GMDL - Final Project

Project Summary

Ben Tuvia

208988634

Method and Rationale:

- For the Open Set Recognition (OSR) approach, I've designed a convolutional neural network (CNN) tailored to distinguish between in-distribution (known) and out-of-distribution (unknown) data. The model was trained on the MNIST dataset, which consists of 10 classes of handwritten digits. To handle unknown data, I've implemented a threshold-based method applied to the output probabilities from a Softmax layer. The model classifies a sample as 'unknown' if the highest SoftMax probability is below a specified threshold. This approach allows the model to effectively recognize known classes while identifying samples from the CIFAR10 (which is used as a placeholder for the real OOD data) dataset as out-of-distribution.
- In previous attempt, I've explored Monte Carlo (MC) dropout, which introduces randomness during inference to generate multiple predictions for uncertainty estimation. While this method can be effective in some scenarios, it didn't perform well for the OSR task. Specifically, MC dropout resulted in significant overlap between known and unknown predictions, leading to poor differentiation between in-distribution and out-of-distribution samples. This experience led me to adopt the threshold-based method, which provided more consistent and interpretable results.

1. Model Architecture Hyperparameters

- **Input Size:**
  - The input size of our model is 28x28, corresponding to the dimensions of the images in the MNIST dataset. Each input image is a grayscale image, which is represented as a single-channel (1x28x28) tensor. The convolutional layers in the model take this input size and process it to extract relevant features for classification.
- **Hidden Sizes:**
  - The model consists of two convolutional layers and two fully connected layers:
    - **Conv1:** The first convolutional layer has 32 filters, each of size 3x3, with a stride of 1 and padding of 1. This layer is followed by a ReLU activation function and max-pooling, reducing the spatial dimensions while increasing the depth to 32 feature maps.
    - **Conv2:** The second convolutional layer has 64 filters, again with a size of 3x3, stride of 1, and padding of 1. This layer also uses ReLU and max-pooling, further reducing the spatial dimensions and increasing the depth to 64 feature maps.
    - **FC1:** The first fully connected layer takes the flattened output of the convolutional layers (which is a vector of size 64x7x7 = 3136) and maps it to 128 hidden units.
- **Output Size:**
  - The final fully connected layer (FC2) maps the 128 hidden units to the output size. For the baseline model, the output size is 10, corresponding to the 10 classes of the MNIST dataset. In the OSR model, the output size is 11, where the

additional output class represents the 'unknown' class for out-of-distribution detection.

2. Learning Rate

- The learning rate is a critical hyperparameter that controls the step size at each iteration while moving toward a minimum of the loss function. For both the baseline and OSR models, i selected a learning rate of **0.001**. This learning rate was chosen after testing different values and finding that 0.001 provided stable training without causing oscillations or slow convergence. It allowed the model to learn effectively from the training data while avoiding overshooting the optimal solution.

3. Batch Size

- The batch size determines the number of training samples used to compute the gradients before updating the model parameters. I used a batch size of **64**, which is a commonly used value in deep learning. This batch size strikes a good balance between efficient use of memory and the ability to generalize from the training data. A larger batch size might have made the training faster but could have required more memory, while a smaller batch size could have led to noisier gradient estimates and slower convergence.

4. Epochs

- The number of epochs represents how many times the entire training dataset is passed through the model during training. I set the number of epochs to **3** for both models. This choice was made based on the model's convergence behavior. Given the simplicity of the MNIST dataset and the efficiency of the convolutional network, three epochs were sufficient for the models to achieve good performance without overfitting.

5. Threshold for OOD Detection

- The threshold for out-of-distribution (OOD) detection is a key hyperparameter in the OSR model. It determines the cutoff for classifying a sample as 'unknown' based on the Softmax output probabilities. I set this threshold to **0.8** after experimenting with various values. A threshold of 0.8 means that if the model's highest SoftMax probability for a given sample is below 80%, the sample is classified as 'unknown'. This value was chosen because it provided a good trade-off between accurately classifying in-distribution samples and correctly identifying OOD samples as 'unknown'.

