Project MVTec Anomaly Detection

Report 2: Modeling

## Data for Training, Testing and (Cross-)Validation

### Databases: Original and Synthetic

The **original database** of the MVTec Anomaly Detection Dataset consists of images with three dimensions of fixed size (height x width x channels) for 15 categories.   
The fact that models have to be trained, tested and evaluated for each of these categories individually, leads to a comparatively small amount of validation data, if different sets for cross validation have to be split from the available test data.

Therefore, in addition to the approach of splitting the test data into test and validation sets, a new base of **synthetic validation data** has been created by taking normal images and randomly inducing anomalies by twisting small parts of them.

### Feature Extraction: Direct and Deep

For use with the various models, features have been extracted from the image databases in two different ways:

* **directly**, by taking the values for each pixel in each channel, as well as additionally computing various statistical values from the images [describe image features?]
* with **deep feature extraction** using transfer learning by feeding the images into ResNet50, a pre-trained Convolutional Neural Network that processes its input through several layers. [add pictures and description for extraction from different layers]

## Combinations of different Datasets and Features to compare Model Performance

All models have been trained on the same training set, and tested on different combinations of test and validation sets. The following five combinations have been used:

features test/val

1. direct features test -> test/val/val/val(40:30:30:30) (original train/test/val)
2. direct features test + synthetic\_val (original train/test + synthetic validation)
3. ResNet(last) test -> test/val/val/val(40:30:30:30) (original train/test/val)
4. ResNet(last) test + synthetic\_val (original train/test + synthetic validation)
5. ResNet(2+3) test + synthetic\_val (original train/test + synthetic validation)

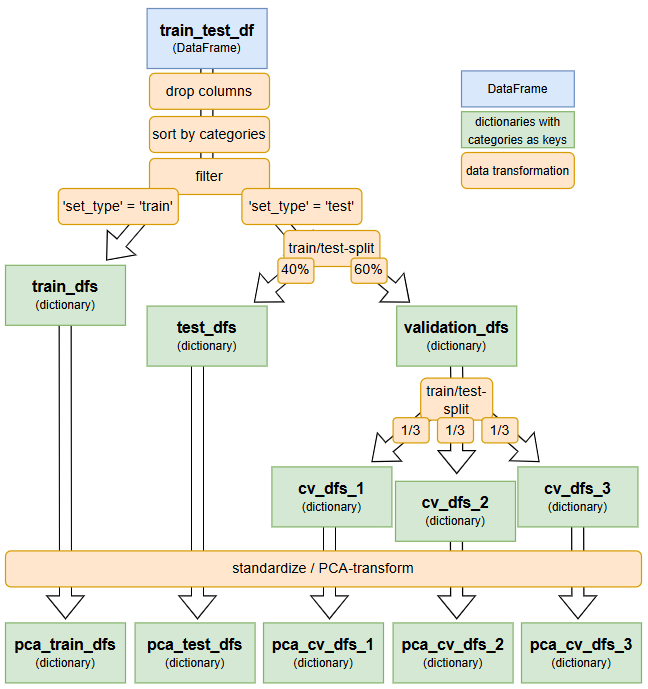
### Splitting: Test- / Validation- Sets

### (original train/test/val)

The initial DataFrame train\_test\_df contains one entry for every single image in the MVTec AD Dataset, containing the directly computed features. Of its columns, features pertaining to bounding boxes are dropped because for several of the categories bounding box computation is imprecise or faulty. Image format features like width and height are dropped because they are constant within any given category. Features pertaining to the subset that an image belongs to are dropped when images are filtered into distinct DataFrames for these subsets.

The following chart shows how data in the train\_test\_df is filtered into different subsets for training, testing and validation of machine learning models.

Each of those subsets is then standardized and transformed by Principal Component Analysis (PCA).



### (original train/test + synthetic validation)

**[???]**

**synthetic: train → 60/40**

## Machine Learning Models

### Multi-model

Several anomaly detection models were evaluated to identify deviations from normal patterns in high-dimensional datasets. The models include One-Class Support Vector Machine (One-Class SVM), Isolation Forest, Local Outlier Factor (LOF), and Elliptic Envelope. Each model employs a different methodology to distinguish normal instances from anomalies.

* **One-Class SVM:** This model maps input data into a high-dimensional feature space using a kernel function and identifies a hyperplane that separates normal data from anomalies.
* **Isolation Forest:** It randomly selects features and splits data points into smaller subsets, isolating outliers more efficiently than normal points.
* **Local Outlier Factor (LOF):** This model measures the local density deviation of a given data point compared to its neighbors, identifying instances with significantly lower densities as anomalies.
* **Elliptic Envelope:** Assumes data follows a Gaussian distribution and detects outliers based on Mahalanobis distance.

#### Hyperparameter Grid

The models were evaluated using the following hyperparameter grids:

* **One-Class SVM**

| Hyperparameter | Values | Explanation |
| --- | --- | --- |
| nu | 0.001 | An upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. |
| kernel | ‘rbf’, ‘linear’, ‘poly’ | Specifies the kernel type to be used in the algorithm. |
| gamma | ‘scale’, ‘auto’ | Kernel coefficient for ‘rbf’, ‘poly’, and ‘sigmoid’. |
| degree | 2, 3 | Degree of the polynomial kernel function (‘poly’). |

* **Isolation Forest**

| Hyperparameter | Values | Explanation |
| --- | --- | --- |
| n\_estimators | 100, 200, 300, 400 | The number of base estimators in the ensemble. |
| contamination | 0.01 | The proportion of outliers in the data set. |
| max\_samples | ‘auto’, 1.0 | The number of samples to draw from the dataset to train each base estimator. |
| max\_features | 0.9, 1.0 | The number of features to draw from the dataset to train each base estimator. |

* **Local Outlier Factor (LOF)**

| Hyperparameter | Values | Explanation |
| --- | --- | --- |
| n\_neighbors | 10, 20, 30, 50 | The number of neighbors to use for k-neighbors queries. |
| contamination | 0.01 | The proportion of outliers in the data set. |
| leaf\_size | 20, 30, 50, 100 | The leaf size passed to the BallTree or KDTree. |
| metric | ‘minkowski’, ‘euclidean’ | The distance metric to use for the tree. |

* **Elliptic Envelope**

| Hyperparameter | Values | Explanation |
| --- | --- | --- |
| contamination | 0.01 | The proportion of outliers in the data set. |
| support\_fraction | 0.9, 1.0 | The proportion of points to include in the support of the raw MCD estimate. |
| assume\_centered | True, False | If True, the support of the data is centered. |

#### Evaluation Approaches

The models were evaluated using five different approaches:

**Manual Feature extraction:**

1. **Manual Features - Train Split:** Selected features from CSV, trained on normal train data.
2. **Manual Features - Synthetic Validation:** Selected features from normal data, validated with 75% anomalous synthetic data.

