# 2025 Data Mining – HW 2

313553014 廖怡誠

## Explain your implementation which get the best performance in detail.

The model that achieved the best performance in this anomaly detection task was a customized Deep AutoEncoder trained exclusively on the inlier training set. The core idea was to let the model reconstruct only the "normal" patterns seen during training. When faced with outliers in the test set, the reconstruction error—measured by the mean squared error (MSE)—tends to increase, making it a natural anomaly score. Several important design and implementation choices contributed to the strong performance, culminating in an AUC score of 0.99443 on the public leaderboard.

To begin with, the architecture was intentionally designed to be compact yet expressive. The encoder maps the 16-dimensional input features through the following sequence: 16 → 128 → 64 → 64 → 32. This structure was chosen for three reasons:

1. The model isn't overly deep to avoid unnecessary complexity, especially given the relatively small input dimension.
2. The latent dimension of 32 was selected as a trade-off: smaller values like 16 or 8 risk losing too much detail during compression, potentially degrading reconstruction quality and anomaly detection precision.
3. Having two consecutive 64-unit layers introduced a slight increase in model capacity without making the network too prone to overfitting.

To enhance generalization, L2 normalization was applied to the latent vector to constrain the feature embeddings onto a unit hypersphere, encouraging the model to learn more compact and discriminative representations.

In terms of training dynamics, a robust loss function was used instead of the standard MSE:

$$loss(x, \hat{x}) = \log\left(1 + \frac{(x - \hat{x})^2}{0.1}\right)$$

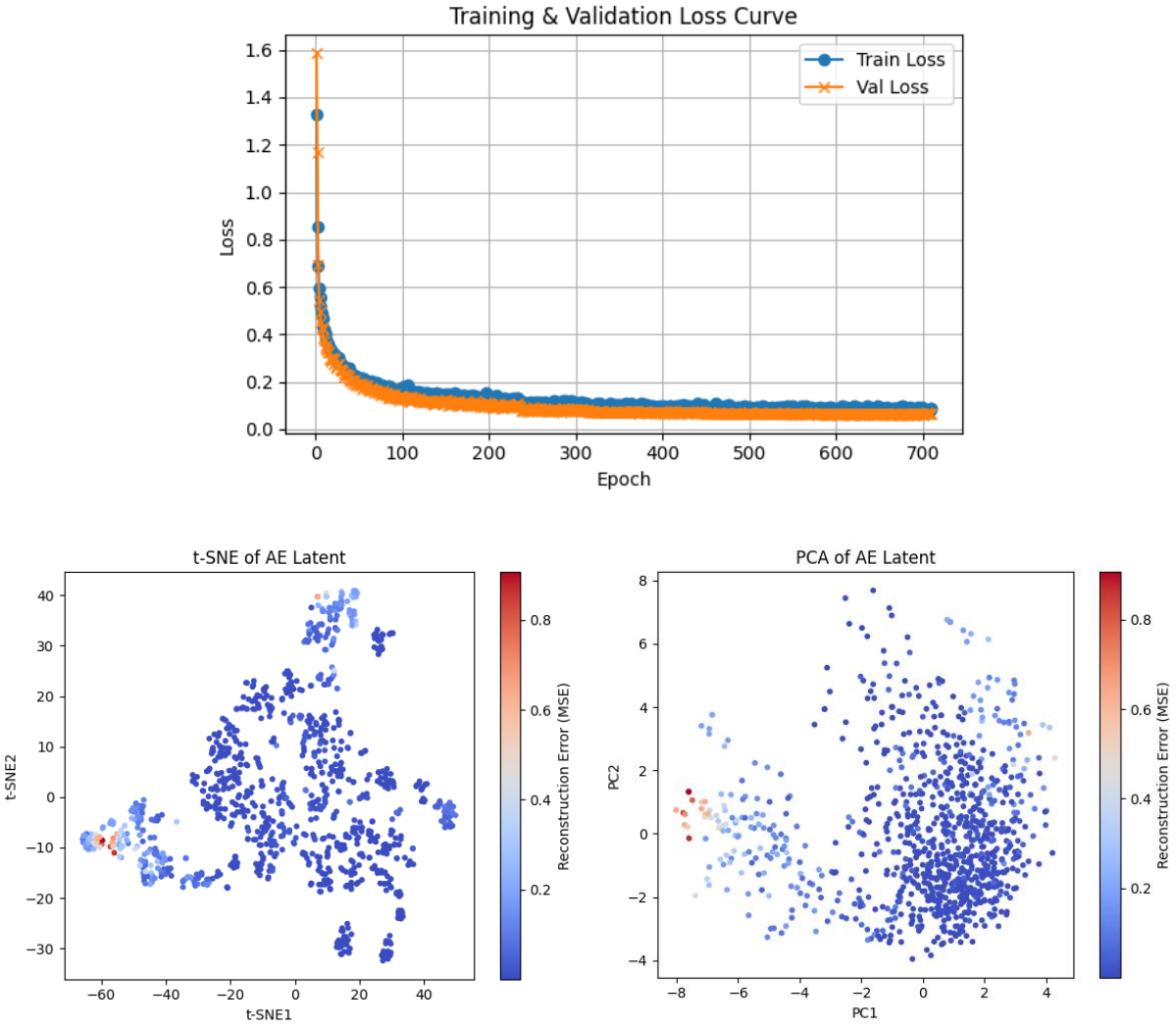
This formulation is more tolerant of occasional large reconstruction errors, which may arise due to the presence of noise or boundary cases during training.

A subtle yet critical decision was the validation split. Instead of the usual 5–20%, I reserved only 1% of the training data for validation. This was because the training dataset was already limited in size (only 6 letters, each with 700 samples), and I wanted to maximize the effective training data to ensure the autoencoder learned the distribution as thoroughly as possible. Despite the minimal

validation set, the model's performance was stable thanks to regularization via batch normalization and the robustness of the loss function.