6. Optimizer Choice

- I used the **Adam optimizer** for training both the baseline and OSR models. Adam is a popular choice because it combines the advantages of both the AdaGrad and RMSProp algorithms, which adapt the learning rate individually for each parameter. Adam also incorporates momentum, which helps in accelerating convergence and smoothing out fluctuations in the learning process. Given its adaptive nature, Adam works well across a wide range of problems, and in our case, it allowed for efficient and stable training with minimal hyperparameter tuning.

7. Loss Function

- For both the baseline and OSR models, i used the **CrossEntropyLoss** function. Cross-entropy loss is the standard choice for classification problems, as it measures the difference between the true label distribution and the predicted label distribution output by the model. It penalizes incorrect classifications based on the confidence of the model's predictions, encouraging the model to output high probabilities for the correct classes. This loss function was particularly suitable for this task because it effectively handled the multi-class classification problem posed by the MNIST dataset and the extended classification in the OSR model.

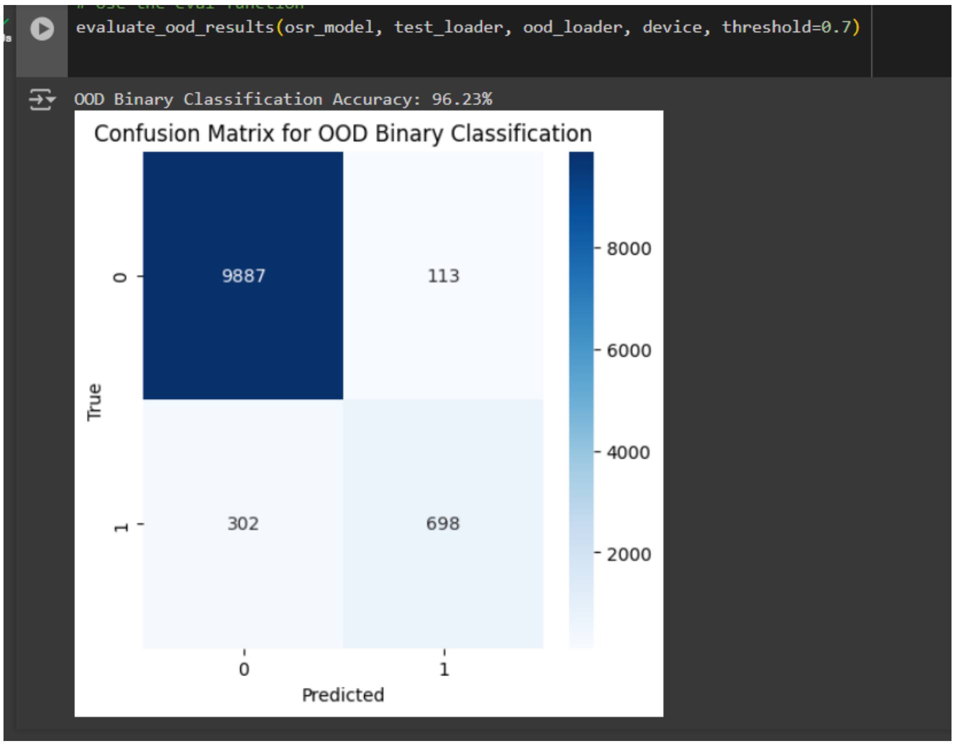
Limitation of the approach:

One of the main limitations of my approach is its reliance on a fixed Softmax probability threshold to distinguish between known and unknown samples. This method assumes that the confidence scores produced by the model are well-calibrated, which might not always be the case. For instance, the model may still misclassify some out-of-distribution samples as known if their Softmax outputs are misleadingly high. Additionally, my approach might not perform well on datasets where the boundary between in-distribution and out-of-distribution samples is less clear or where the OOD samples are very similar to the known classes.

