**Introduction**

**GMDL222, Final Project**

# Ofir Gilad

The Open Set Recognition problem is one of the most famous topics that researchers keep investigating until today.  
Given a distribution with K-classes, the mission is to find a model that discriminates open-set data that belongs to a (K+1) “other” class, outside the K closed-set classes.  
In this project we have tried different models to solve this problem, and looked for articles and researches on this topic to learn new ideas on how to solve this problem.

Some of our models had performed better than the others, but on each try we gained new insights until we found the model can achieve the best results for this problem.

**Data & Preprocessing**

The datasets, transformations and the dataloaders we have used were as follows:

**The datasets we have used are:**

We used the MNIST training and testing datasets as our given distribution with 10-classes of the well-known digits.  
And the CIFAR10 testing dataset as an out of distribution dataset, which we changed a little bit to our experiments, as we will describe shortly.

**The transformations we have used are:**

The MNIST dataset we have normalized with mean and standard deviation of 0.5.  
The CIFAR10 dataset we have normalized with mean of [0.4914, 0.4822, 0.4465] and standard deviation [0.2471, 0.2435, 0.2616], resized to a size of 28x28, and transformed the images to grayscale.

This was done since we trained our models to work on images with one channel of colors on sizes of 28x28.

**The dataloaders we have used are:**

**Training dataloders:**

30,000 randomly selected images from the MNIST training dataset, where we had split to:

1. 80% of the images to the training dataloader.
2. 20% of the images to the validation dataloader.

**GAN Training dataloader:**

A dataloader with all the 60,000 images from the MNIST training dataset for the OpenGAN training.

**MNIST Testing dataloader:**

A dataloader with all the 10,000 images from the MNIST testing dataset.

**OOD Testing dataloader:**

A dataloder with 10,500 randomly selected images as a testing dataset as follows:

1. 10,000 of the images are all the images on the MNIST testing dataset.
2. 500 of the images are from the CIFAR10 dataset with the transformation we have defined above.

**Models**

The models we were using for our experiments are:

1. Binary CNNs per Class:

The Baseline model was a combination of 10 Binary CNNs, where each CNN was trained to classify a specific digit, and by votes of the CNNs, the image got its classification.

The OSR model was built from the baseline model with an addition of anomaly detection function that if the results from the CNNs were unexpected for a given image, the image got classified as Unknown.

The architecture of each one of the CNN was: 2 Convolutional Layers and 3 Fully Connected Layers

1. CNN with Autoencoder

The Baseline model was a CNN trained with an Autoencoder, where the Autoencoder was learning to reconstruct the digits images, and the CNN was trained to classify both the original images and the reconstructed images.

For the OSR model we tried 2 different separation methods with the combination of these 2 models:

1. The first method detects differences on the classifications of the CNN on the original and the reconstructed images and based on that classify the image as Unknown.
2. The second method was the same as the first one but with an addition of checking the probabilities differences with a given threshold to verify if the image is Unknown.

The architecture of the CNN was: 2 Convolutional Layers and 2 Fully Connected Layers with dropout.  
And the architecture of the Autoencoder was: 4 Fully Connected Layer for the encoder and the decoder.

1. OpenGAN with CNN (The chosen model)

The Baseline model is the CNN with dropouts as we saw in the practical session.

The OSR model was built with the [[1](#Reference1)] Discriminator and the CNN from the Baseline model, which based on the Discriminator output:

1. If the image was found as “real”, it got classified based on the output from the CNN.
2. If the image was found as “fake”, it got classified as Unknown.

The architecture of the CNN is: 2 Convolutional Layers and 2 Fully Connected Layers with dropout.  
And the architecture of the GAN is: 5 Fully Convolutional Layers for the Discriminator and for the Generator.

We have also tried to use training, as seen on the research’s linked [GitHub](https://github.com/aimerykong/OpenGAN), by converting each given image to 1D feature vector and sending it to the GAN.

We got perfect score on the OOD testing **only** if the dataloader with the open&closed-set of images wasn’t shuffling between them on each given batch (As seen on the GitHub, where at first the output for the images from the closed-set was calculated, then the output for the images from the open-set, and evaluation of the model with AUROC metric).

As we investigated the issue, we found out that the Batch Norm Activation Function calculations are different in training and testing modes (And depend on the distribution of the given batch).

Without it we couldn’t achieve the same performance. Therefore, we preferred to use the training, without Batch Norm Activation Function, which helped to model to generalize better on batches with shuffled images from the open&closed-sets.