Additionally, the training process included a learning rate scheduler (ReduceLROnPlateau) and early stopping (with patience set to 100 epochs) to ensure convergence without overfitting. The optimizer was Adam, a widely adopted choice for training deep networks with adaptive learning rates.

Finally, the anomaly scores were computed as the MSE between input and reconstructed output, and the latent features were visualized using PCA and t-SNE to validate separation in embedding space. The overall pipeline effectively identified test-time outliers based on their deviation from learned reconstruction patterns, resulting in a high-quality, stable anomaly detection model.



## Explain the rationale for using auc score instead of F1 score for binary classification in this homework.

This anomaly detection task involves training a model using a dataset containing only inlier samples, and evaluating its performance on a test set that includes both inliers and outliers. Specifically, the training set consists of 4,200 samples drawn evenly from six capital letters: B, E, K, N, X, Z, each contributing 700 instances. In contrast, the test set contains 1,000 unlabeled samples, among which 600 belong to the same six letters (inliers), while the remaining 400 come from four unseen letters and are considered outliers.

Given this setup, the model is not trained to perform binary classification directly. Instead, it learns to reconstruct normal patterns, and then outputs a continuous anomaly score (e.g., reconstruction error) for each test sample. This score reflects how dissimilar a sample is from the learned inlier distribution.

In this context, the AUC (Area Under the ROC Curve) score is a more appropriate evaluation metric than the F1 score, for several key reasons:

**1. AUC measures ranking performance based on continuous outputs**

The model produces real-valued anomaly scores, not categorical predictions. AUC evaluates the model's ability to assign higher scores to outliers than to inliers, which directly aligns with the objective of anomaly detection. In contrast, F1 score requires a hard decision threshold to convert scores into binary labels, and this threshold is not trivial to determine without label information.

**2. AUC is threshold-independent and robust**

Since the true labels of the test set are not available during inference, there is no principled way to choose an optimal classification threshold. AUC evaluates model performance across all possible thresholds, providing a more holistic and robust measure of effectiveness. On the other hand, F1 score only reflects performance at one arbitrary threshold, making it sensitive to threshold selection and less informative in unsupervised settings.

**3. AUC is less affected by class imbalance**

The test set contains an imbalanced distribution: 60% inliers and 40% outliers. AUC is inherently suited to such cases, as it considers the true positive and false positive rates across thresholds. In contrast, F1 score can be significantly skewed by class imbalance, leading to misleading conclusions when one class dominates the other.

**4. Anomaly detection emphasizes ranking over classification**

The fundamental goal of this task is not to classify each point strictly as inlier or outlier, but rather to rank samples by their likelihood of being anomalous. AUC directly measures the quality of this ranking, by assessing whether true outliers are generally assigned higher anomaly scores than

inliers. This property makes AUC particularly well-suited to the problem formulation of this homework.

Overall, AUC provides a more faithful, stable, and informative assessment of model performance in an unsupervised anomaly detection setting. It accommodates the continuous nature of anomaly scores, avoids the pitfalls of threshold dependence, and maintains robustness under class imbalance—factors that make it a better choice than F1 score for evaluating this task.

## Discuss the difference between semi-supervised learning and unsupervised learning.

The distinction between semi-supervised learning and unsupervised learning lies primarily in the availability of labeled data during training and the role labels play in guiding the model's understanding of the data.

To illustrate this difference clearly, we can refer to the below diagram, which visually categorizes machine learning into four main paradigms. The leftmost and middle-left plots correspond to unsupervised and semi-supervised learning, respectively. In the unsupervised learning example, data points are grouped into clusters (e.g., using K-means or hierarchical clustering), but the algorithm has no access to any label information. It learns patterns purely from the feature space, aiming to discover structure or regularities—such as clusters or manifolds—without supervision.

In contrast, semi-supervised learning leverages both a small amount of labeled data and a large quantity of unlabeled data. The plot shows how a small subset of data points (e.g., the yellow points) have labels, and a decision boundary is inferred not only from them but also with the help of unlabeled examples (green points) that inform the overall data distribution. A good semi-supervised model uses the unlabeled data to refine or reshape the decision boundary in a more accurate or smooth manner, often assuming that data points in the same cluster or manifold likely share the same label.

**Example Comparison:**

- **Unsupervised Learning Example**: A customer segmentation task where we have only purchase histories, and no prior knowledge of customer categories. We might use clustering to group customers into behavioral segments (e.g., frequent buyers vs. occasional shoppers).
- **Semi-Supervised Learning Example**: Suppose we are building an email spam classifier. We might have 500 labeled emails (spam or not spam) and 10,000 unlabeled emails. A semi-supervised algorithm would start from the labeled examples, then propagate label information into the unlabeled emails based on similarity or neighborhood structure, resulting in a more powerful classifier than using the labeled data alone.

In the context of this homework, although the autoencoder is trained only on "normal" samples (inliers), and thus might seem similar to semi-supervised learning, it is technically a form of

**unsupervised learning**, because the model is never explicitly given label information about what is normal or anomalous during training. Instead, it relies on the implicit assumption that training data represents the dominant class (normal), and uses reconstruction error to flag deviations (anomalies) during inference.

To summarize:

- **Unsupervised learning** finds structure in fully unlabeled data.
- **Semi-supervised learning** builds better predictors using a small number of labeled samples alongside many unlabeled ones.

The choice between the two approaches depends on the availability of labels and the task at hand. Semi-supervised learning can outperform supervised learning when labels are scarce but unlabeled data is abundant, while unsupervised learning remains a powerful tool when no labels are available at all.