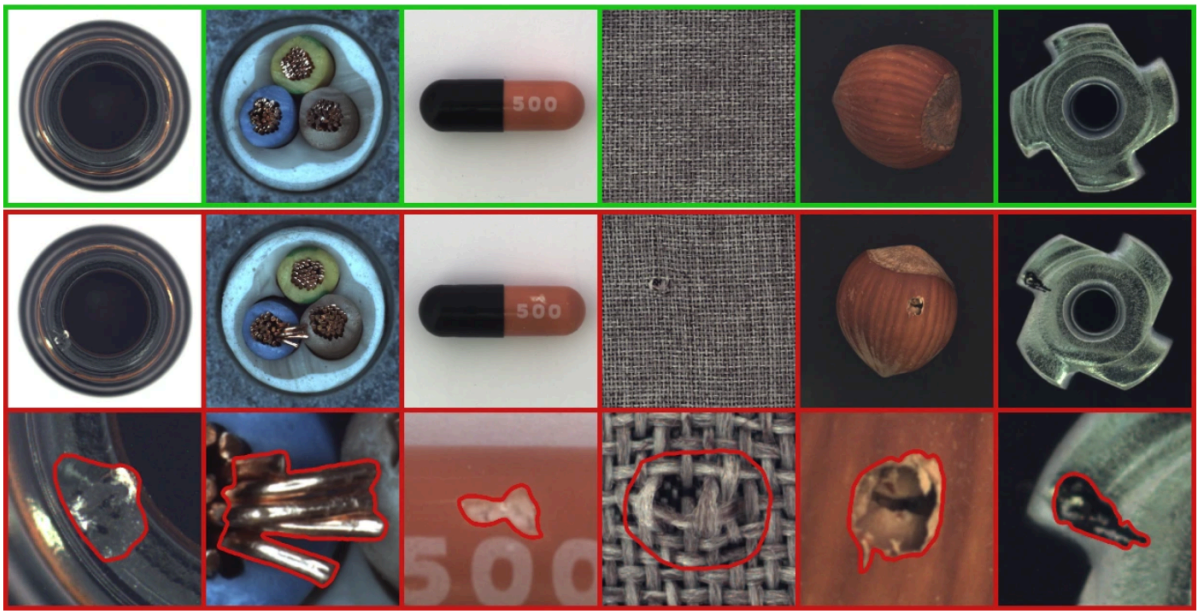
Project MVTec Anomaly Detection Report 3

Table of Contents

Project Description



Ensuring production quality in industrial processes relies heavily on monitoring output to identify defects. An important part of this monitoring consists of visual inspection. While this task is no problem for human observers, it is much more challenging for fully automated systems. As production volumes scale significantly, and human observers can at best inspect sporadic samples, automated systems need to learn to identify defects independently. If such systems can be made to perform their task reliably, they not only solve the problem of scalability, but also eliminate the risk of failure through human exhaustion in a task that is monotonous and repetitive. For this to work, the system has to be able to analyse image data and distinguish between normal or acceptable variations, differences due to e.g. image rotation and actual defects.

This is exactly what anomaly detection in Machine Learning is supposed to do. Various models exist in Machine Learning and Deep Learning that can be trained on available data and learn representations of what is considered ‘normal’. Anything that does not fit this model, will be flagged as an anomaly and considered a defect.

During this project, an image database of industrial components with and without anomalies has been used as a basis to adjust and evaluate several Machine Learning and Deep Learning models to perform the task of anomaly detection. The aim of this project is to separate anomalous images from normal ones. While it would be possible to train the models to differentiate between different types of anomalies, to do this, it would be necessary to previously define the number and types of anomalies to be identified. In order to enable the models to flag any kind of anomaly, even previously unknown ones, the approach will be to train the model on normal images only, and flag those that deviate from this pattern.

MVTec Anomaly Detection Dataset

The available data is a public dataset provided by MVTec, a company specializing in software products and services for machine vision.