**Training**

Each one of our experimented models had a different type of training:

1. The Binary CNNs per Class model was trained as follows:

Each one of the CNNs was trained for 10 epochs on the same input training set, with different labels from each other.

To each CNN, according to the digit which it represents, we changed all the labels of the images representing that digit to 1, and the labels of all the other images to 0.

The criterion we used to train the CNNs was the Cross Entropy Loss.

The graph that we got:

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Notice: The Loss and the Accuracy are calculated as the mean of the results from all the CNNs. The overall Loss of the model is getting smaller with each epoch, and the overall Accuracy might look unstable from the graph, but converges to 82%, where each CNN reaches its optimal accuracy.

1. The CNN with Autoencoder model was trained as follows:

At first, we trained the Autoencoder for 25 epochs to generate good reconstructions of the images from the input training set.

Then we train the CNN for 50 epochs on both the input images from the input training set and the reconstruction of those images that were created by the trained Autoencoder.

Lastly, we took the CNN hyper parameters that got the highest accuracy on the validation set and reloaded the CNN with them.

The criterions we used were:

1. The MSE Loss to the Autoencoder.
2. The Cross Entropy Loss to the CNN.

The graphs we got:

A picture containing chart

Description automatically generated A picture containing chart

Description automatically generated

A picture containing text

Description automatically generated

1. The OpenGAN with CNN model was trained as follows:

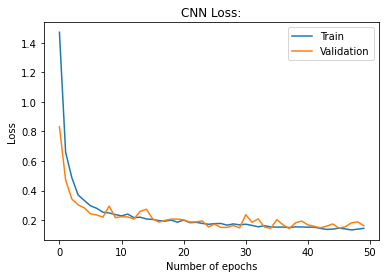
At first, we trained the CNN on the same input training set for 50 epochs and reloaded it with hyper parameters that got the highest accuracy on the validation set.

Then, we trained the GAN for 50 epochs on all the training set of the MNIST dataset (the whole 60,000 images) through OpenGAN training, which was done by training the discriminator with both fake data (synthesized from the generator) and real open-training examples, and updating the generator based on the discriminator results.

The criterions we used were:

1. The Cross Entropy Loss to the CNN.
2. The BCE Loss to the OpenGAN.

The graphs we got for the training (The chosen model):

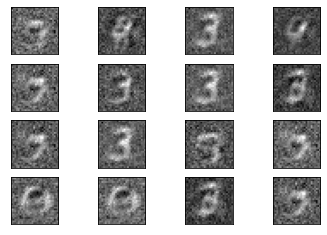
 A picture containing text

Description automatically generated

Chart

Description automatically generated

The trained generator results:



The graphs we got for the training:

Shape

Description automatically generated with medium confidenceA picture containing text

Description automatically generated

Chart, histogram

Description automatically generated

Notice: In this model the generator was trained to generate feature vectors and not images.   
Therefore, we couldn’t convert the generator outputs back to images.

Also, since we trained the CNN again for this model, we got different Baseline, OOD and OSR results.

**Evaluation**

**Baseline results:**

The results we got for each baseline model were:

1. The Binary CNNs per Class model:

Number Of Images Tested = 10000

Model Accuracy = 93.89999999999999

A picture containing text, electronics

Description automatically generated

Notice: For a case where none of the CNNs agreed, the image was labeled 0 by default.

1. The CNN with Autoencoder model:

Number Of Images Tested = 10000

Model Accuracy = 95.16

A picture containing text, monitor, electronics

Description automatically generated

1. The OpenGAN with CNN model (CNN results):

|  |  |
| --- | --- |
| (The chosen model) |  |
| Number Of Images Tested = 10000  Model Accuracy = 95.49 | Number Of Images Tested = 10000  Model Accuracy = 95.55 |
|  |  |

**OSR rational:**

1. The Binary CNNs per Class model:

In this method, for a given image, we check the output of each one of the CNNs and if we found an anomaly behavior**\***, we classified the image as Unknown.  
Otherwise, the image was classified as the number of the CNN that returned an output of “1” for it.

**\***Our definition for anomaly behavior was:  
If none of the CNNs was able to return an output of “1” for the given image, or at least two CNNs were returning an output of “1”.

The rationales behind this method were:

1. If each one of the CNNs will be able to distinguish images of 1 digit from all the others with high accuracy, an image from the MNIST would be classified as “1” by only one of the CNNs.  
   And as a result we expect that the CNN, which returned output of “1” for the image, is representing the true label of the image.
2. The CNNs weren’t trained on OOD images, and therefore we expect an anomaly behavior to occur.  
   Because if the given image is OOD, we expect that the CNNs will guess the label of it with a probability close to 0.5 and it will be more likely that an output of only “0” from all the CNNs or at least two outputs of “1” will be returned.

