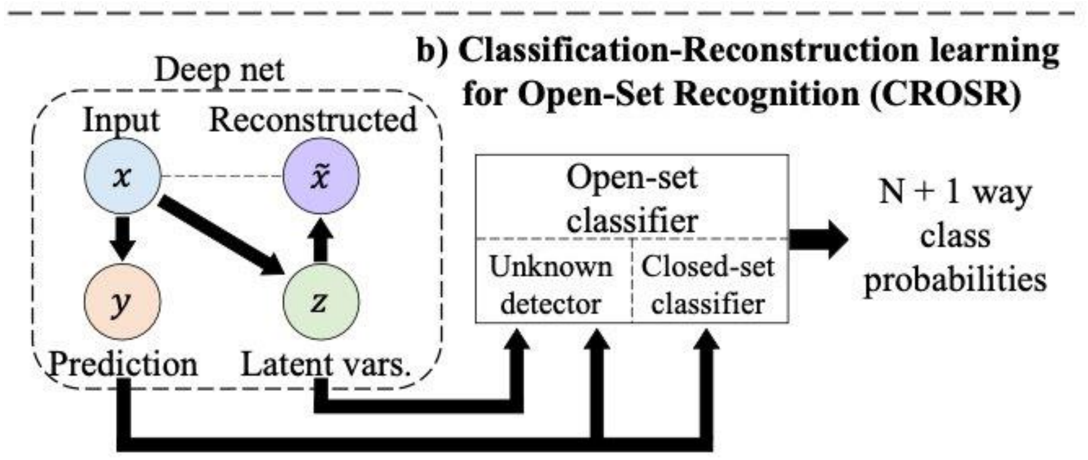
Project Open Set Recognition

Approach:

In our initial research, we thoroughly investigated the open set recognition (OSR) problem, focusing on its challenges, applications, and recent advancements as discussed in article [1]. One method that yielded the most promising results, which also resonated with our understanding from the course, involves a combination of traditional classification techniques and a reconstruction mechanism derived from an autoencoder. Given our familiarity with these topics and the documented success of this approach on the MNIST dataset (as mentioned in article [1]), we decided to implement and further explore this region.

CROSR (Clarification Reconstruction Open Set Recognition):

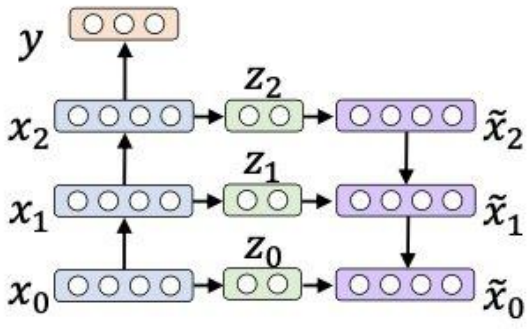
The classification approach for the OSR capitalizes on the concept that by feeding training images into an autoencoder and minimizing the reconstruction error, the latent features extracted hold not only crucial information about the image but also effectively represent the overall data distribution. This methodology is exemplified in article [2], where the general strategy is illustrated with the following image:



In this approach, the input undergoes not merely a classification process that generates a logits vector corresponding to the number of classes, but also leverages the latent variables to forge a new classifier with a logits vector of size N + 1 (N representing the known classes within the original dataset, also referred to as Known Knowns). The network described in the article ingeniously creates such latent representations, integrating them with the classification process to enhance connectivity and extract more valuable data, as depicted in the subsequent image:

$$
\begin{aligned}
\boldsymbol{x}_{l+1} &= \boldsymbol{f}_l(\boldsymbol{x}_l), \\
\boldsymbol{z}_l &= \boldsymbol{h}_l(\boldsymbol{x}_l), \qquad (7) \\
\tilde{\boldsymbol{x}}_l &= \boldsymbol{g}_l(\tilde{\boldsymbol{x}}_{l+1} + \tilde{\boldsymbol{h}}_l(\boldsymbol{z}_l)).
\end{aligned}
$$

In this model,
f represents a step-forward classification layer employing a specific convolutional setup.
hh functions as a dimension-reduction encoder, while g serves as a non-linear reconstruction function. This architecture synergistically combines all components, enhancing the value and complexity of the latent data represented in each zi. Ultimately, we preserve the latent representations zl and return x0 for assessing reconstruction error and facilitating the backward pass, as illustrated in the subsequent image:



This is the general structure of DHRnet (Deep hierarchical reconstruction net) the one that was presented in article [2].

Our network:

Our network is implemented in the same manner as described in article [2] for the MNIST data set.
We used a seven-layer plain CNN . It consists of five convolutional layers with 3 × 3 kernels and 100 out- put channels, followed by ReLU non-linearities. Max pool- ing layers with a stride of 2 are inserted after every two con- volutional layers. At the end of the convolutional layers, we put two fully connected layers with 500 and 10 units,.