- 5,450 high-resolution images across 15 object and texture categories, each a close-up photograph of an object of that specific category of similar size and perspectives
- Training set: Only contains defect-free, ‘normal’ images
- Test set: Includes images with over 70 different types of defects
- Ground truth dataset: Information, where in anomalous images anomalies are located
- Goal: Provide data to test and benchmark anomaly detection approaches, focusing on industrial inspection

Analysis

Preliminary analysis of images by categories indicates that although the size of the anomalous area in images varies significantly between categories (see Fig. 1), scatterplots show that for categories with larger anomalies, as well as such with smaller ones, there are features for which the anomalous images form tighter clusters within the normal ones (see Fig. 2 and Fig. 3). This clustering is a prerequisite for models to be able to learn how to distinguish between classes.

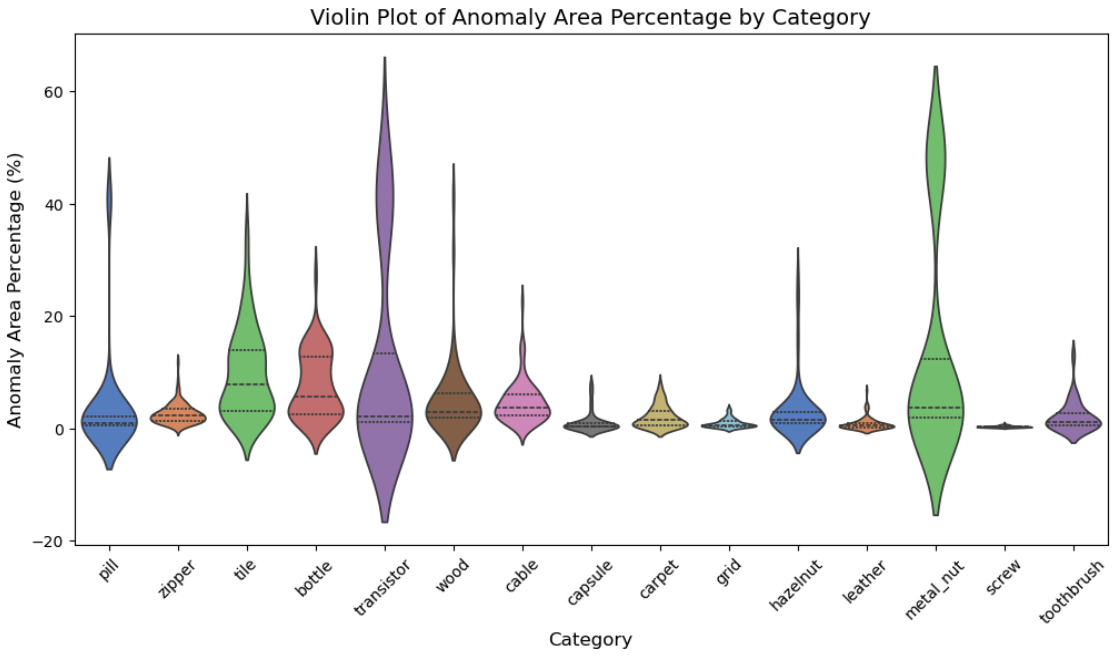


Fig. 1: Violin plot visualizing anomaly area percentages across categories

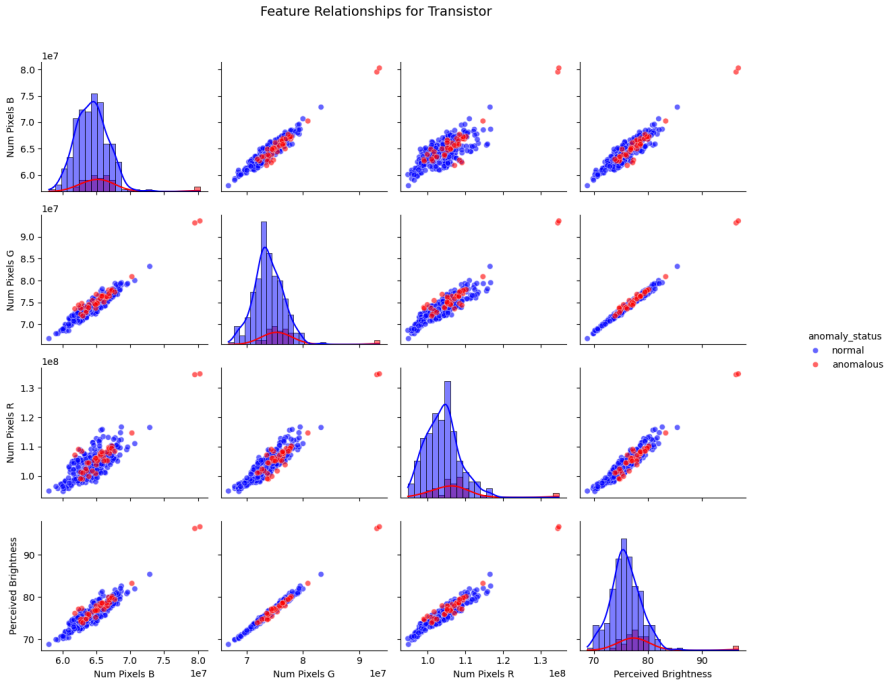


Fig. 2: scatterplots for pixel counts across channels BGR and perceived brightness for category 'transistor'

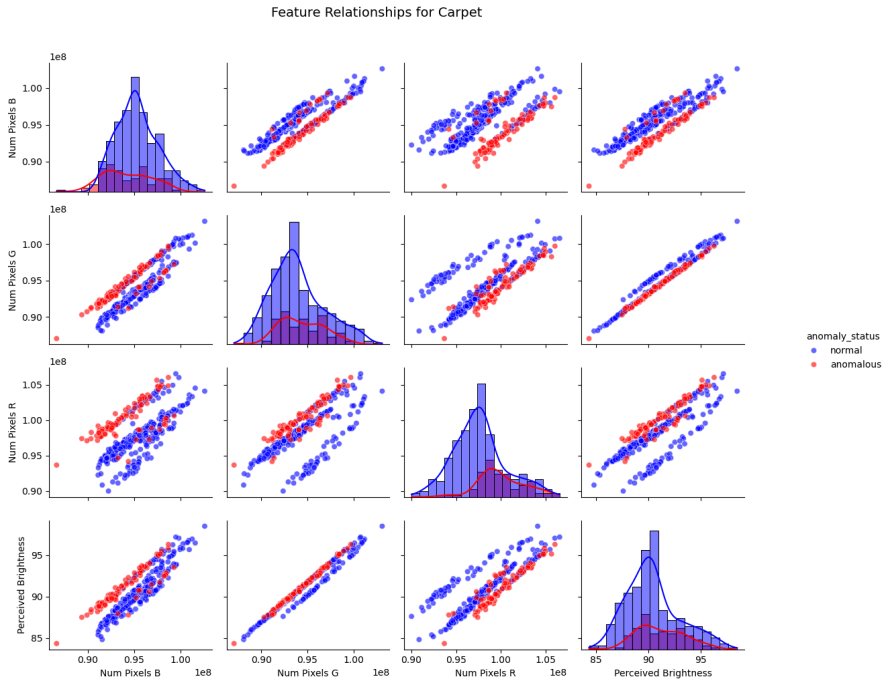


Fig. 3: scatterplots for pixel counts across channels BGR and perceived brightness for category 'carpet'

For more detailed information on dataset structure and analysis results, please see project report 1.

Modelling

Feature Extraction - direct and deep

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

- **Directly**: Various statistical values are computed from the images. Additionally, all images have been resized to an equal size of 224 by 224 pixels in each colour channel and then serialized, interpreting each channel in each pixel directly as one of `224 x 224 x 3 = 150.528` features. For more detailed information, please see project report 2.

- With **deep feature extraction** using transfer learning by feeding the images into ResNet50, a pre-trained net consisting of consecutive blocks of convolutional layers and an output layer that classifies the input. To do this, the convolutional blocks in between extract meaningful features from the input images.

  Instead of using the classification output of the last, fully connected layer, we use outputs from different convolutional blocks as features to feed into our own models.
  The following picture (Fig. 4) shows the ResNet50 architecture as well as the locations after convolutional blocks 1, 2 and 3, where we extract features.