The experiment went not so bad. The model was able to classify 89% of the images from the OOD dataset as Unknown (445 out of 500 were classified as OOD). But more than 600+ images from the MNIST were wrongly classified as Unknown.

From this model we noticed a few things:

1. From the Baseline results, we saw that a lot of images were classified as “0” (by default), and we speculate that since there were images that were very similar to two or more digits, the CNNs that were supposed to return output of “1”, returned output of “0” instead.
2. As we checked different sizes of training sets for the model, we saw that as the training set gets bigger, the model accuracy on images from the MNIST dataset increases, but the accuracy on images from the OOD dataset decreases.   
   And this is due to the decrease of anomaly behaviors of the model, as it becomes more confident about its results.

Because we wanted to find a model that can be more accurate on the MNIST, and less naïve than the model we have tried, we looked for more possible options.

That’s when we thought about using Autoencoder that learned how to reconstruct images from the MNIST training dataset, to use them as another training dataset for our baseline model to increase its variability.  
And if the Autoencoder failed to reconstruct an image, it got classified as Unknown.

1. The CNN with Autoencoder model:

In this method, for a given image, we created a reconstruction of the image using the trained Autoencoder, and then we send both the original image reconstruction of it to our CNN and compare the results we got.  
For each of the OSR separation methods we tested we checked that:

1. For the first method we compared only the difference of the output labels from the CNN for the original and the reconstructed images, and if they were different the image was classified as Unknown.
2. For the second method other than checking the difference in the output labels, we also checked the difference in the probabilities for the chosen label of the CNN on the original image and the probability of the same label on the reconstruction of the image.  
   If the difference between the two results was bigger than the threshold of “0.5”, the image was classified as Unknown.

The rationales behind those models were:

1. Well-trained Autoencoder will generate good reconstructions of images from the MNIST dataset and will fail to do so on images from the OOD dataset. Therefore, if a huge difference was found between the original image and the reconstruction of it, the image is more likely to come from the OOD dataset.
2. The CNN model was trained on the original images from the MNIST dataset and the reconstructions of them from the trained Autoencoder. Therefore, CNN was learning how to classify both real and fake MNIST images and is more likely to detect outliers in the MNIST dataset.

The experiment didn’t go the way we expected it to be.

For the first method the was able to classify 89% of the images from the OOD dataset as Unknown (447 out of 500 were classified as OOD). But more than 2800+ images from the MNIST were wrongly classified as Unknown.

And for the second method the model was able to classify 60% of the images from the OOD dataset as Unknown (360 out of 500 were classified as OOD). But more than 2500+ images from the MNIST were wrongly classified as Unknown.

From this model we have noticed a few things:

1. The Autoencoder reconstructs the original images the way it sees fit.  
   Therefore we notice that some of the images representing one digit, got reconstructed as an image of a different digit, which resulted in confusing our CNN, and as seen in the OOD results, the CNN was failing to classify the original images and the reconstructions of them with the same label. Which resulted in a lot of images from the MNIST dataset classified wrong as Unknown.
2. The first separation method is better than the second separation methods to correctly classify images from the OOD dataset as Unknowns, at the cost of misclassifying more images from the MNIST dataset as Unknowns.   
   But overall, these methods weren’t good enough to achieve great accuracy for our problem, and a different approach was required.

We believe that this model can be improved, and maybe we weren’t using the Autoencoder correctly.

We thought about trying different approaches like comparing differences in one or more of the hidden layers of the CNN or the Autoencoder. But due to the complexity of rebuilding the model for doing so, we preferred to look for other methods instead, before trying it out.

That’s when we thought about using a GAN to solve the OSR problem:  
By defining the images from the MNIST dataset as “real” images and the images from the OOD dataset as “fake” images, we can train the GAN discriminator to be an open-set discriminator.

1. The OpenGAN with CNN model:

In this method, for a given image, based on the trained Discriminator output:

1. If the image was found as “real”, it got classified based on the output from the CNN.
2. If the image was found as “fake”, it got classified as Unknown.

The rationales behind it were:

1. OpenGAN trains with both the real open&closed-set data and the fake open-data into a single (GAN-like) mini-max optimization over D and G:

Which is equivalent to training a normal GAN and using its discriminator as the open-set likelihood function. And that’s exactly what we are looking for.

1. The CNN model was trained to classify images from the MNIST with high accuracy. Therefore, if an image from the testing dataset was found “real” by the discriminator, the CNN will predict the true label of the image with high accuracy as seen on the baseline results.

