	**0.9832**
	EOS + Combined-GAN (Trained)	0.9457	0.9883	0.9821

Table A.4: COMPARISON BETWEEN COMBINED-GAN AND TRAINED COMBINED-GAN. In our work, unknown samples are generated on the fly, thus, our generative model and open-set model are trained alternatively. In comparison, we trained the Combined-GAN first, after the our generative model is fully trained, we use it to generate unknown samples for open-set model. For this comparison experiment, LeNet++ is trained on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and generated samples from MNIST, and then we test on different datasets. The results show the evaluation performance of LeNet++ model using generated samples from our Combined-GAN. From the results (the best results presented in bold font), when testing on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$), Combined-GAN performs slightly better regarding AUC and AUOC, since in the early training time, the generative can only generate noisy images, the open-set samples can focus on classifying the known classes, it gets better confidence than the trained generative model.

Latent Representation Dimension	Accuracy	Confidence	AUC	AUOC
64	0.9845	0.8064	0.9857	0.9830
128	0.9809	0.7307	0.9794	0.9776
256	**0.9860**	**0.8089**	**0.9871**	**0.9846**
512	0.9845	0.8064	0.9857	0.9830

Table A.5: COMPARISON OF DIFFERENT LATENT REPRESENTATION DIMENSIONS. We compare different dimensions of latent representation for the small generative model (Table 4.2a). For this comparison experiment, LeNet++ is trained on the MNIST ($\mathcal{D}_{\mathcal{KK}}$) and generated samples by our Combined-GAN with different loss functions, and then LeNet++ is evaluated on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$). The evaluation results (best results presented in bold font) show that when we set the dimension of latent representation as 256, LeNet++ achieves the best result. For simplicity, we set 256 as the dimension of latent space for small generative model for all experiments in our work.

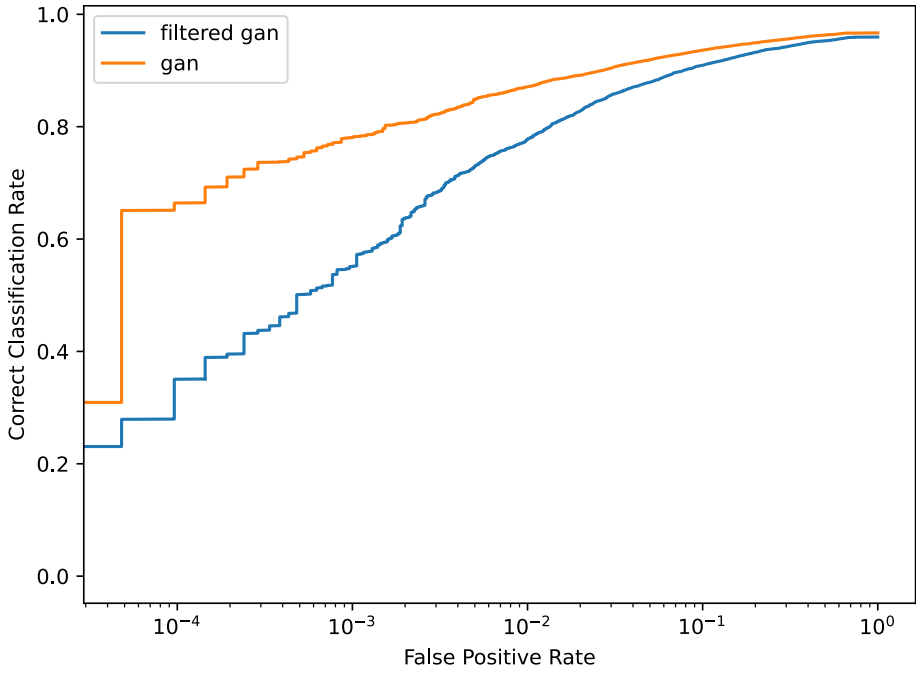


Figure A.9: IMPACT OF FILTERING ON THE GENERATED SAMPLES. For this comparison experiment, LeNet++ is trained on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and generated samples from MNIST, and tested on MNIST ($\mathcal{D}_{\mathcal{KK}}$) and EMNIST letters ($\mathcal{D}_{\mathcal{UU}}$). The evaluation results (best results presented in bold font) evaluate the performance of LeNet++ using all generated samples or generated samples after filtering. As we can see, when LeNet++ is trained on all generated samples, it can achieve better OSCR results.