  Two different sets of features are extracted:
  - (a) 2048 features from block 3, or
  - (b) 512 features from block 1 and 1024 features from block 2
    (1536 features in total)

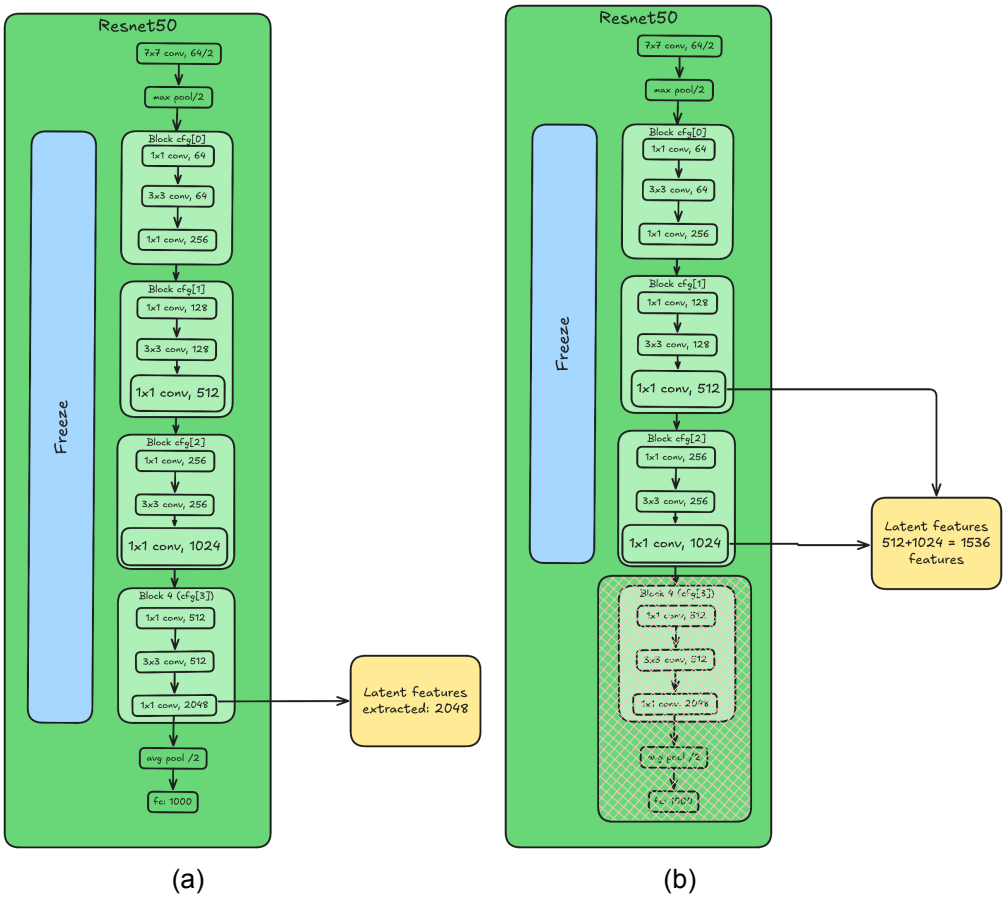


Fig. 4: (a) 2048 features extracted from cfg[3] blocks (b) 1536 features extracted from cfg[1] and cfg[2] blocks

Preprocessing

The data preprocessing pipeline consists of two main steps:

- **Feature standardization** and
- **Dimensionality reduction using PCA**.

First, feature values are standardized using **Min-Max scaling to the range [0,1]**. This ensures that all features have a uniform scale, preventing those with larger numerical ranges from dominating the analysis.
After standardization, **Principal Component Analysis (PCA)** is applied to reduce dimensionality. The number of principal components is determined by retaining **95% of the variance** in the data. This step helps in reducing computational complexity while preserving the most relevant information.
StandardScaler as well as PCA are fitted on the training set, and then used to transform train, test and (cross-)validation sets.

Machine Learning

Baseline Models

Several Machine Learning models for anomaly detection were compared and tested with different parameter configurations. Each model employs a different methodology to distinguish normal instances from anomalies.

- **One-Class SVM:** 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:** Randomly selects features and splits data points into smaller subsets, isolating outliers more efficiently than normal points.
- **Local Outlier Factor (LOF):** 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.

In a first try on the original MVTec AD train and test data, none of these models yielded satisfactory results. To improve model performance, hyperparameter tuning was employed in a grid search.
The fact that the models must be trained, tested and evaluated for each of the 15 categories individually leads to a comparatively small amount of validation data, if different sets for cross validation must be split from the available test data. To remedy that, an alternative approach was added, using part of the original train set and adding synthetic anomalies. This resulted in larger cross-validation sets, for the trade-off of slightly reducing the amount of training data.

