Ex. 1 Report

After comparing between the following configurations:

* One vs. All – Hamming/ Loss
* All Pairs – Hamming/ Loss
* Random – Hamming/ Loss

We’ve come to conclusion that **All-Pairs** with **Loss Base Decoding** performed the best on the Original MNIST data set.

Following the accuracy comparison between the configurations (on validation):

|  |  |  |
| --- | --- | --- |
|  | Hamming | LBD |
| One vs. All | 86.6% | 87.6% |
| All-Pairs | 91.9% | 92.9% |
| Random | 82.3% | 88.1% |

The decoding function we’ve used, that got the best performance is: Loss Base Decoding. We noticed that it preformed best in all scenarios because it takes into account the magnitude of the predictions rather than just the sign.

Also, we have noticed that using random matrix preforms better than One vs All. We believe that using different scalars rather than just +-1 adds more flexibility to the model allowing it to be a little bit more accurate.

The role of the distance in rows in each code matrix is to decide how far is our current example from each class, because we want to predict the label of the current example. In order to achieve that, we need to come up with a metric how to calculate the distance between two vectors- meaning, the output vector from all the classifiers and a row from the ECOC matrix.

The first metric is Hamming distance, which is a hard-decision metric. Because we   
calculate the actual distance, and only consider whether we are right or wrong.  
Due to that, Hamming distance doesn’t take into account the magnitude of the prediction thus, ignoring the confidence of the model. It’s a binary decision – either right or wrong.

The second approach is Loss Base Decoding which does consider the relevant information which Hamming distance ignores, and therefore is a soft decision metric.   
This approach considers the loss and therefore can tell us how far are we, which adds more flexibility in the process.