The method is expected to perform well on datasets like MNIST, where the classes are clearly defined and distinct from typical out-of-distribution datasets like CIFAR10. However, it may struggle with more complex or ambiguous datasets where the difference between known and unknown classes is less obvious, such as distinguishing between digits and letters. Furthermore, the use of a fixed threshold may not generalize well across different datasets, necessitating careful tuning of the threshold for each new scenario.

Previous attempts with changed hyper-paramters:

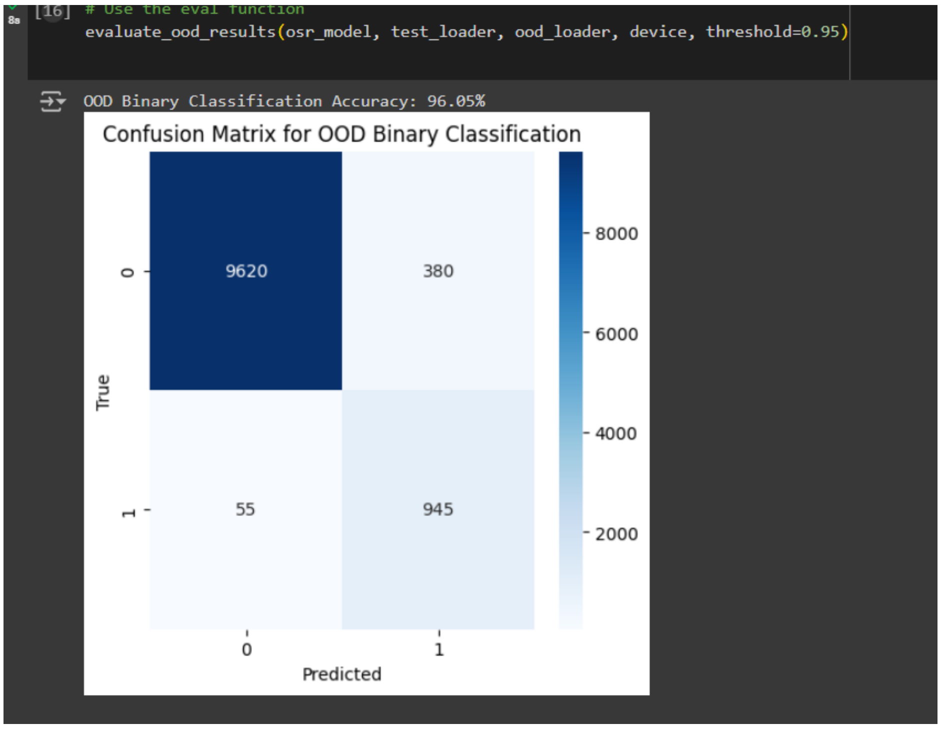
- Previous attempts on figuring out the best threshold parameter:
    - 0.7 (OOD results):



```
evaluate_ood_results(osr_model, test_loader, ood_loader, device, threshold=0.7)
```

```
OOD Binary Classification Accuracy: 96.23%
```