**Resnet50 feature extraction:**

1. **ResNet50 (Last Layer) - Train Split:** Extracted last layer features, trained on normal train data. 2048 features extracted
2. **ResNet50 (Last Layer) - Synthetic Validation:** Extracted last layer features, trained on normal data, validated with 75% anomalous synthetic data. 2048 features extracted. **A diagram of a diagram

   Description automatically generated**
3. **ResNet50 (2nd & 3rd Layers) - Synthetic Validation:** Extracted second- and third- latent layer features, trained on normal data, validated with 75% anomalous synthetic data. 512+1024= 1536 features extracted.A diagram of a diagram

   Description automatically generated

In all cases, a 3-fold cross-validation was applied.

#### Results

1. **Manual Features - Train Split:** Selected features from CSV, trained on normal train data.

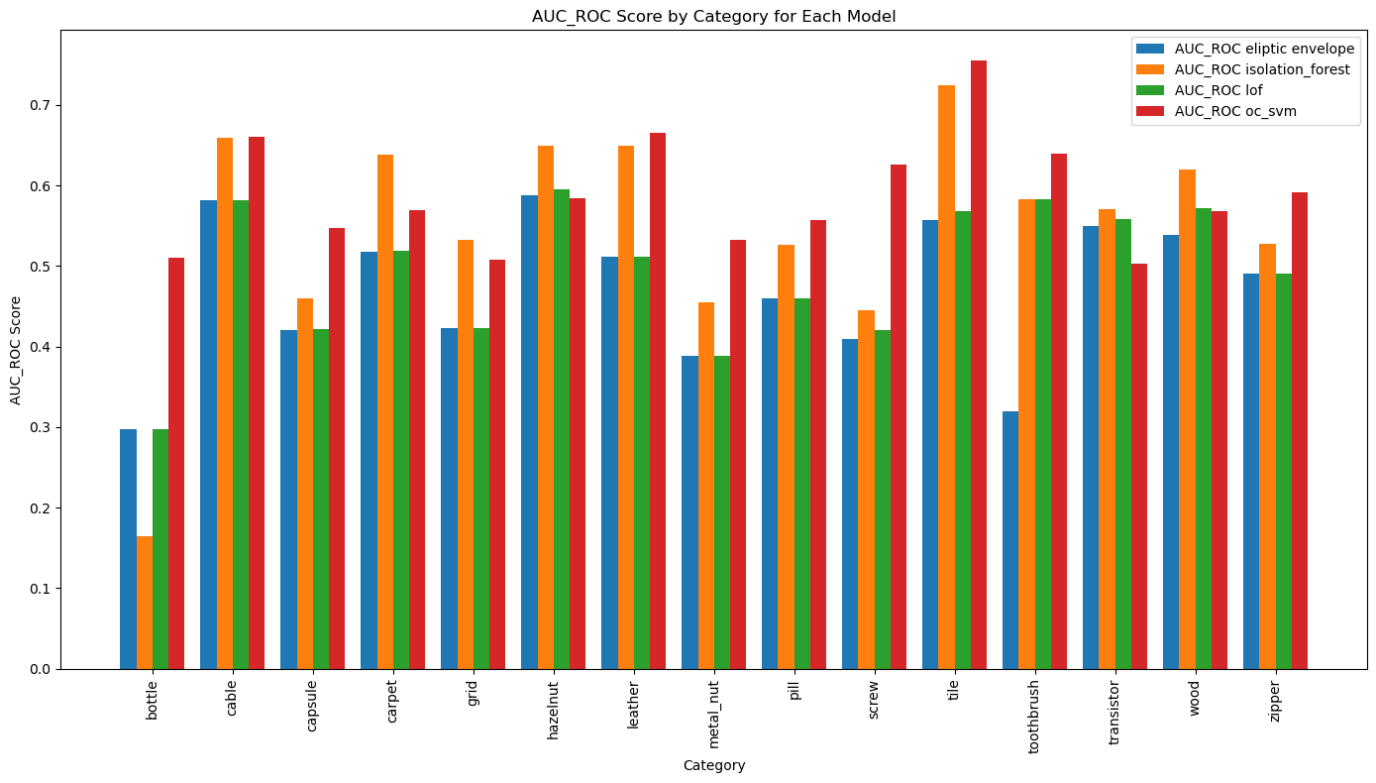
A graph of different colored lines

Description automatically generated

The bar chart represents the **AUC-ROC scores** for different models across various categories. Here are some observations:

* **Overall Performance:**
  + The OC-SVM (red) model appears to perform well across most categories, often scoring higher than other models.
  + The LOF (green) model performs well in some cases but has lower scores in certain categories.
  + Isolation Forest (orange) and Elliptic Envelope (blue) show more variance in performance across different categories.
* **High-Performing Categories:**
  + The "screw" category shows the highest AUC-ROC scores across all models, indicating that all models perform well for this class.
  + The "zipper" and "tile" categories also exhibit strong performance across models.
* **Low-Performing Categories:**
  + The "bottle", "wood", and "grid" categories have relatively low AUC-ROC scores across all models, indicating difficulty in anomaly detection for these objects.
* **Model-Specific Performance:**
  + Elliptic Envelope (blue) generally has lower performance across many categories.
  + Isolation Forest (orange) and LOF (green) show mixed performance, with some high and some low scores.
  + OC-SVM (red) is consistently one of the best-performing models.

1. **Manual Features - Synthetic Validation:** Selected features from normal data, validated with 75% anomalous synthetic data.