Since the discriminator output is between 0 and 1, the threshold we chose was “0.5” where: An output above 0.5 mean that the discriminator has high confidence that the image is “real” (Known), and under 0.5 that the image is “fake” (Unknown).

The generated samples from the trained Generator look like a very noisy version of MNIST images.   
But we expected it to happen:   
“Because a perfectly trained generator G would generate realistic closed-set images, eventually making the discriminator D inapplicable for open-set discrimination.”

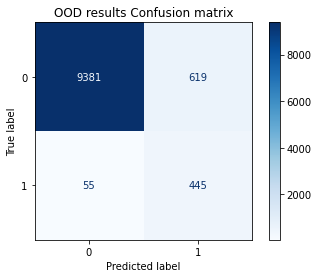
**OOD results:**

The results we got for each OSR model were:

1. The Binary CNNs per Class model:

Number Of Images Tested = 10500

Model Accuracy = 93.58095238095238



1. The CNN with Autoencoder model:

|  |  |
| --- | --- |
| *Difference by labels only* | *Difference by labels and probabilities* |
| Number Of Images Tested = 10500  Model Accuracy = 72.26666666666667 | Number Of Images Tested = 10500  Model Accuracy = 73.40952380952382 |
|  |  |

1. The OpenGAN with CNN model:

|  |  |
| --- | --- |
| (The chosen model) |  |
| Number Of Images Tested = 10500  Model Accuracy = 97.28571428571429 | Number Of Images Tested = 10500  Model Accuracy = 100.0 |
|  |  |

**OSR results:**

The results we got for each OSR model were:

1. The Binary CNNs per Class model:

Number Of Images Tested = 10500

Model Accuracy = 92.65714285714286

A picture containing text, monitor, electronics

Description automatically generated

1. The CNN with Autoencoder model:

|  |  |
| --- | --- |
| *Difference by labels only* | *Difference by labels and probabilities* |
| Number Of Images Tested = 10500  Model Accuracy = 71.42857142857143 | Number Of Images Tested = 10500  Model Accuracy = 71.1047619047619 |
|  |  |

1. The OpenGAN with CNN model:

|  |  |
| --- | --- |
| (The chosen model) |  |
| Number Of Images Tested = 10500  Model Accuracy = 93.4952380952381 | Number Of Images Tested = 10500  Model Accuracy = 95.76190476190476 |
|  |  |

**Prediction about the model performance on other OOD data**

Our method’s success relies on the Discriminator output.

Since in the OpenGAN training we trained the Discriminator on both the real open&closed-set data and the fake open-data, by using the only the MNIST dataset as our training dataset, we expect that our Discriminator behavior will stay the same on any other new testing datasets, and not only datasets with OOD images as transformed images from CIFAR10 dataset.

If the given images are from OOD dataset that are distributed differently than the MNIST dataset, we expect that our Discriminator will classify them as “fake” with “negative” high confidence (with output very close to “0”).

And if the images are from the MNIST dataset they will be classified as “real” by our Discriminator with “positive” high confidence (with output very close to “1”), and the CNN will classify them correctly with almost the same accuracy as seen in the baseline results.

Our OSR model is composed of two models that work “independently”:  
That the CNN output doesn’t depend on the Discriminator output and vice versa (In contrast to what was done on the CNN with the Autoencoder model).

Therefore, the performance of one model doesn’t affect the performance of the other model, and that’s in our opinion, one of the strongest advantages of our model.

**How to load the trained model**

The OSR model uses only the CNN and the Discriminator from the OpenGAN training.

Therefore, only the weights of the CNN and the Discriminator are required, but we also added the Generator weights as it’s part of our OpenGAN model.

The CNN weights are under the file: “cnn\_parameters.pth”

The Discriminator weights are under the file: “discriminator\_parameters.pth”

The Generator weights are under the file: “generator\_parameters.pth”

In the final section of the notebook there are all the required imports, classes, and code sections with explanations of how to load our model and check that it was loaded correctly.

**Notices:**

1. Before evaluating our model, make sure it was set to evaluation mode!
2. Don’t use the OSR model for training, as it wasn’t designed to handle this kind of procedure.

**References**

[1] Kong, S., & Ramanan, D. (2021). Opengan: Open-set recognition via open data generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 813-822).

<https://openaccess.thecvf.com/content/ICCV2021/papers/Kong_OpenGAN_Open-Set_Recognition_via_Open_Data_Generation_ICCV_2021_paper.pdf>