List of Figures

List of Tables

List of Acronyms

AUOC	Area Under the OSCR Curve
AUC	Area Under the Curve
CNN	Convolutional Neural Network
DNN	Deep Neural Network
EOS	Entropic Open-Set
FGSM	Fast Gradient Sign Method
FPR	False Positive Rate
GAN	Generative Adversarial Network
OSCR	Open-Set Classification Rate
OSR	Open Set Recognition
TPR	True Positive Rate

Bibliography

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Bendale, A. and Boult, T. E. (2016). Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572.

Berthelot, D., Schumm, T., and Metz, L. (2017). BEGAN: boundary equilibrium generative adversarial networks. *CoRR*, abs/1703.10717.

Bishop, C. M. et al. (1995). *Neural networks for pattern recognition*. Oxford university press.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*.

Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters (2017). *arXiv preprint arXiv:1702.05373*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Dhamija, A. R., Günther, M., and Boult, T. (2018). Reducing network agnostophobia. *Advances in Neural Information Processing Systems*, 31.

Farnia, F. and Ozdaglar, A. (2020). Do GANs always have nash equilibria? In *International Conference on Machine Learning*, pages 3029–3039. PMLR.

Geng, C., Huang, S.-J., and Chen, S. (2021). Recent advances in open set recognition: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3614–3631.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Goodfellow, I., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein GANs. *Advances in neural information processing systems*, 30.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, Z., Zuo, W., Kan, M., Shan, S., and Chen, X. (2019). AttGAN: Facial attribute editing by only changing what you want. *IEEE transactions on image processing*, 28(11):5464–5478.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Jain, L. P., Scheirer, W. J., and Boult, T. E. (2014). Multi-class open set recognition using probability of inclusion. In *European Conference on Computer Vision*, pages 393–409. Springer.

Jo, I., Kim, J., Kang, H., Kim, Y.-D., and Choi, S. (2018). Open set recognition by regularising classifier with fake data generated by generative adversarial networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2686–2690.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Cortes, C., and Burges, C. (2010). MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*.

Liu, M.-Y. and Tuzel, O. (2016). Coupled generative adversarial networks. *Advances in neural information processing systems*, 29:469–477.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Neal, L., Olson, M., Fern, X., Wong, W.-K., and Li, F. (2018). Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 613–628.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

Pan, Z., Yu, W., Yi, X., Khan, A., Yuan, F., and Zheng, Y. (2019). Recent progress on generative adversarial networks (GANs): A survey. *IEEE Access*, 7:36322–36333.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Rozsa, A., Günther, M., and Boult, T. E. (2017). Adversarial robustness: Softmax versus openmax. *arXiv preprint arXiv:1708.01697*.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242.

Scheirer, W. J., de Rezende Rocha, A., Sapkota, A., and Boult, T. E. (2013). Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772.

Scheirer, W. J., Jain, L. P., and Boult, T. E. (2014). Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2317–2324.

Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., and Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging*, pages 146–157. Springer.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*, abs/1708.07747.

Yu, Y., Qu, W.-Y., Li, N., and Guo, Z. (2017). Open-category classification by adversarial sample generation. *arXiv preprint arXiv:1705.08722*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

Zongyuan Ge, S. D. and Garnavi, R. (2017). Generative openmax for multi-class open set classification. In Tae-Kyun Kim, Stefanos Zafeiriou, G. B. and Mikolajczyk, K., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 42.1–42.12. BMVA Press.