* **Overall Performance:**
  + The OC-SVM (red) model remains the most consistent top performer across most categories.
  + Isolation Forest (orange) also performs well but has some categories where it is slightly lower.
  + LOF (green) and Elliptic Envelope (blue) have varying performances across different categories.
* **High-Performing Categories:**
  + The "tile" category has the highest AUC-ROC score, especially for OC-SVM, followed by Isolation Forest.
  + The "cable" and "screw" categories also show strong performance across most models.
  + The "toothbrush" and "transistor" categories have consistently high scores.
* **Low-Performing Categories:**
  + "bottle" has the lowest AUC-ROC scores, with LOF (green) struggling the most.
  + "hazelnut" and "metal\_nut" categories have relatively lower scores compared to others.
* **Model-Specific Performance:**
  + OC-SVM (red) remains the strongest model in almost all categories, making it the best choice for overall anomaly detection.
  + Isolation Forest (orange) performs better in categories like cable, tile, and wood.
  + LOF (green) has more variability, with weaker performance in categories like bottle and hazelnut.
  + Elliptic Envelope (blue) shows stable but lower performance across many categories.

1. **ResNet50 (Last Layer) - Train Split:** Extracted last layer features, trained on normal train data.

A graph of different colored bars

Description automatically generated

* **Overall Performance:**
  + OC-SVM (red) consistently outperforms all models, often achieving AUC-ROC near 1.0.
  + Elliptic Envelope (blue), Isolation Forest (orange), and LOF (green) exhibit lower and more variable scores across categories.
  + OC-SVM shows remarkable consistency, making it the most reliable anomaly detection model.
* **High-Performing Categories (OC-SVM Dominance):**
  + OC-SVM achieves near-perfect scores in: Bottle, Carpet, Grid, Screw, Tile, Zipper
  + These categories indicate OC-SVM's superior detection capabilities.
* **Low-Performing Categories:**
  + Bottle, wood, and metal\_nut have significantly lower AUC-ROC scores across all models except OC-SVM.
  + Elliptic Envelope, Isolation Forest, and LOF perform poorly in these categories, struggling to detect anomalies effectively.
* **Model-Specific Insights:**
  + OC-SVM (red):
    - Best performance across all categories.
    - Consistently achieves the highest AUC-ROC scores.
    - Most reliable model for anomaly detection.
  + Isolation Forest (orange) & LOF (green):
    - Moderate performance, with some categories showing decent AUC-ROC.
    - Not as stable as OC-SVM, but still viable with tuning.
  + Elliptic Envelope (blue):
    - Weakest model overall.
    - Highly inconsistent performance, often the lowest AUC-ROC scores.

1. **ResNet50 (Last Layer) - Synthetic Validation:** Extracted last layer features, trained on normal data, validated with 75% anomalous synthetic data. A graph of different colored bars

   Description automatically generated with medium confidence

* **Overall Performance:**
  + OC-SVM (red) consistently outperforms all models, with AUC-ROC scores close to 1.0 across most categories.
  + Elliptic Envelope (blue), Isolation Forest (orange), and LOF (green) exhibit much lower and more variable scores.
  + OC-SVM's dominance indicates it is the most effective anomaly detection model among those tested.
* **High-Performing Categories (OC-SVM Dominance):**
  + OC-SVM achieves near-perfect scores in: bottle, carpet, grid, screw, tile, zipper.
  + These results suggest that OC-SVM is highly reliable in detecting anomalies in these categories.
* **Low-Performing Categories:**
  + Categories with low AUC-ROC scores for all models except OC-SVM: **bottle, wood, metal\_nut, hazelnut, pill**.
  + Elliptic Envelope, Isolation Forest, and LOF fail to detect anomalies effectively in these categories.
* **Model-Specific Insights:**
  + **OC-SVM (red)**:
    - Best-performing model across all categories.
    - Achieves the highest AUC-ROC scores, making it the most reliable option for anomaly detection.
  + **Isolation Forest (orange) & LOF (green)**:
    - Show moderate performance but are inconsistent across categories.
    - May require additional tuning or feature engineering for improvement.
  + **Elliptic Envelope (blue)**:
    - Weakest model overall, with very low and inconsistent AUC-ROC scores.
    - Struggles in almost all categories, making it the least effective option.

1. **ResNet50 (2nd & 3rd Layers) - Synthetic Validation:** Extracted second- and third- latent layer features, trained on normal data, validated with 75% anomalous synthetic data.

**A graph of different colored bars

Description automatically generated**

* **Overall Performance:**
  + OC-SVM (red) consistently demonstrates superior performance, achieving high AUC-ROC scores across most categories.
  + Isolation Forest (orange) and LOF (green) exhibit moderate and variable performance depending on the category.
  + Elliptic Envelope (blue) tends to underperform in most scenarios, with lower and less consistent scores.
* **High-Performing Categories:**
  + **OC-SVM** achieves the highest scores in categories such as hazelnut, screw, and tile.
  + **Isolation Forest and LOF** also perform well in detecting anomalies in categories like grid, capsule, toothbrush, and transistor.
* **Low-Performing Categories:**
  + Categories with relatively lower AUC-ROC scores across multiple models include bottle, capsule, metal nut, pill, and wood.
  + These categories may benefit from improved feature extraction or alternative anomaly detection methods.
* **Model-Specific Insights:**
  + **OC-SVM (red):**
    - Demonstrates robust performance across nearly all categories.
    - Achieves the highest AUC-ROC scores in most cases, making it the most reliable choice overall.
  + **Isolation Fores**t (orange) & LOF (green):
    - Exhibit good performance in certain categories but show variability.
    - It can serve as alternatives but may require additional fine-tuning for optimal results.
  + **Elliptic Envelope (blue):**
    - The least effective model with highly variable outcomes.
    - Faces difficulties in most categories, rendering it a less dependable option.

#### Overall Summary of AUC-ROC Score Analysis Across Different Feature Extraction Methods

**1. Manual Features - Train Split**

* Selected features from the CSV dataset.
* Trained only on normal train data.
* Performance varies across categories, indicating that handcrafted features alone may not generalize well for all cases.

**2. Manual Features - Synthetic Validation**

* Selected features from normal data.
* Validated using 75% anomalous synthetic data.
* Performance improves slightly but remains inconsistent across distinct categories.
* The introduction of synthetic anomalies provides a broader test environment but may not always align with real-world anomaly distributions.

**3. ResNet50 (Last Layer) - Train Split**

* Extracted features from the last layer of ResNet50.
* Trained on normal train data.
* Shows better performance than manual features, indicating the importance of deep learning-based feature extraction.
* Still has some variability in performance, suggesting potential overfitting to normal training data.

