Report

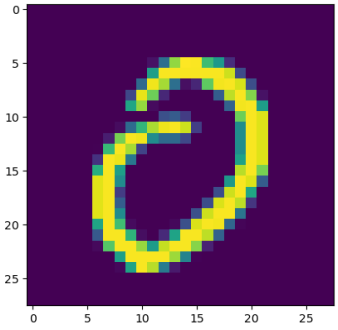
Project Title: Anomaly detection with Autoencoders

by: William Ambrosetti

The report goes over the application of autoencoders for anomaly detection. We are training autoencoders on all the handwritten digits of the MNIST data set, with the exception of digits 4 and 8 (which are considered anomalous).
Anomaly detection is the process of identifying what is normal data such that we can use it to spot abnormal data i.e. anomalous data. This process can be done in various ways, but the goal of this project is to analyze a method using data reconstruction through autoencoders. To create the best anomaly detector, we use grid search searching for which model performs best.
In the final tuning stage, the report follows through on how to set a threshold, for the error score, to consider a sample anomalous or not.

1. Data Selection and Splitting

Digits 4 and 8 are considered anomalous thus they are separated by the rest which will be considered “normal”. Each set will be split in training and testing, with the exception of the rest of the data which will be split in training, evaluating and testing. The anomalous digit 8 is used to tune hyperparameters and establish an optimal threshold value to determine if a sample is anomalous or not. Finally, anomalous samples of the digit 4 together with the additional final test set of “normal” numbers, are reserved for the evaluation of generalization capabilities of the model selected. We used 35444 “normal” samples for training, 8044 “normal” samples for MNIST evaluating, 974 “anomalous” samples for evaluating, 982 “anomalous” samples for final testing and 7088 “normal” samples for final testing.



(1) 0 digit sample form MNIST

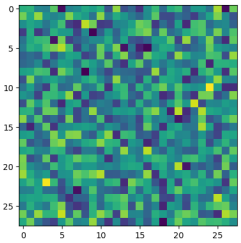
2. Model Architecture

To find which model would be most appropriate for our task, we need to find the best hyperparameters for anomaly detection. Through grid search I have explored: *learning rate, weight decay, max epochs and batch size* for 5 different types of models. Each model has a few variations from the other;

model 0: 6 layers with a latent size of 32 and ReLu and Sigmoid activation functions
model 1: 6 layers with a latent size of 32 and Tanh and Sigmoid activation functions
model 2: 4 layers with a latent size of 64 and ReLu and Sigmoid activation functions
model 3: 4 layers with a latent size of 64 and Tanh and Sigmoid activation functions
model 4: 4 layers with a latent size of 128 and ReLu and Sigmoid activation functions
model 5: 4 layers with a latent size of 128 and Tanh and Sigmoid activation functions

The purpose of the layer variations and latent size variations in each model, is to understand on which occasion the *Manifold Assumption* [ref1] takes place. But in addition we want to know if the activation functions also have a role in anomaly detection.

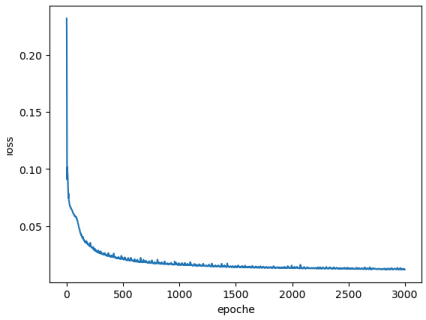
3. Training and Loss



To build the models we will use the Pytorch framework. We can easily create a sequential layer workflow for the data to be compressed to a latent size and reconstructed to a 28 by 28 grid.
Figure (2) shows the output of an untrained model.

(2) untrained model 0

The training process involves minimizing a loss value which describes the reconstruction error for a given sample. This error is the Mean Squared Error (MSE). Essentially we are trying to find the identity function for some given data whilst compressing it into a latent dimension.



Here is an example of model 0 being trained (3) using batches. The loss falls pretty quickly to 0.02 at around 500 epochs and keeps on lowering until around 0.015 towards the end.
Whilst experimenting it was interesting to notice that during model learning, if the parameters were not good enough the loss would be stuck in a plateau. This plateau was simply the combined mean of all of the numbers used.

