Project MVTec Anomaly Detection

Report 2: Modeling

### Overview

* **Train- / Test- / Validation-Sets**
  + Split proportions of train/test/validation/crossvalidation
  + synthetic validation data
* **Data Preprocessing**
  + Feature Selection?
  + Standardization / Normalization
  + PCA
  + ResNet for feature extraction
* **Machine Learning Models and results on different datasets**
  + One Class SVM
  + Isolation Forest
  + Local Outlier Factor
  + Elliptic Envelope
  + DBSCAN
* **Deep Learning Models and results on different datasets**
  + Autoencoder
  + Variational Autoencoder
  + CNN
* **Conclusion**

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

**[Mici]**

* which sets have been divided into which subsets, at which proportion?
* creation of synthetic data vor (cross-)validation

### Data Preprocessing

**“Standard” features**

* which columns have been dropped?
* sorting of datasets by categories
* Standardization and Normalization
* PCA

**ResNet Features**

* which Nets
* which Layers

### Machine Learning Models

**Model Description**

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:

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.
3. **ResNet50 (Last Layer) - Train Split:** Extracted last layer features, trained on normal train data.
4. **ResNet50 (Last Layer) - Synthetic Validation:** Extracted last layer features, trained on normal data, validated with 75% anomalous synthetic data.
5. **ResNet50 (2nd & 3rd Layers) - Synthetic Validation:** Extracted second- and third- latent layer features, trained on normal data, validated with 75% anomalous synthetic data.

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.

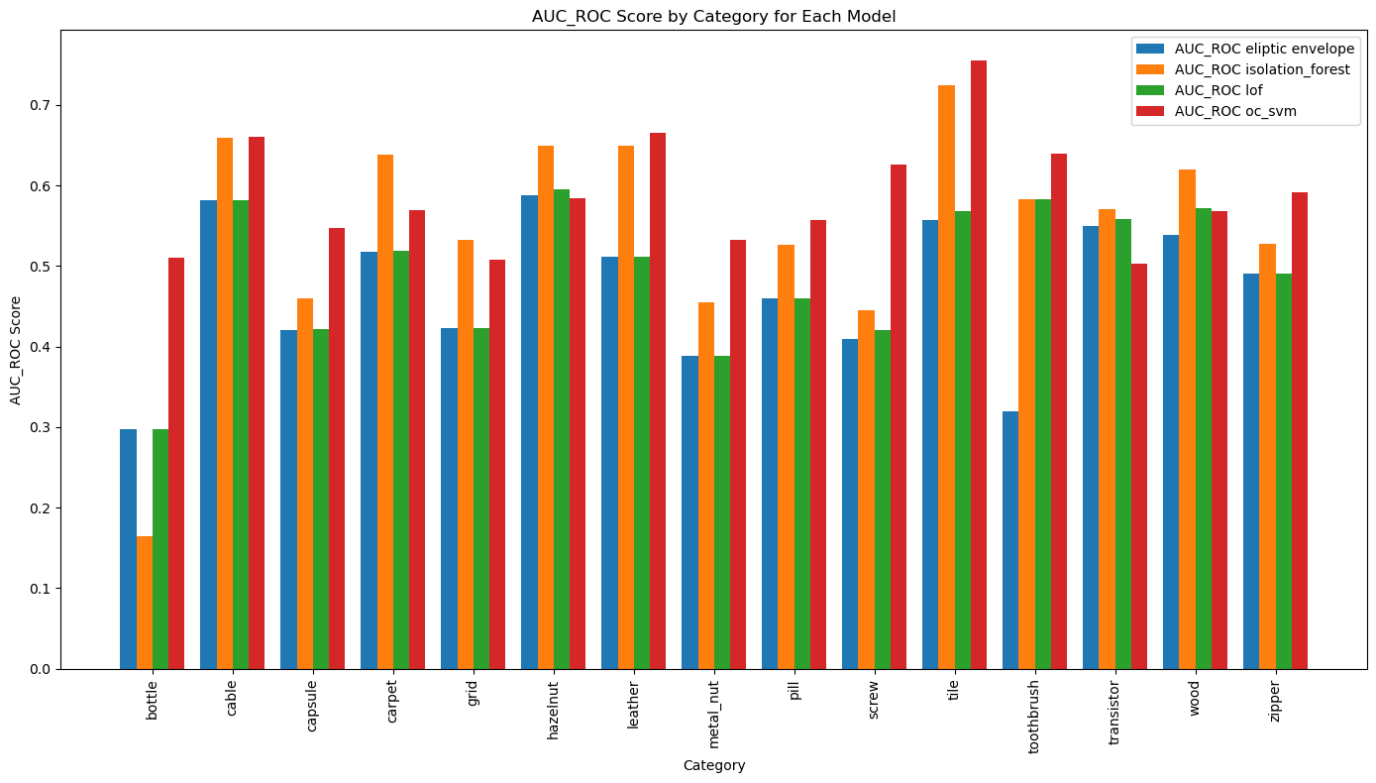
A graph of different colored lines

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.

**DBSCAN [Mici, optional]**

* short model description
* hyperparameter grid
* results on different datasets

### KNN+Resnet50 Model

To implement **KNN for anomaly detection**, we utilized **deep features extracted from ResNet50’s 2nd and 3rd latent layers**. 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.

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 kkk 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 defect localization for both subtle (glue) and more structurally significant (large break) anomalies.

### Deep Learning Models

**Autoencoder + ResNet50 for FE**

* short model description
* hyperparameter grid
* results on different datasets

**Autoencoder + custom CNN for FE**

* short model description
* hyperparameter grid
* results on different datasets

### Conclusion

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