**4. ResNet50 (Last Layer) - Synthetic Validation**

* Extracted last-layer features from ResNet50.
* Trained on normal data and validated with 75% synthetic anomalous data.
* Higher AUC-ROC scores compared to manual features.
* Demonstrates improved anomaly detection, due to richer feature representations learned by ResNet50.

**5. ResNet50 (2nd & 3rd Layers) - Synthetic Validation**

* Extracted features from the second and third latent layers of ResNet50.
* Trained on normal data and validated with 75% synthetic anomalous data.
* Shows the best overall performance among all methods.
* Using features from deeper layers of ResNet50 captures more useful information for anomaly detection.

**Key Takeaways**

* ResNet50-based feature extraction significantly outperforms manual features in anomaly detection.
* Using multiple latent layers (2nd & 3rd) from ResNet50 provides the best results, suggesting that deeper feature representations are more informative for anomaly detection.
* Synthetic validation helps evaluate model robustness but might introduce distributional mismatches compared to real-world anomalies.
* Manual feature selection struggles with generalization, making deep learning-based approaches more effective.

#### Evaluation approach

Based on the analysis, Method 5 (ResNet50 - 2nd & 3rd Layers - Synthetic Validation) showed strong performance in anomaly detection. This method achieved high AUC-ROC scores across various categories, indicating the effectiveness of deep feature representations extracted from multiple latent layers.

For clarity and to focus on the effective approach, the AUC-ROC graphs and the confusion matrix for Method 5 will be presented.

##### AUC-ROC Curves: A graph of a graph Description automatically generated with medium confidence

* The AUC-ROC graphs for each category show the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR).
* Higher AUC values indicate better discrimination between normal and anomalous samples.
* Categories such as cable, grid, hazelnut, leather, and metal\_nut show strong AUC scores, indicating effective anomaly detection.
* Some categories, like capsule, carpet, tile, and zipper, have lower AUC scores, suggesting that these classes are more challenging for the model.

##### Threshold Optimization

* To achieve the best classification performance, **we conducted a threshold search** to maximize the **F1-score**. This ensures that our model is optimally balancing precision and recall for detecting anomalies.

To determine the optimal threshold for classifying normal and anomalous data, we analyzed the **score distribution** for both **train (normal) and test (anomalous) samples**.

The image above illustrates an example of this process for the **"cable"** category:

* + **Green histogram**: Represents the distribution of scores for the normal (train) data.
  + **Red histogram**: Represents the distribution of scores for the anomalous (test) data.
  + **Dashed black line**: Represents the **chosen threshold**, which acts as a decision boundary between normal and anomalous samples.

**Threshold Selection Strategy**

* + The threshold is set to **best separate normal and anomalous distributions**, ensuring minimal overlap.
  + **Ideally**, the threshold should:
    - Minimize **false positives** (normal data misclassified as anomalies).
    - Minimize **false negatives** (anomalies misclassified as normal).
    - Maximize the **F1-score**, balancing precision and recall.

A graph of a graph of a graph

Description automatically generated with medium confidence

* The reported AUC-ROC and F1-score values are from the **test dataset**, which typically has slightly lower performance than the validation dataset used for hyperparameter tuning. This is expected, as the test set represents unseen data.A screenshot of a diagram

  Description automatically generated
* The **confusion matrices** provide a direct visualization of model predictions.
* Each matrix includes the **optimized threshold** used for classification, along with the corresponding **AUC-ROC, F1-score, Precision, and Recall** values.
* The **high recall values** in many categories indicate that the model is effective at detecting most anomalies.
* In some cases, precision is lower, suggesting that the model may generate some false positives.

#### Key Takeaways

* ResNet50-based feature extraction (2nd & 3rd layers) is the best-performing method for anomaly detection.
* Test dataset performance is slightly lower than the tuned validation set, which is expected in real-world evaluation.
* AUC-ROC graphs confirm robust performance in several categories, though some remain challenging.
* Confusion matrices highlight the impact of threshold tuning, with an emphasis on balancing recall and precision.

### KNN+Resnet50 Model

To implement KNN for anomaly detection, we utilized deep features extracted from ResNet50’s 2nd and 3rd latent layers (this method is suggested in the paper "DFR: Deep Feature Reconstruction for Unsupervised Anomaly Segmentation").

These features were stored in a memory bank, which allowed for efficient similarity calculations using KNN. This approach follows the similar methodology as the previous models, ensuring consistency in evaluation.

Unlike OC-SVM or Isolation Forest, which are model-based approaches, KNN operates purely on distances.

#### KNN Implementation Details

* **Feature Extraction**:
  + ResNet50’s 2nd and 3rd latent layers were used to extract deep feature representations for each sample.
  + The extracted features were stored in a memory bank, acting as a reference set for normal samples.
* **Distance Metric**:
  + Euclidean distance (L2 norm) was chosen to compute similarity between a test sample and normal samples in the memory bank.
  + The lower the distance, the more similar the test sample is to normal data.
* **Number of Neighbors (k)**:
  + k = 5: Each test sample was compared against its 5 nearest neighbors in the memory bank.
  + The anomaly score was computed based on the average distance to these 5 nearest normal samples.
* **Anomaly Score Computation**:
  + For each test sample, distances to all normal samples in the memory bank were computed.
  + The minimum distance (s\_star) was selected as the anomaly score.
  + A higher score indicates a sample is more anomalous.
* **Threshold Selection**:
  + To achieve the best separation between normal and anomalous data, we performed a threshold search.
  + The threshold was optimized based on the F1-score, balancing precision and recall.

#### Results

A collage of graphs

Description automatically generated

A group of blue squares

Description automatically generated

In this k-NN anomaly detection experiment on MVTec, the method demonstrates consistently strong performance across most product categories. Several classes—like bottle, hazelnut, leather, and tile—achieve near-perfect separation between normal and defective samples, as evidenced by AUC values very close to 1.0. Even for categories such as screw and zipper, which have slightly lower AUCs, the model still achieves notably high F1 scores, indicating a solid balance between precision and recall.

A plausible explanation for k-NN’s effectiveness is that the chosen feature representations create tight clusters for normal samples, while defective ones end up sufficiently distant in feature space. This makes a local distance-based approach very effective. The primary caveat to keep in mind is that k-NN’s performance can be sensitive to the choice of k and the distance metric, and it may become memory-intensive as the dataset grows. Nonetheless, these results show that, under the right conditions, k-NN can serve as an excellent option for industrial anomaly detection tasks.