The unknown detector:

This section of our project proved to be the most challenging to navigate, primarily because the latent representations and logits generated offered numerous potential applications. We will sequentially present the methods we employed, delineating those we categorized as failures and explaining our rationale for retaining or discarding each approach:

Evaluating the mode:

To evaluate our model, we adopted the methodology described in article [1], utilizing a validation set comprising both CIFAR-10 and Fashion MNIST. During the validation phase, we assessed in-distribution accuracy, out-of-distribution accuracy, total accuracy, and the F1 score for open set recognition as outlined in article [1]. Subsequently, we implemented a cross-validation fold method that generates a validation subset from the training data and equitably distributes the OOD data, thereby ensuring a rigorous test of our model's robustness. In the cross-validation setup, the ratio between in-distribution and OOD data is 9:1, ensuring a match, while outside of cross-validation, the ratio is adjusted to 3:2.

$$P_{ma} = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FP_i}, R_{ma} = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FN_i} \quad (14)$$

$$P_{mi} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C}(TP_i + FP_i)}, R_{mi} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C}(TP_i + FN_i)} \quad (15)$$

here is the F scores formula as mentioned in article [1]:
Where formula 14 represents the Macro-F-Measure which averages the precision and recall for each class and then combines them.
And formula 15 represents the Micro-F-Measure which calculates precision and recall globally by considering the aggregate contributions of all classes.
In our case we want to see that both values are close to each other and both values are as high as possible(ranging form 0-1)

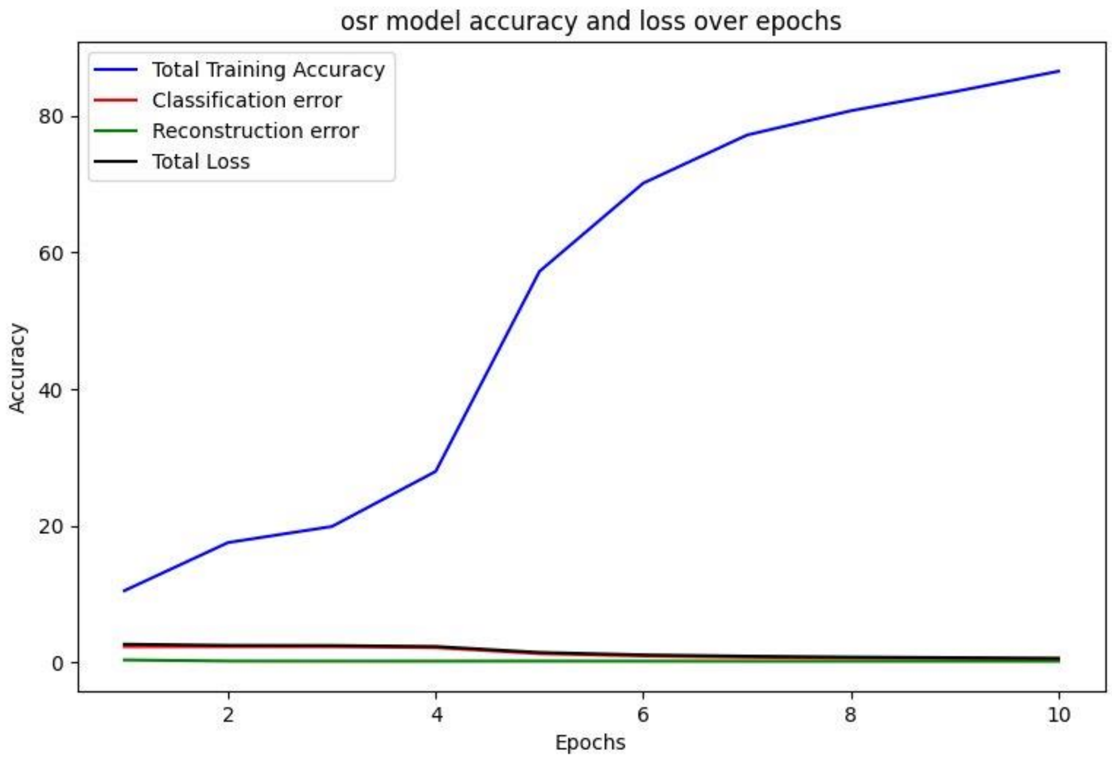
Training the model:

To train the model, we used the MNIST dataset, running it through the network and calculating the appropriate losses for both the logits and latent representations. For the logits, the loss

function was the standard Cross-Entropy Loss, computed from the true labels and the predicted logits output by the network. For the latent representations, we employed Mean Squared Error (MSE) loss, comparing the reconstructed images to the original input images to gauge reconstruction accuracy.
To enhance the network's robustness, we applied a series of data transformations. Specifically, we utilized random application of rotations, Gaussian blur, and resize-crop operations. These transformations were designed to increase the model's ability to generalize by exposing it to a wider variety of input variations.

```
transforms.RandomApply([
    transforms.RandomRotation(degrees=15),  # Randomly rotate
images
    transforms.GaussianBlur(kernel_size=(5, 9), sigma=(0.1,
5)),  # Apply Gaussian blur
    transforms.RandomResizedCrop(size=(28, 28), scale=(0.8,
1.0)),  # Randomly resize and crop images
], p=0.5),  # Apply the augmentations with a probability of 0.5
```

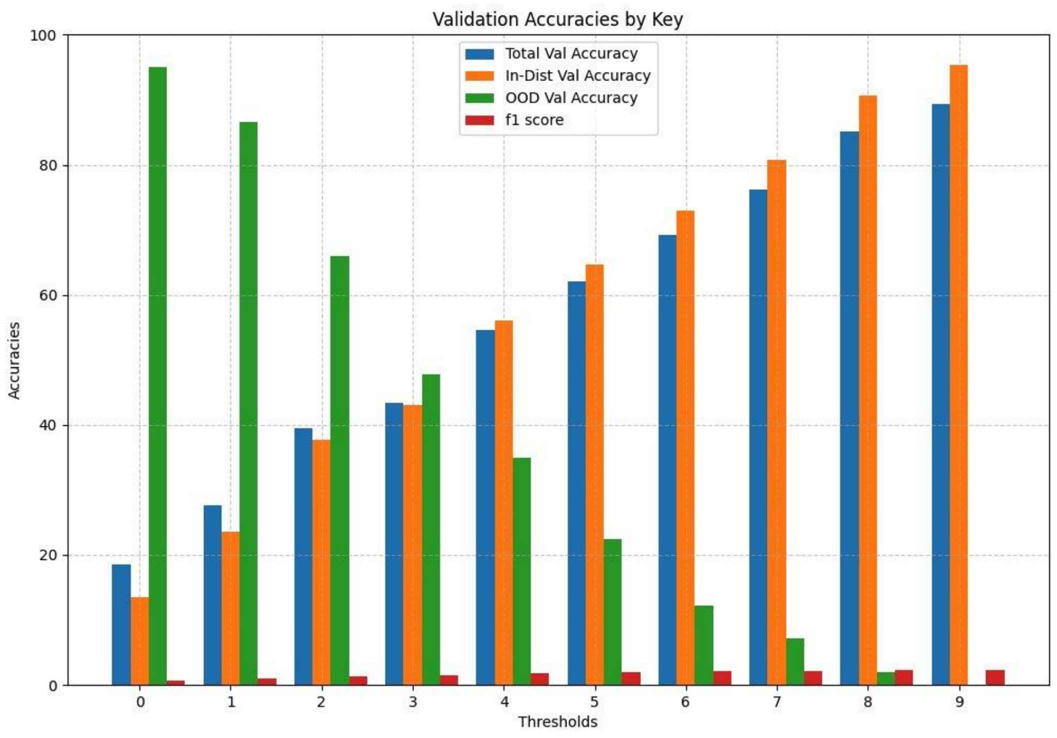
here is the loss over epochs for the network:



Methods of classification of OOD:

class means and thresholding:

Our initial strategy involved calculating the mean latent representations of correctly classified Known Knowns at the end of the training phase and storing these means within the network. When a new unknown sample was introduced, we utilized its latent representation to compute its distance from these class means. An optimal hyperparameter threshold was then applied to determine whether the sample was in-distribution or out-of-distribution. However, this method proved ineffective, as adjustments to the threshold led to almost linear variations in the classification weights between in-distribution and out-of-distribution samples, as depicted in the following image:



Note that the F1 score appears low because it ranges from 0 to 1. This presentation was an oversight on our part, and we opted not to allocate additional computational resources to revise the bar graph, deciding instead to retain it as is, and the threshold values are normalized.

Weibull distribution and and open Max:

After looking more deeply in to the article [2] we opted to utilize a concept that was unknown to us, Weibull disterbution and Open max classifier for open set recognition.

Open max:

This method, as detailed in both articles [1] and [2], generates a new logits vector of size N+1, where N represents the number of known classes. It achieves this by augmenting the original logits vector with belief weights, calculated in a specified manner, to reassess the probability of a new unknown sample being in-distribution, as illustrated in the following image:

$$\begin{aligned} \mathrm{Openmax}_i(\boldsymbol{x}) &= \mathrm{Softmax}_i(\hat{\boldsymbol{y}}), \qquad (2) \\ \hat{\boldsymbol{y}}_i &= \begin{cases} \boldsymbol{y}_i \boldsymbol{w}_i & (i \le N) \\ \sum_{i=1}^{N} \boldsymbol{y}_i (1 - \boldsymbol{w}_i) & (i = N+1), \end{cases} \end{aligned}$$

Here, each wi represents the belief that the current sample belongs to class i of the known classes.