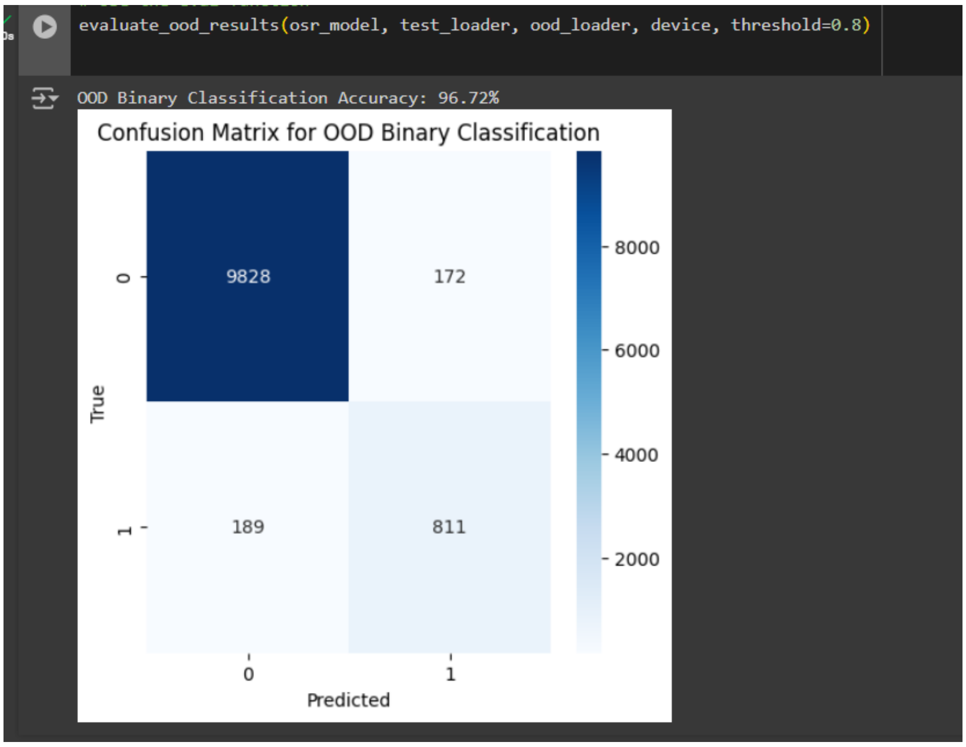
    - 0.95 (OOD results):



```
[16] # Use the eval function
evaluate_ood_results(osr_model, test_loader, ood_loader, device, threshold=0.95)
```

```
OOD Binary Classification Accuracy: 96.05%
```

- 0.8(OOD results) - **<u>optimal</u>**:



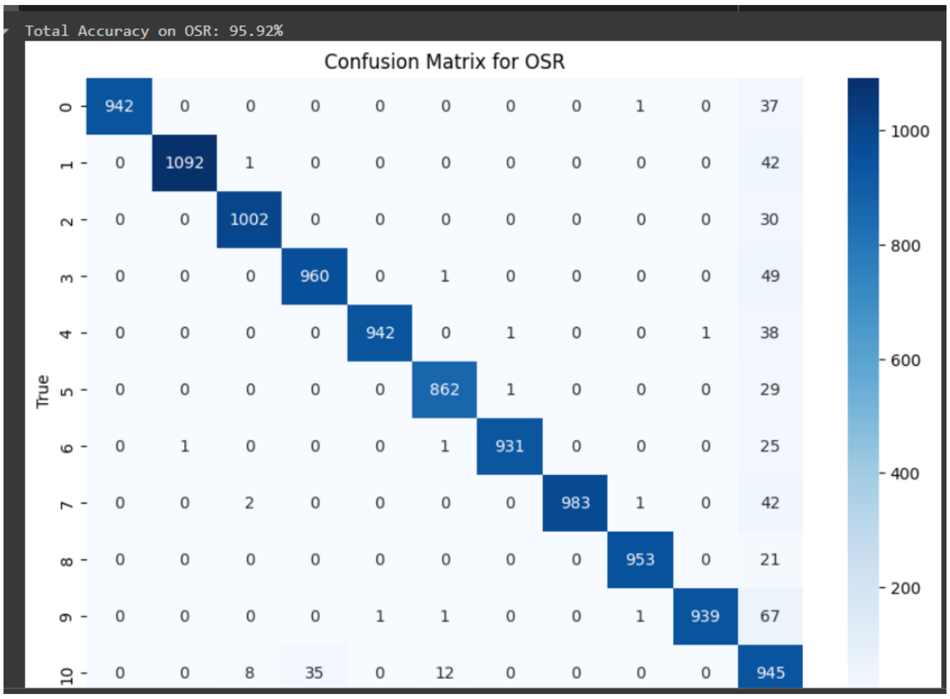
```
evaluate_ood_results(osr_model, test_loader, ood_loader, device, threshold=0.8)
```

```
OOD Binary Classification Accuracy: 96.72%
```