A blue and black image

Description automatically generated

A close-up of a heat map

Description automatically generated

In the **leather/glue** example (first figure), the method sharply localizes the small glue spot, with a high-intensity region on the heatmap aligning closely to the ground-truth mask. Meanwhile, in the **bottle/broken\_large** example (second figure), the algorithm similarly highlights the large broken area in red/yellow, assigning a clearly elevated anomaly score (1.7287) and producing a segmentation that closely corresponds to the ground-truth defect region. Taken together, these examples illustrate robust anomaly localization for both subtle (glue) and more structurally significant (large break) anomalies.

## Deep Learning Models

A Convolutional Neural Network (CNN) is a type of deep learning model designed to process spatially structured data, such as images. CNNs use convolutional layers to extract hierarchical features from input data, enabling them to detect patterns such as edges, textures, and objects. Autoencoders, on the other hand, are neural networks designed for unsupervised learning, primarily used for dimensionality reduction, feature learning, and anomaly detection.

A CNN-based autoencoder consists of an encoder and a decoder. The encoder compresses the input into a lower-dimensional latent representation by extracting essential features. The decoder then reconstructs the input from this latent space, ensuring that the meaningful information is retained while discarding noise or redundant data. This structure makes CNN autoencoders highly useful in applications such as image denoising, feature extraction, and anomaly detection.

### Autoencoder + Custom Resnet3 for Feature Extraction

#### Overview and Motivation

* **Why a Custom Architecture?**  
  When each category only provides about 100–200 original images, large networks (like ResNet-50) are likely to overfit, if we want to train from scratch instead of using transfer learning. By creating a smaller “ResNet-3” style encoder—using only a single residual block—you reduce the risk of overfitting, while still benefiting from the improved gradient flow afforded by residual connections.
* **Use of Augmentation**  
  Even though each category starts with a small number of images, the training set has been inflated to ~1,500 images per category via data augmentation. This provides additional samples and variation needed to train a neural network effectively.

#### Structure and Number of Layers

A screenshot of a computer

Description automatically generated

1. **The Single Residual Block**

* **Residual Connection**  
  The core of any ResNet is a residual path that adds the input to the block’s output. This design addresses issues of vanishing or exploding gradients by making it easier for gradients to backpropagate. A diagram of a diagram

  Description automatically generated
* **Dimensional Matching**  
  When the number of output channels or the stride changes, a 1×1 convolution ensures the input matches the shape required for the final addition step.

1. **Encoder (ResNet-3)**

* **Basic Convolutions**  
  The encoder uses several 3×3 convolutions with stride=2, progressively halving the spatial dimensions.
* **Residual Block**  
  A single block is placed in the middle of the encoder to help capture richer features without increasing the network’s depth too much.
* **Feature Dimension**  
  The final layer maps inputs into a lower-dimensional feature space (e.g., 256 channels), effectively forming the latent representation.

1. **Decoder (Upsampling with Transposed Convolutions)**

* **Transposed Convolutions**  
  The decoder mirrors the encoder by using stride=2 transposed convolutions to gradually restore spatial resolution.
* **Sigmoid Output**  
  A final sigmoid activation step restricts the reconstructed image values to the 0–1 range, suitable for image data.

#### Model Training

1. **Data Splits**

**Train Dataset**: Up to 1,500 augmented images per category.

**Validation Dataset**: A smaller set of the original, non-augmented training images used to compute the mean squared error (MSE) loss for early stopping.

**Test Dataset**: Kept separate; used to assess anomaly detection performance (e.g. via the AUC metric).

1. **Loss and Optimizer**

**Loss Function**: The network minimizes the MSE between the reconstructed output and the original input.

**Cropping**: Both reconstructed and original images are cropped slightly (e.g., by a few pixels on each edge) when computing MSE to reduce boundary effects introduced by transposed convolutions.

**Optimizer**: An adaptive method (e.g., Adam) with a moderate learning rate (like 0.001) helps the model converge without elaborate hyperparameter tuning.

1. **Early Stopping**

**Validation Monitoring**: After each epoch, the model’s performance on the validation set is measured.

**Patience Parameter**: If no improvement occurs for a specified number of epochs, training halts to prevent overfitting.

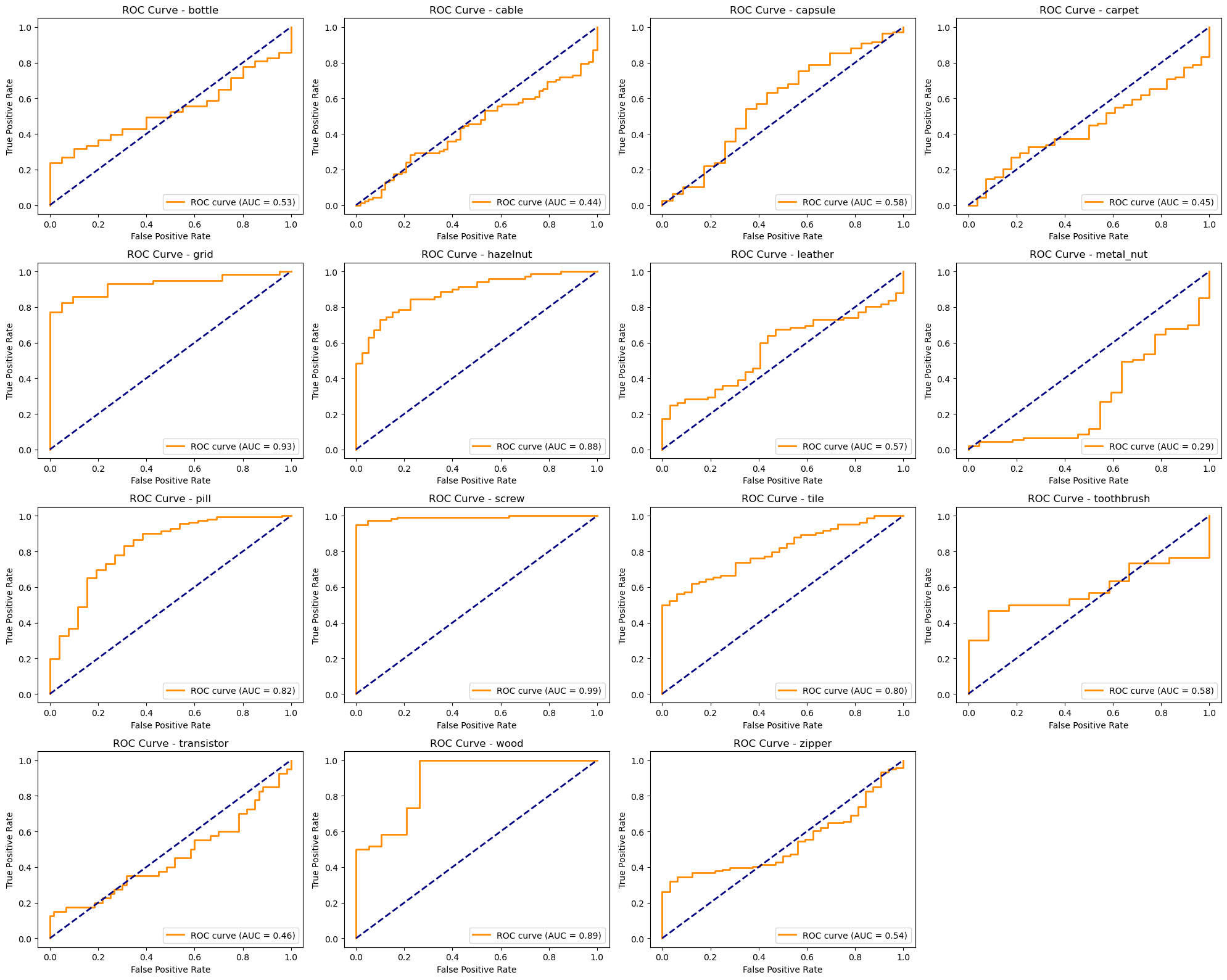
1. **Checkpointing**
   * **Periodic Saving**: The model’s parameters and optimizer states are saved each epoch. If validation performance has improved, the saved checkpoint is labeled as the “best model.”
   * **Resuming Training**: If needed, training can resume from these checkpoints without losing previous progress.