Now the big decision was how to find such wi to create a proper belief state for the distributions classes.

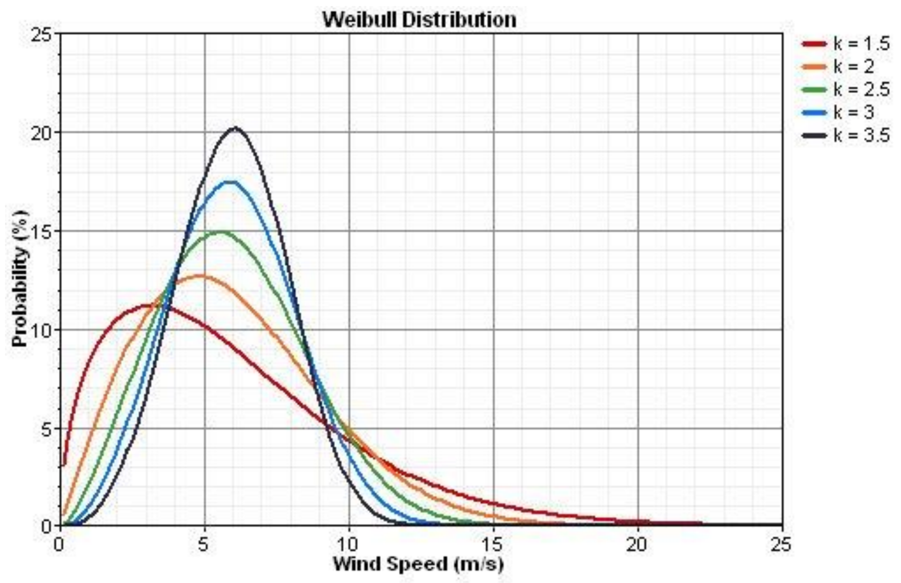
Weibull disterbution:
The Weibull distribution is a flexible probability distribution that models data over a wide range of shapes and scales, commonly used for reliability and life data analysis, and is a big part of the EVT (extreme value theory), which is a branch of statistics dealing with the extreme deviations from the median of probability distributions. EVT is used to model the risk of extreme events in various fields. An example of the usage is shown in the following image:

$$\begin{aligned} p(\boldsymbol{x} \in C_i) &= 1 - R_\alpha(i) \cdot \mathrm{WeibullCDF}(d(\boldsymbol{x}, C_i); \rho_i) \\ &= 1 - R_\alpha(i) \exp\left(- \left(\frac{d(\boldsymbol{x}, C_i)}{\eta_i} \right)^{m_i} \right). \quad (3) \end{aligned}$$

To determine the belief state, we chose to establish a Weibull distribution from the latent representations of the training samples. This involved calculating the class means and measuring the distances from each class's latent representations to its respective class mean. These measurements served as the parameters for constructing the Weibull distribution used in detecting unknown samples.

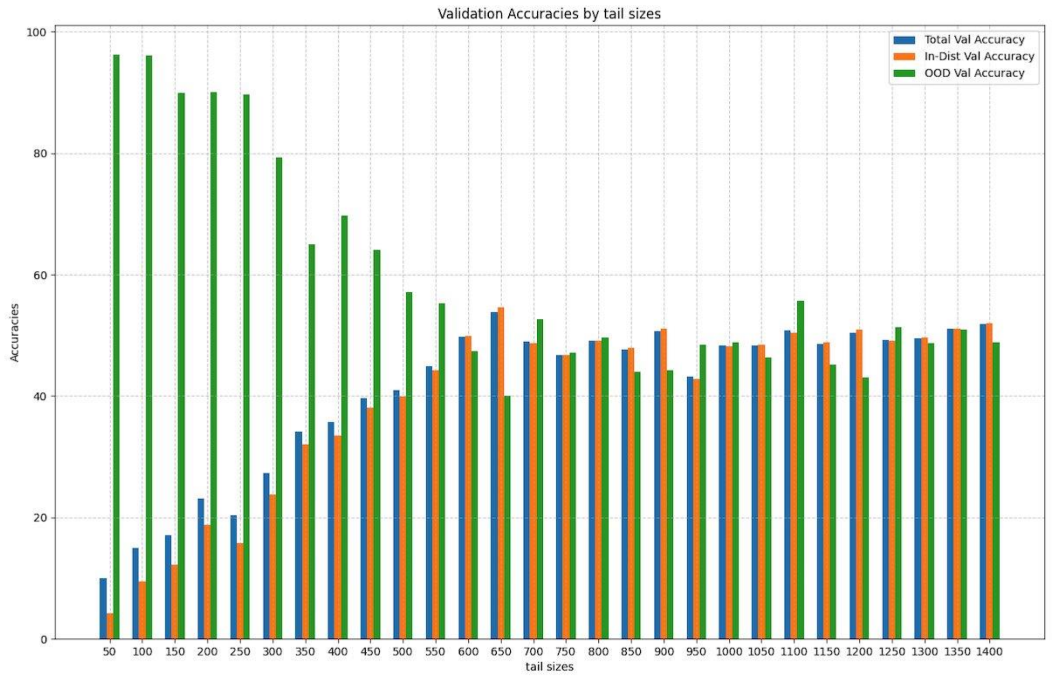
Note we used the python module sclearn to compute the weibull idsterbution.