Confusion Matrix for OOD Binary Classification

True \ Predicted	0	1
0	9828	172
1	189	811

- 0.95 (OSR results):

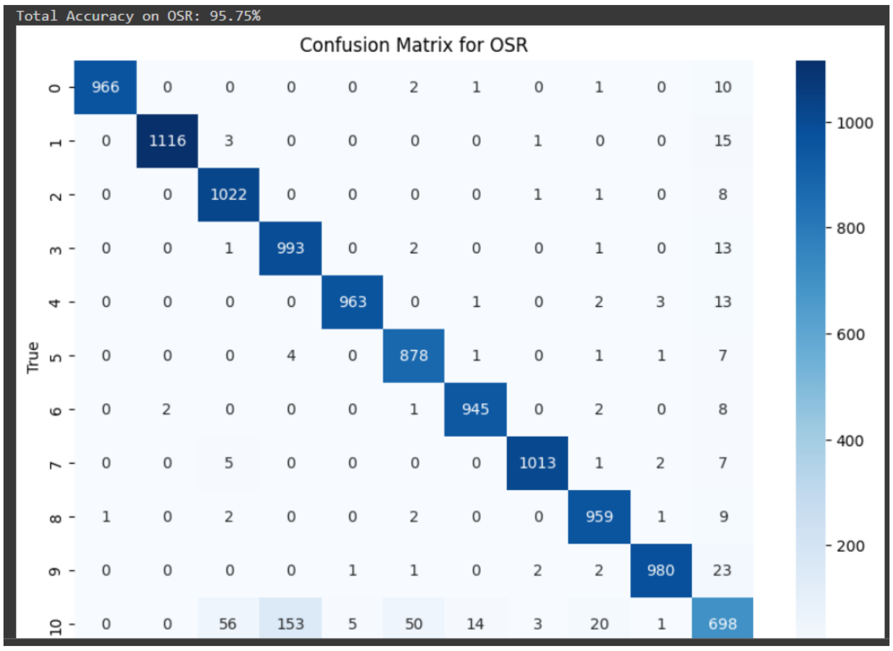


```
Total Accuracy on OSR: 95.92%
```

Confusion Matrix for OSR

True											
0	942	0	0	0	0	0	0	0	1	0	37
1	0	1092	1	0	0	0	0	0	0	0	42
2	0	0	1002	0	0	0	0	0	0	0	30
3	0	0	0	960	0	1	0	0	0	0	49
4	0	0	0	0	942	0	1	0	0	1	38
5	0	0	0	0	0	862	1	0	0	0	29
6	0	1	0	0	0	1	931	0	0	0	25
7	0	0	2	0	0	0	0	983	1	0	42
8	0	0	0	0	0	0	0	0	953	0	21
9	0	0	0	0	1	1	0	0	1	939	67
10	0	0	8	35	0	12	0	0	0	0	945

○ 0.7 (OSR results):