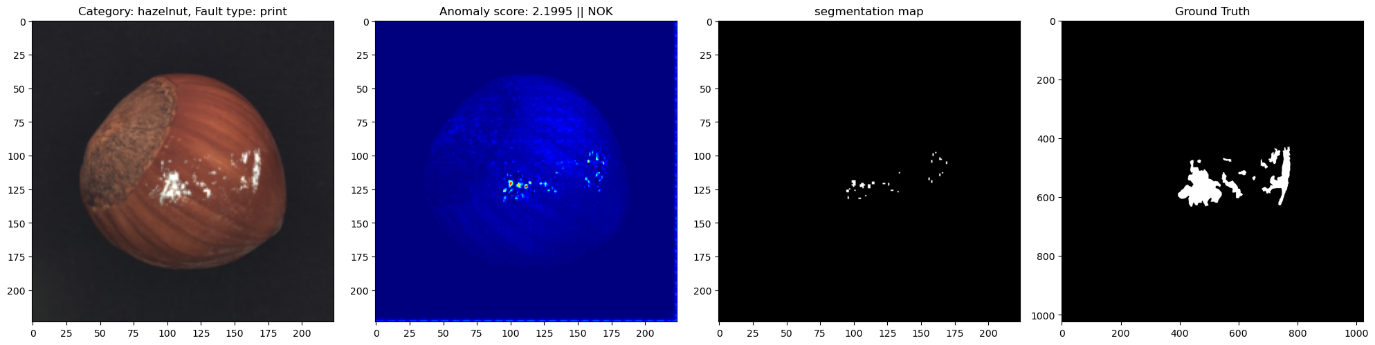
#### Testing and AUC Calculation

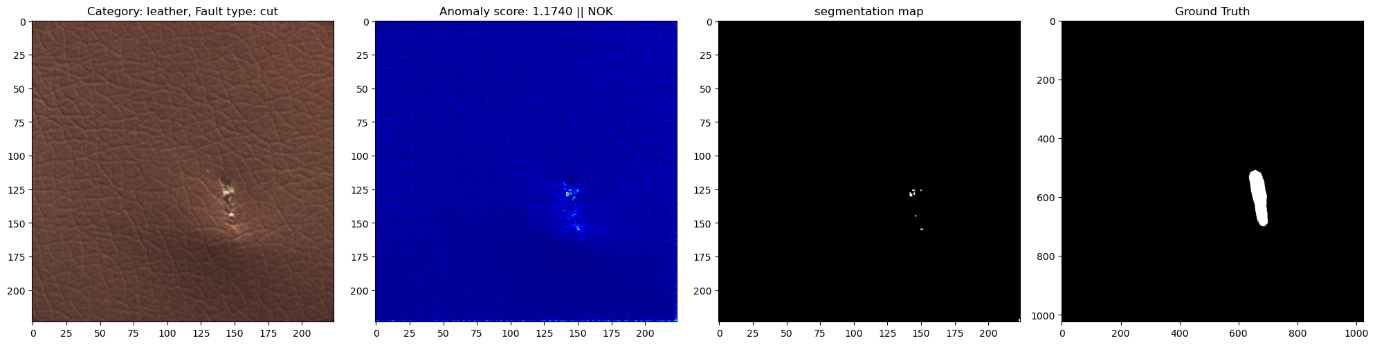
1. **Reconstruction Error as Score**  
   Each test image’s pixel-wise reconstruction error is aggregated into a single anomaly score. Higher reconstruction errors typically indicate more “anomalous” regions.
2. **Thresholding or Ranking**  
   You can compare the reconstruction error to a threshold or compute a ranking-based metric.
3. **AUC**  
   By comparing ground-truth labels (normal vs. anomalous) against the reconstruction error scores, you calculate the Area Under the ROC Curve (AUC). This AUC reflects how well the autoencoder distinguishes between normal and anomalous samples.

#### Results

Overall, the ROC curves show highly variable performance across categories. Some (like **grid**, **hazelnut**, **wood**, and especially **screw**) achieve strong detection capabilities (AUCs above 0.80). Others (such as **metal\_nut** and **cable**) are near or below random chance, indicating that the model struggles to differentiate normal from anomalous samples in those classes.



In the example segmentation maps (e.g., **hazelnut** “print” fault, **leather** “cut” fault), the network identifies the rough location of the anomaly regions, but the highlighted areas often only partially align with the true defects. This partial overlap suggests the model is recognizing some—but not all—distinguishing features of the anomalies.