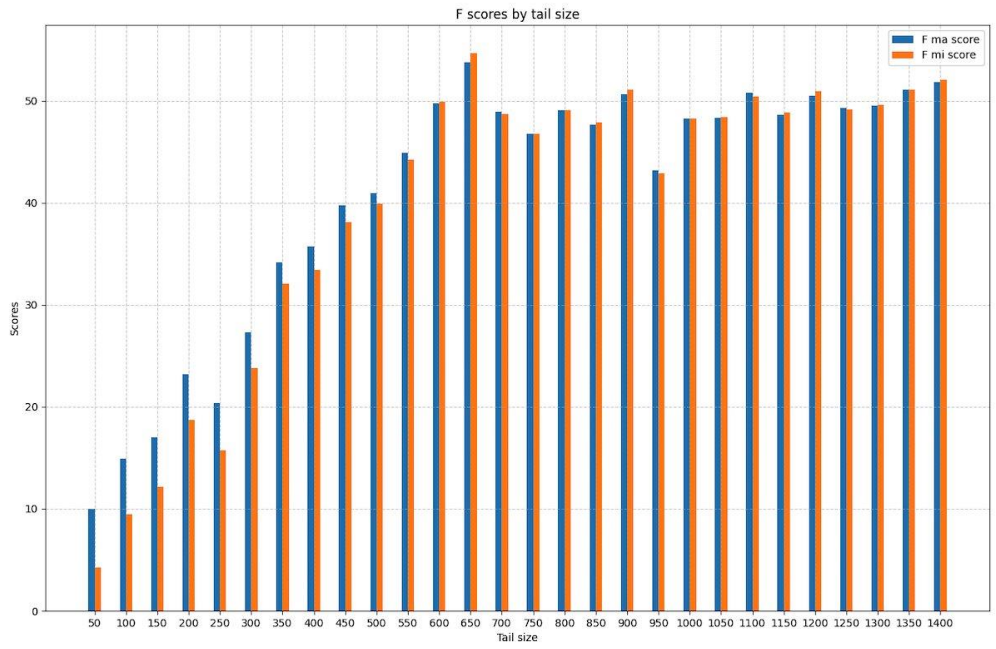
here is a 2D image to represent what it means to fit a weibul distribution to some parameters



Weibull tail size:

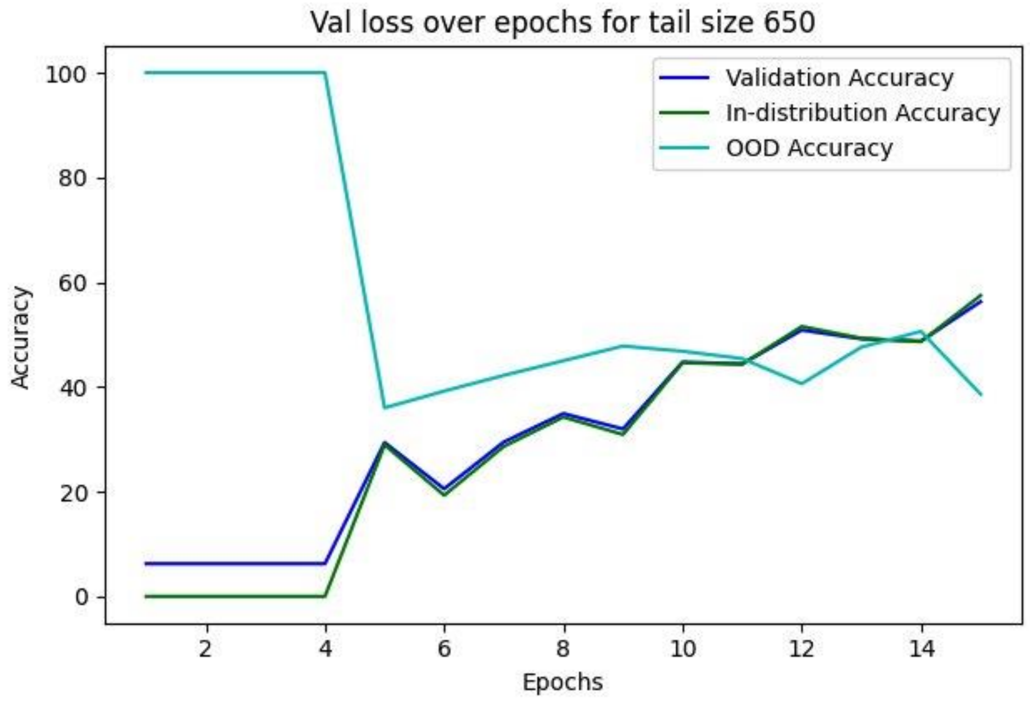
To enhance the resilience of our model and the Weibull distribution to outliers, we computed the Weibull distribution using only the most distant points. This approach makes the distribution more sensitive to outliers and increases its resilience to out-of-distribution (OOD) samples. We sorted the distances of the latent representations and selected a specific tail size of vectors from which to derive the Weibull distribution. The key takeaway is that a smaller tail size simplifies OOD detection but also raises the likelihood of misclassifying an in-distribution sample as OOD, and vice versa. Consequently, we conducted cross-validation on the tail size to optimize this parameter, with the results illustrated in the following image:





By analyzing the results, we observe that the highest F scores are achieved with a tail size of 650. In terms of validation accuracy, although the tail size of 650 results in slightly lower OOD accuracy, it provides the highest in-distribution accuracy. This balance is crucial, as our goal is to preserve in-distribution accuracy while enhancing the model's ability to recognize OOD samples. Consequently, we selected a tail size of 650 as the optimal choice.

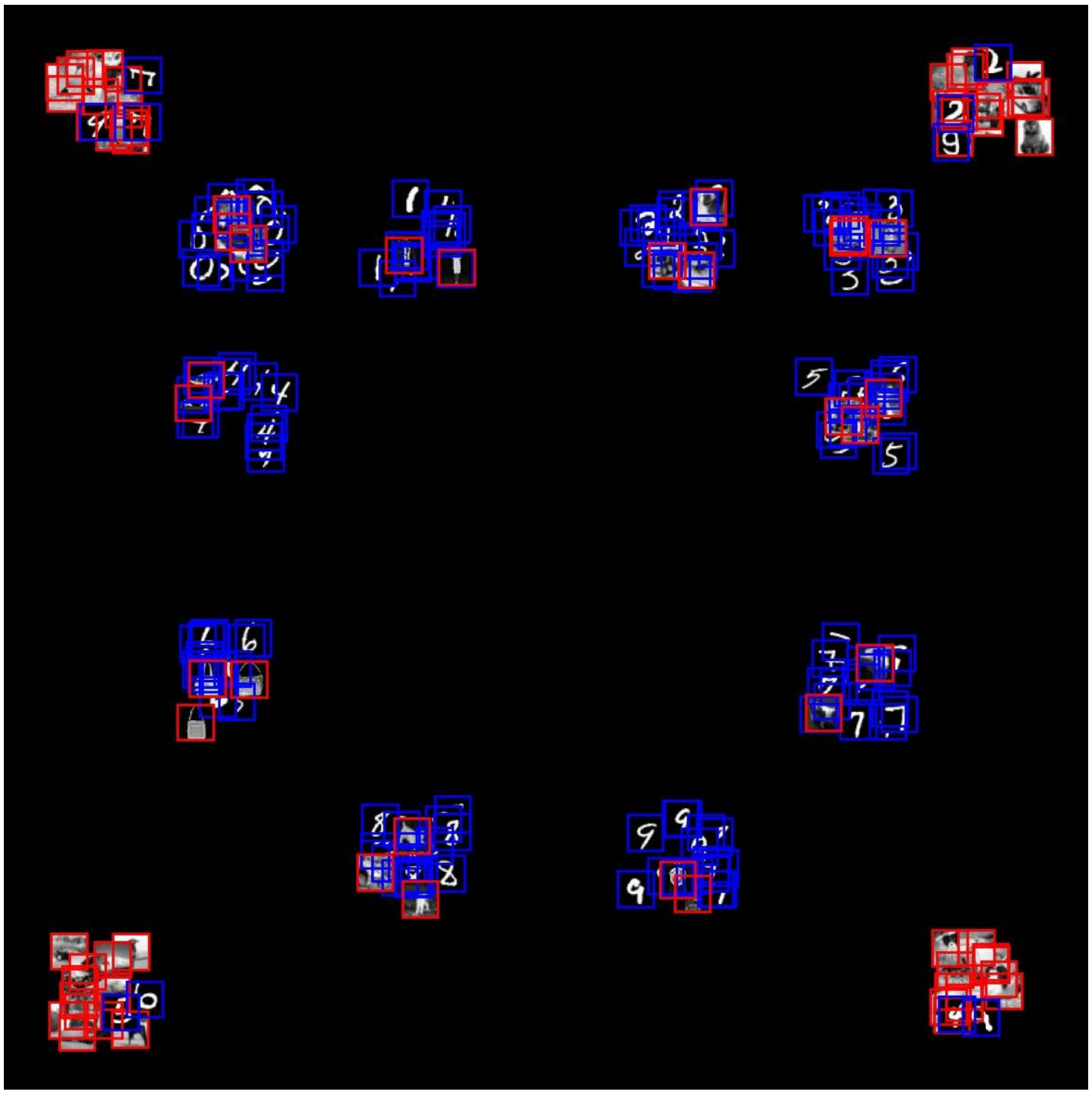
here is the accuracy over epochs with validation for tail size 650:



Note that all plots from the cross validation are saved in the plots folder in the CROSR folder.

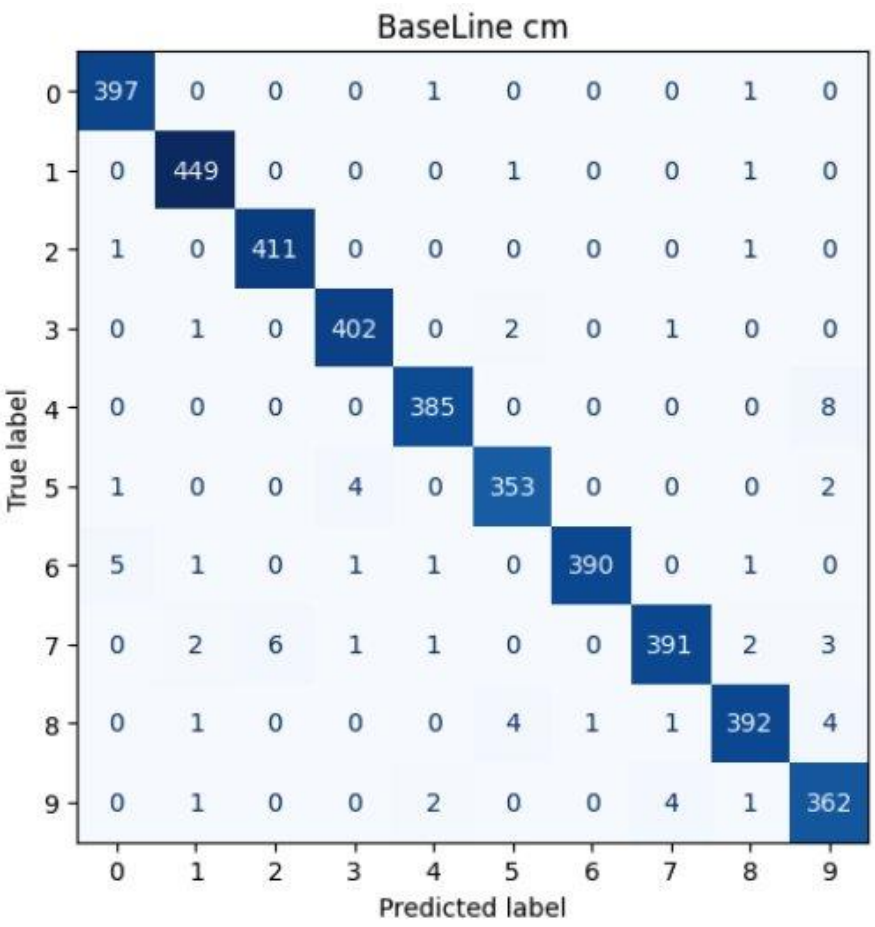
Evaluation and conclusion

First lets show a visual representation of our reslauts with MNIST data having blue frame and ood with red frames:



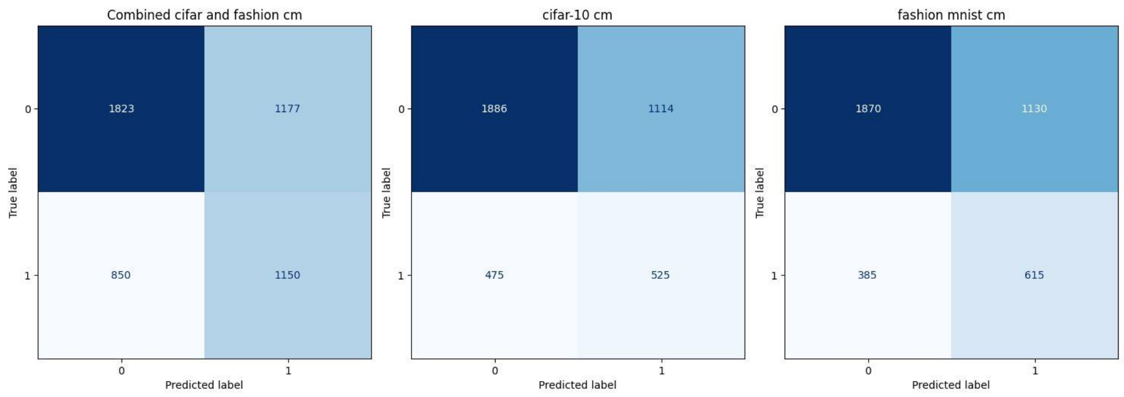
The visualization clearly indicates that the results are suboptimal—indeed, they are far from satisfactory. To better understand the performance quantitatively, we turn to the confusion matrix, which provides a detailed breakdown of the actual results.

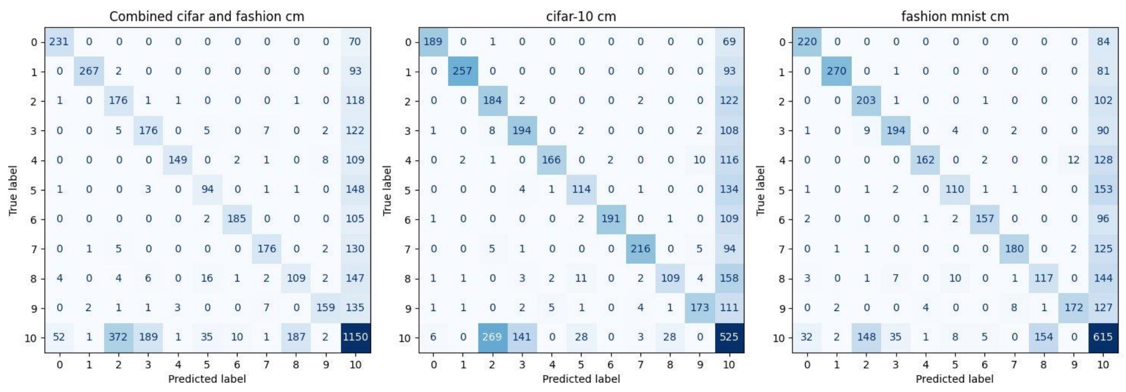
We first examine the Baseline model, which consists solely of the classification layer, excluding the latent representation layer. This model serves as a reference point, allowing us to evaluate the impact of incorporating the latent representation on the overall performance.



The baseline accuracy on the MNIST dataset alone is approximately 98%, indicating that the network performs well for standard classification tasks.

However, when it comes to Open Set classification, the results were less than optimal. To begin our analysis, we will present the confusion matrices, one showing the binary classification of in-distribution samples versus those misclassified as out-of-distribution (OOD):





We observe that, in most cases—whether with FashionMNIST, CIFAR, or their combined dataset—the model tends to classify in-distribution MNIST samples as out-of-distribution (OOD). However, when we examine the full class confusion matrix, it becomes evident that once a sample is correctly identified as in-distribution, the model is quite capable of accurately predicting its true label.

Overall conclusions

The initial outcomes of this study were somewhat disappointing, given the extensive research and cross-validation efforts invested in the subject. However, it is also evident that the existing infrastructure provides a solid foundation, with numerous opportunities for enhancements that could make the framework more robust and increase its accuracy. Despite the potential for improvement, these avenues will not be explored further in this project.
Potential strategies that might enhance the network's performance include:
Increasing the Latent Layer Dimensionality: In this project, the dimensionality of the latent representation was fixed at 32. Future work could explore varying the size of the latent layer, which may improve the network's ability to recognize out-of-distribution (OOD) samples.
Combining Distribution Methods: A hybrid approach combining the Weibull tail distribution with a threshold-based method could be beneficial. By examining the maximum or minimum components of the logits and integrating this information with the Weibull distribution and a threshold parameter, one could gain a deeper understanding of whether a sample is in-distribution or OOD.
In summary, the findings and experiences gained from this project have been insightful, contributing to a deeper understanding of how to approach novel and complex problems using familiar tools.

Articles used:

[1] - Recent Advances in Open Set Recognition: A Survey, Chuanxing Geng, Sheng-Jun Huang and Songcan Chen

[2] - Classification-Reconstruction Learning for Open-Set Recognition, Ryota Yoshihashi1 Shaodi