(3) model 0 being trained. hyper parameters:
(learning rate 0.01, weight decay 0.0,
max epochs 3000, batch size 10000)

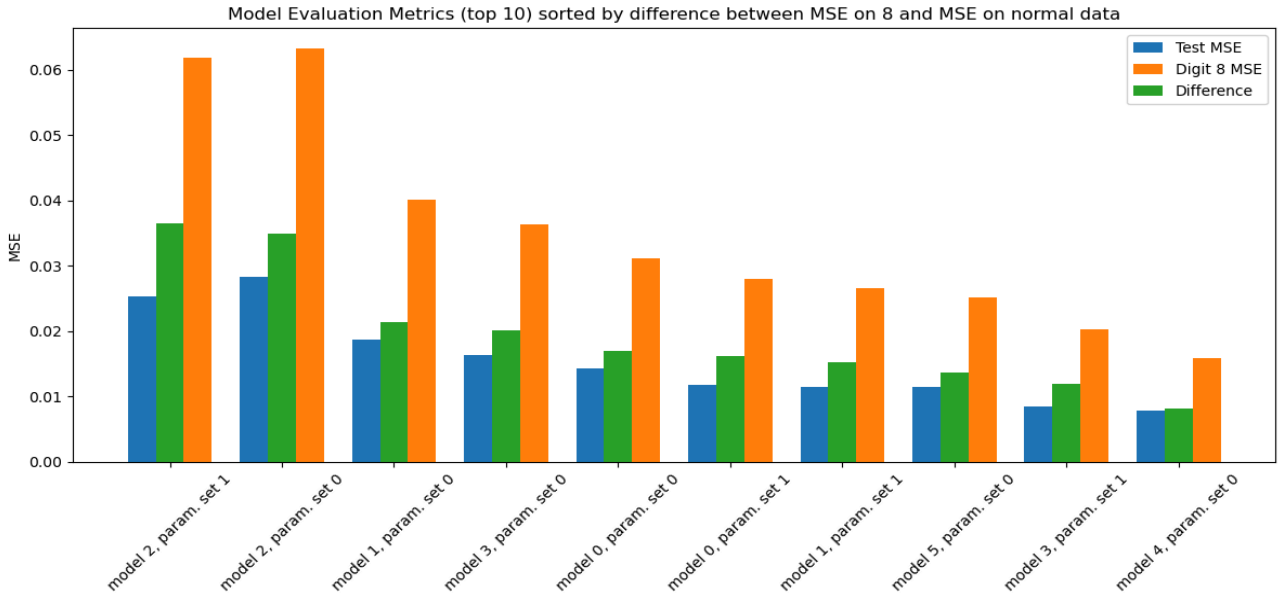


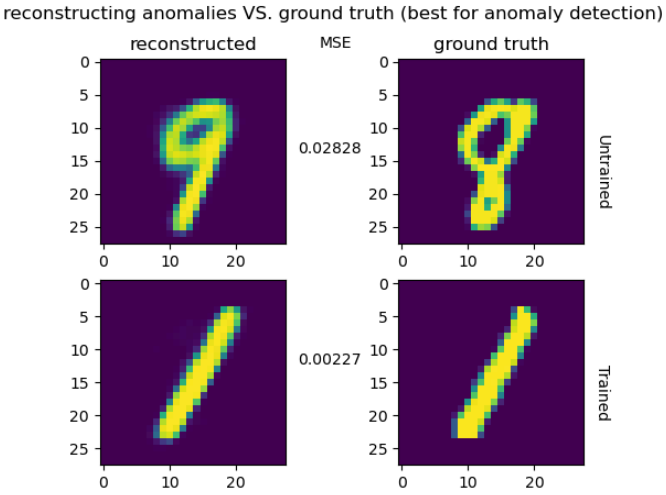
figure (4)

4. Hyperparameter Optimization

Typically when creating an autoencoder it is common to select the best model architecture and parameter set to achieve the lowest MSE. In our case, for anomaly detection, we want to get the best model in determining if a value is anomalous or not, thus we try to find a model with the highest possible error on the 8-data and the lowest possible error on normal data. Hence we look at the difference.

Grid search was used to explore the best combination of hyperparameters and models. Figure (4) has the top models ranked by loss difference between the two sets of data (error on anomalous and non anomalous data). In summary, the biggest difference in error value is due to a combination of factors, like having particular parameters, activations function and correct latent size, and not necessary to one particular factor. But overall model 2 seems to perform best regardless of the parameter set (in this particular instance).

The bar plots in figure (4) show that the best model (model 2) doesn't necessarily have the lowest mean square error. This means that although a model might not reconstruct a number perfectly, it finds the underlying differences between a “normal” and anomalous sample.
If we take a look at the best model, we can notice that its errors are much higher compared to the 10th best model (model 4). Singularly looking at the “Test MSE” error or the “Digit 8 MSE” is not a good indicator whether a model is a good anomaly detector or not.