Despite a few promising results, the model’s overall performance is inconsistent across classes, and it appears to overfit or miss relevant details for certain item types. Ultimately, these results are not satisfactory because the training data is too small to generalize well and capture the full variety of features needed for reliable anomaly detection.

### Autoencoder + ResNet50 (latent layers) for Feature Extraction

#### Feature Extraction using ResNet-50

To extract feature maps from images, a ResNet-50-based feature extractor was employed. ResNet-50 is a deep convolutional neural network known for its ability to learn hierarchical representations through its residual connections, which help mitigate the vanishing gradient problem and enable the training of deeper networks. This architecture has been widely used in computer vision tasks due to its efficiency in capturing both low-level and high-level features.

In this work, feature representations were captured from the second and third layers of ResNet-50. The model was initialized with pre-trained ImageNet weights and set to evaluation mode to prevent updates to the parameters, ensuring that the learned representations remain unchanged. This approach follows the principles of transfer learning, where a model pre-trained on a large dataset is repurposed for a different but related task, leveraging its learned features without requiring full retraining.

To facilitate feature extraction, hooks were registered on the last blocks of layer2 and layer3, allowing access to their intermediate feature maps. These extracted feature maps were then processed using average pooling and adaptive resizing to standardize their spatial dimensions. The resulting feature maps were subsequently concatenated along the channel dimension, ensuring a rich and comprehensive representation of the input images.

A total of 1536 features were obtained, with 512 features extracted from the second layer and 1024 features from the third layer. These feature maps capture different levels of abstraction: earlier layers tend to focus on fundamental image characteristics such as edges and textures, while deeper layers encode more complex semantic information.

The choice of ResNet-50 for feature extraction is motivated by its strong performance on image recognition benchmarks, its efficient residual learning framework, and the availability of pre-trained weights that generalize well across various datasets. By leveraging these pre-trained features through transfer learning, the need for extensive labeled data is reduced, making the model more adaptable to new tasks with minimal fine-tuning.

A diagram of a diagram

Description automatically generated

#### Convolutional Autoencoder (CAE) for Feature Reconstruction

A Convolutional Autoencoder (CAE) was designed to learn feature representations in a lower-dimensional space and reconstruct them. The encoder progressively reduced input feature map dimensions using convolutional layers, while the decoder mirrored this process to restore the input. Batch normalization was applied to stabilize activations, and ReLU activation functions introduced non-linearity for better learning.

Feature maps were extracted from intermediate layers, processed with average pooling and adaptive resizing for consistency, and then concatenated for downstream tasks. The model leveraged transfer learning by initializing with pre-trained ImageNet weights, ensuring efficient feature extraction without full retraining.

* **Convolutional Layers:** Extract key spatial features from input images and progressively reduce dimensions to form compact latent representations. Multiple layers allow hierarchical feature extraction, capturing both fine and coarse details.
* **Batch Normalization:** Normalizes activations within each mini-batch, stabilizing training and reducing internal covariate shifts.
* **ReLU Activation:** Introduces non-linearity, allowing the model to learn complex patterns more effectively while mitigating vanishing gradient issues.
* **Pooling Operations:** Average pooling is used to ensure consistent feature sizes and emphasize dominant patterns by reducing spatial dimensions.
* **Adaptive Resizing:** Ensures feature maps maintain uniform dimensions for concatenation, facilitating better downstream processing.
* **Transfer Learning:** Utilizes pre-trained weights from ImageNet to enhance feature extraction capabilities without requiring large amounts of labeled data.
* **MSE Loss Function:** Measures reconstruction error by calculating the mean squared differences between the original and reconstructed images, guiding model optimization.