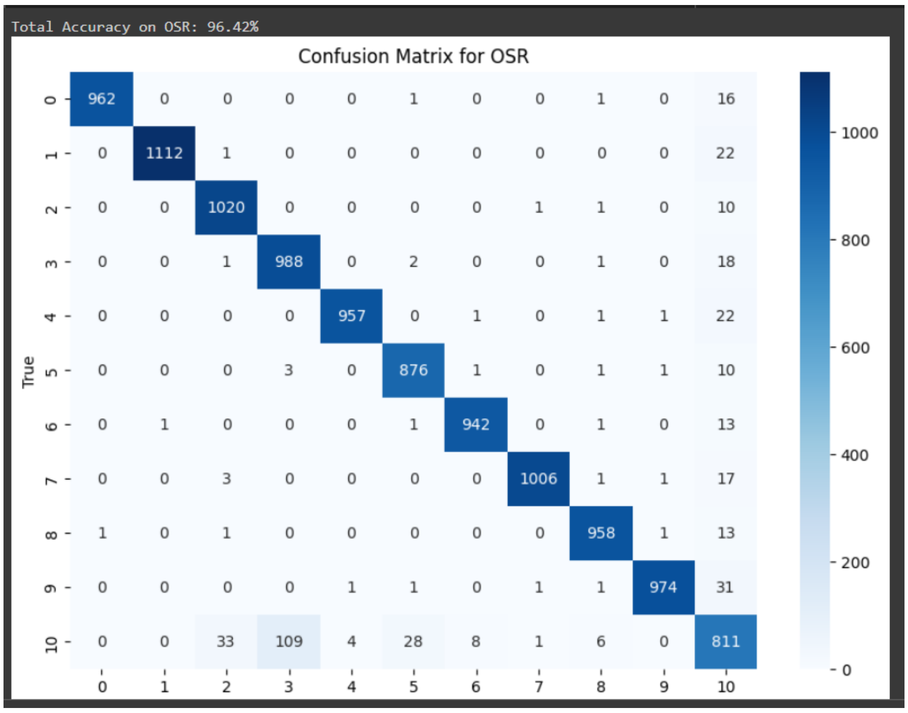


```
Total Accuracy on OSR: 95.75%
```

Confusion Matrix for OSR

True											
0	966	0	0	0	0	2	1	0	1	0	10
1	0	1116	3	0	0	0	0	1	0	0	15
2	0	0	1022	0	0	0	0	1	1	0	8
3	0	0	1	993	0	2	0	0	1	0	13
4	0	0	0	0	963	0	1	0	2	3	13
5	0	0	0	4	0	878	1	0	1	1	7
6	0	2	0	0	0	1	945	0	2	0	8
7	0	0	5	0	0	0	0	1013	1	2	7
8	1	0	2	0	0	2	0	0	959	1	9
9	0	0	0	0	1	1	0	2	2	980	23
10	0	0	56	153	5	50	14	3	20	1	698

○ 0.8 (OSR results) - **optimal**:

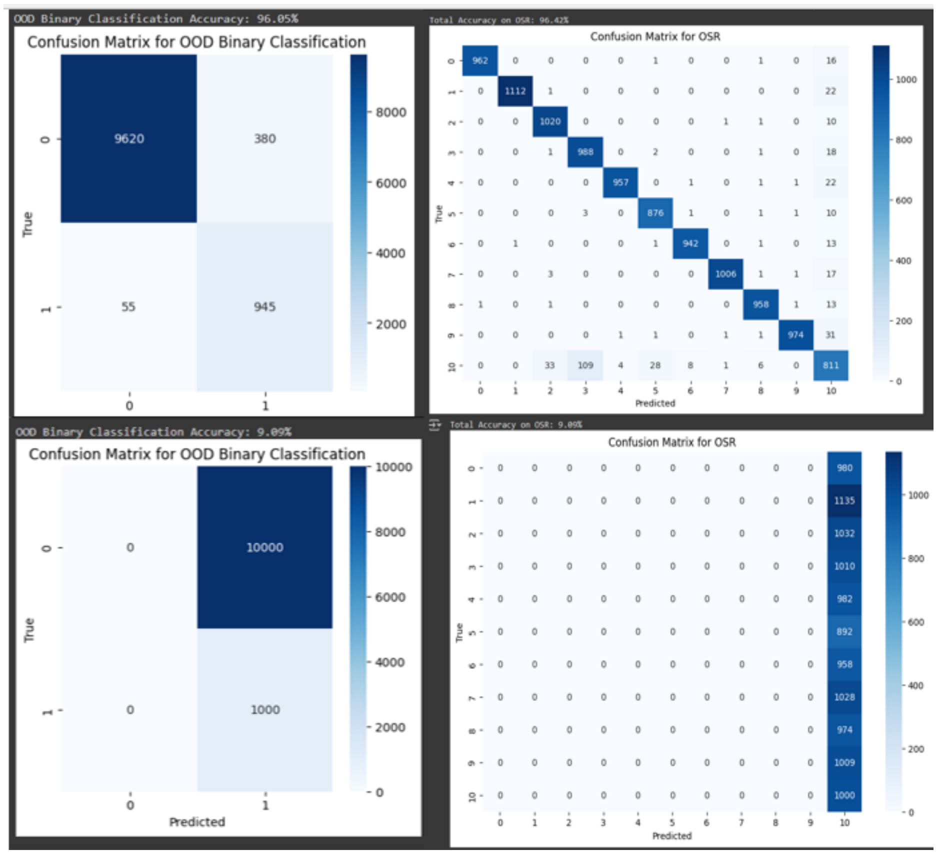


```
Total Accuracy on OSR: 96.42%
```

Confusion Matrix for OSR

True	0	1	2	3	4	5	6	7	8	9	10
0	962	0	0	0	0	1	0	0	1	0	16
1	0	1112	1	0	0	0	0	0	0	0	22
2	0	0	1020	0	0	0	0	1	1	0	10
3	0	0	1	988	0	2	0	0	1	0	18
4	0	0	0	0	957	0	1	0	1	1	22
5	0	0	0	3	0	876	1	0	1	1	10
6	0	1	0	0	0	1	942	0	1	0	13
7	0	0	3	0	0	0	0	1006	1	1	17
8	1	0	1	0	0	0	0	0	958	1	13
9	0	0	0	0	1	1	0	1	1	974	31
10	0	0	33	109	4	28	8	1	6	0	811

- Previous attempts of learning rate:
  My previous attempt of determining the correct learning rate has started with 0.01, which preformed shockingly bad – here is comparison between this attempt (bottom) to the final learning rate (up - 0.001):



```
OOD Binary Classification Accuracy: 96.05%
```

True \ Predicted	0	1
0	9620	380
1	55	945

```
Total Accuracy on OSR: 96.42%
```

True \ Predicted	0	1	2	3	4	5	6	7	8	9	10
0	962	0	0	0	0	1	0	0	1	0	16
1	0	1112	1	0	0	0	0	0	0	0	22
2	0	0	1020	0	0	0	0	1	1	0	10
3	0	0	1	988	0	2	0	0	1	0	18
4	0	0	0	0	957	0	1	0	1	1	22
5	0	0	0	3	0	876	1	0	1	1	10
6	0	1	0	0	0	1	942	0	1	0	13
7	0	0	3	0	0	0	0	1006	1	1	17
8	1	0	1	0	0	0	0	0	958	1	13
9	0	0	0	0	1	1	0	1	1	974	31
10	0	0	33	109	4	28	8	1	6	0	811

```
OOD Binary Classification Accuracy: 9.09%
```

True \ Predicted	0	1
0	0	10000
1	0	1000

```
Total Accuracy on OSR: 9.09%
```

True \ Predicted	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	980
1	0	0	0	0	0	0	0	0	0	0	1135
2	0	0	0	0	0	0	0	0	0	0	1032
3	0	0	0	0	0	0	0	0	0	0	1010
4	0	0	0	0	0	0	0	0	0	0	982
5	0	0	0	0	0	0	0	0	0	0	892
6	0	0	0	0	0	0	0	0	0	0	958
7	0	0	0	0	0	0	0	0	0	0	1028
8	0	0	0	0	0	0	0	0	0	0	974
9	0	0	0	0	0	0	0	0	0	0	1009
10	0	0	0	0	0	0	0	0	0	0	1000