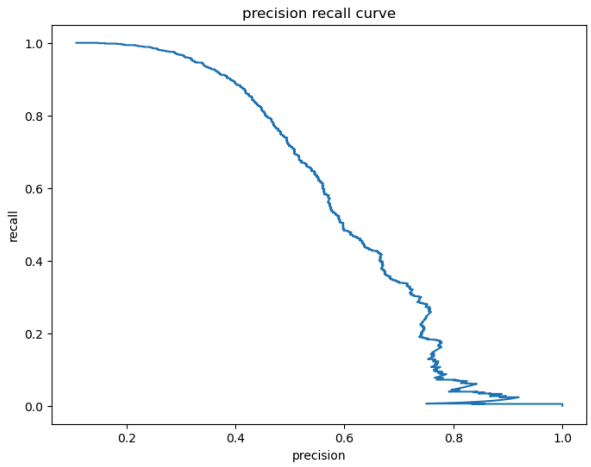


To visualize the difference between a reconstructed number (which the model has been trained for) and an anomalous number, here is illustrated an example (5).

Clearly the reconstruction error is much higher for untrained numbers than numbers which have been seen by the model in the training data. This therefore allows us to distinguish abnormal data from the rest.

(5) reconstructed vs. real with loss values

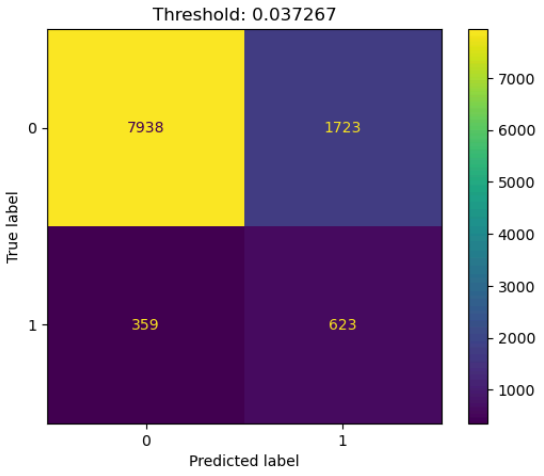
5. Model Selection and Performance Analysis

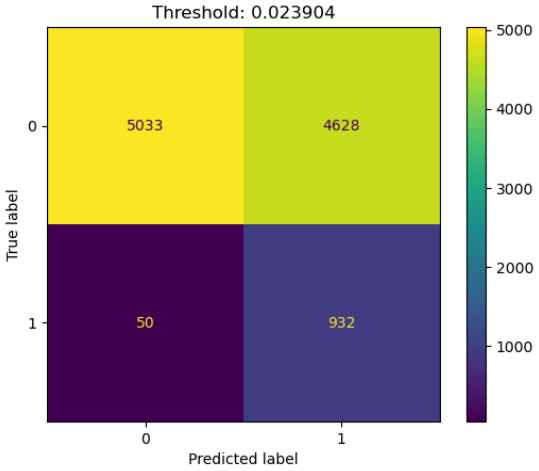


In the final step of the model parameter extraction, we have to determine a threshold to set allowing us to say if an input is anomalous or not. To do that we can look at the precision recall curve and find a balance between the true positive rate and the false positive rate (6). We use a precision recall curve because it accounts for the imbalances of the dataset. The ROC curve might show a better performance than it should.

(6) precision recall curve

Depending on our use case and what prioritization we give to an anomaly, we need to decide the recall and precision levels we are searching for. Unfortunately if we want a high metric we need to sacrifice another. For the final evaluation I am getting the confusion matrices of the best model (found in section 4) predicting on the test anomalous data (4 digit dataset) in combination with the reserved final set of normal data.
By getting the threshold with the closest value of precision and recall to the top right corner (minimizing the distance to the ideal point [1, 1]) we get the best accuracy the model provides (7).
To prioritize recall we can increase the threshold to predict more samples to be anomalous, but it would come at the cost of a lower precision and accuracy. (8)
If we still need a decent precision, it would be better to stick to the first model since it has a higher harmonic mean between precision and recall.





(7) confusion matrix with best accuracy and recall
accuracy: 0.804, recall: 0.634, precision: 0.266
f1-score: 0.374

(8) confusion matrix with higher recall
accuracy: 0.560, recall: 0.949, precision: 0.168
f1-score: 0.285

6. Conclusion

In conclusion, anomaly detection models can be tuned to obtain certain outcomes just due to their architecture and parameter choice. Seen from figure (4) the best models for reconstruction aren't necessarily the best for anomaly detection, so according to its use case, a model can be tuned accordingly. In our example we wanted to have a slightly higher recall whilst still keeping a decent accuracy. But depending on the use case we could have a very high recall sacrificing the other metrics. From this particular task we can determine that autoencoders can be successfully used for anomaly detection, and can provide insight to determine if an anomaly is present or not.

Performance varies slightly with random initialization and data splits; results reported are based on a particular run.

[ref1] Normal data is found on a low-dimensional manifold in the high-dimensional feature space. : This suggests the use of encoder-decoder style methods: encode to a low-dimensional space to force an encoding which extracts the manifold of normal data. : A sample which is reconstructed badly is probably anomalous. (slides) Anomaly Detection - Reconstruction Methods [Michael Wand, Dalle Molle Institute for Artificial Intelligence (IDSIA) USI - SUPSI]