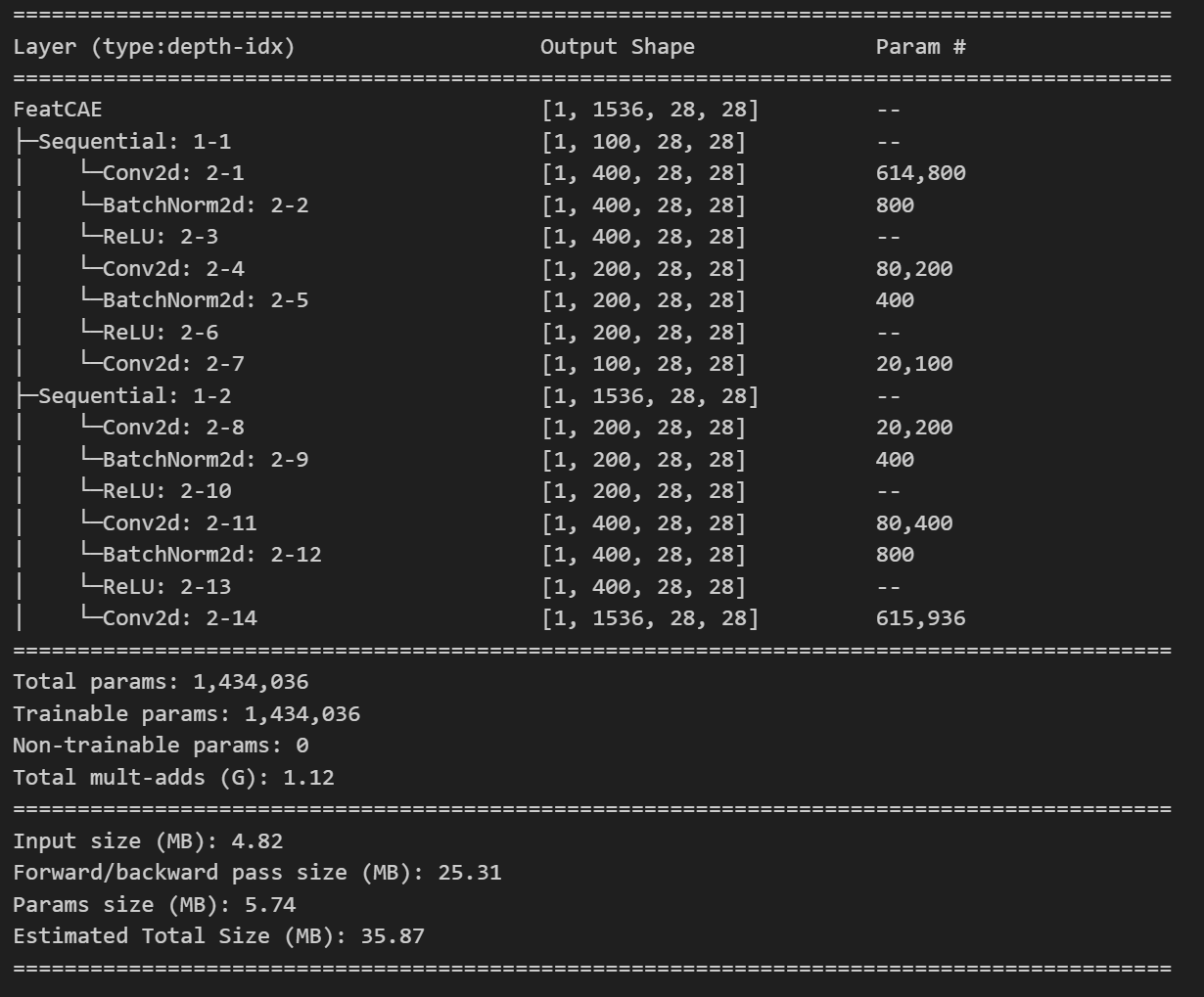
#### Structure and Number of Layers

The CAE was designed with a structured depth to balance feature extraction and reconstruction quality.

* **Encoder:**
  + Uses three convolutional layers to progressively reduce spatial dimensions while increasing the number of feature channels.
  + The first layer captures low-level features like edges and textures, while the following layers extract more complex patterns.
  + Batch normalization is applied after each convolution to maintain numerical stability.
  + ReLU activation ensures non-linearity for better representation learning.
* **Bottleneck:**
  + Represents the compressed latent space where the most important feature representations are stored.
  + This layer has fewer parameters but retains essential information for reconstruction.
* **Decoder:**
  + Mirrors the encoder with three transposed convolutional layers, progressively increasing spatial dimensions to reconstruct the input.
  + The first decoder layer expands basic structures, while deeper layers refine details.
  + Final layer reconstructs the original feature map dimensions with minimized loss.

This structured approach ensures that the CAE learns meaningful representations efficiently while preserving details necessary for reconstruction. The number of layers was chosen to balance computational efficiency and model expressiveness, avoiding excessive complexity while maintaining high-quality feature extraction and generation.

Training aimed to minimize reconstruction error using Mean Squared Error (MSE) loss, optimizing the model to preserve essential image characteristics. This architecture provides a robust foundation for further analysis, classification, or clustering tasks.



#### Model Training

The autoencoder was trained using an Adam optimizer with a learning rate of 0.001. The training process included early stopping to prevent overfitting, where validation loss was monitored over epochs. If no improvement in validation loss was observed for two consecutive epochs, training was halted. The training loss and validation loss were recorded for each epoch to track the model's learning process.

#### Threshold Selection for Anomaly Detection

After training, the reconstruction error was analysed for each category to determine an optimal anomaly detection threshold. The threshold was set based on the mean reconstruction error plus three standard deviations, ensuring that anomalous samples were effectively identified. Histograms of reconstruction errors were plotted, and threshold values were marked to visualize the separation between normal and anomalous data points.

#### Results

A screenshot of a graph

Description automatically generated

The ROC curves presented in the image provide an evaluation of the model’s performance in distinguishing between normal and anomalous samples across different categories. The key observations are:

1. **High AUC Scores:**
   1. Most categories exhibit an Area Under the Curve (AUC) value close to 1.00, indicating excellent performance.
   2. Categories such as *bottle, hazelnut, leather, and tile* have perfect AUC scores of 1.00, suggesting flawless discrimination.
2. **Moderate Performance for Certain Categories:**
   1. The *screw* category has an AUC of 0.75, which is significantly lower than others. This indicates that the model struggles with distinguishing anomalies in screws.
   2. The *capsule* and *pill* categories show slightly lower AUC values (0.94 and 0.92), meaning there is some overlap between false positives and true negatives.
3. **Strong Generalization Across Categories:**
   1. The consistent high AUC scores suggest that the autoencoder-based feature extraction and anomaly detection pipeline generalizes well across various types of industrial components.
   2. This indicates that the chosen feature representation and threshold selection were effective.

A grid of blue squares

Description automatically generated

The confusion matrices in the image provide a detailed evaluation of the model's classification performance for different categories. Key observations include:

1. **High Classification Performance for Most Categories:**
   1. The *bottle, hazelnut, leather,* and *tile* categories exhibit perfect classification with an F1-score of 1.00, meaning there were no misclassifications.
   2. Other categories such as *metal nut, carpet, and transistor* also achieved high F1-scores (>0.95), indicating strong generalization.
2. **Moderate Misclassification in Some Categories:**
   1. The *capsule* category has a relatively higher number of false positives (11) compared to the other categories, impacting its F1-score (0.94).
   2. The *screw* category has the lowest F1-score (0.86), showing more significant misclassification errors, particularly with false negatives.
3. **Class Imbalance Considerations:**
   1. Some categories have a lower number of samples (e.g., *capsule, toothbrush*), which could affect generalization.
   2. Categories with more data points (*pill, screw, wood*) still performed well overall.

**Anomaly Detection Visualization**

A blue and yellow light

Description automatically generated with medium confidence

A blue and black background

Description automatically generated

A comparison of a blue and a red circle

Description automatically generated with medium confidence

A comparison of a blue and a black background

Description automatically generated with medium confidence

A comparison of a color image

Description automatically generated with medium confidence

To evaluate the model's performance, anomaly detection results were visualized using defect detection heatmaps and segmentation maps. Each row in the visualization represents a different defect category and consists of:

1. The original input image showing the defective object.
2. A heatmap indicating the anomaly score, where red areas correspond to higher detected anomalies.
3. A segmentation map highlighting the detected defective regions.
4. The ground truth mask showing the actual defect locations.

**Observations**

* The model successfully localized defects such as **small breaks in bottles, scratches on carpets, holes in hazelnuts, and cuts on leather.**
* The **heatmaps strongly correspond to the defective regions**, confirming the effectiveness of feature extraction and anomaly segmentation.
* Some minor misalignments between the segmentation maps and the ground truth masks suggest areas for potential refinement.

Overall, the anomaly detection pipeline demonstrated **strong localization ability and effective defect segmentation**, making it highly suitable for real-world quality control applications.

### Conclusion

* best Model
* outlook: what could/should be done next?