Synthetic Anomalies

The synthetic data has been created by taking a percentage of the much larger set of training data of normal images and randomly inducing anomalies into some of them, by twisting small parts of the image.

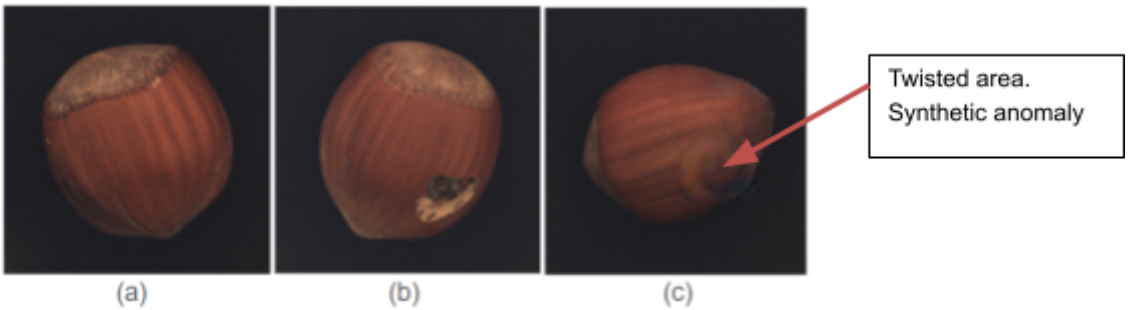


Fig.5: (a) original normal, (b) original anomalous and (c) synthetic anomalous images of category hazelnut

Data Splitting

While the images to train the different models are always taken from the 'train' portion of the original MVTec Dataset, test and validation sets are built in two ways:

- (i) Splitting the original test-set, or
- (ii) Splitting the original train dataset and adding synthetic anomalies to a fixed proportion.

(i) Original MVTec Data

The following chart shows how data from the original MVTec Dataset is sorted by categories and then filtered into different subsets for training and testing.
The **original test data is then split** into 40% remaining test data and 60% validation data, which is further split evenly into three cross validation sets.

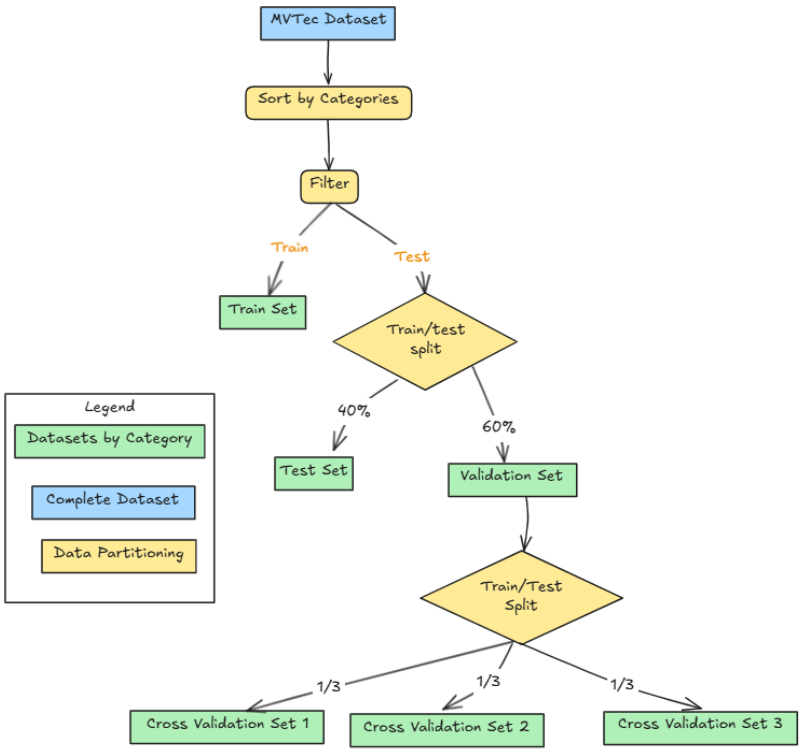


Fig. 6: process of splitting the MVTec dataset into train-, test-, and (cross-)validation-sets, using only original data

(ii) Original MVTec Test Data + Synthetic Validation Data

The following chart shows how data from the original MVTec Dataset is sorted by categories and then filtered into different subsets for training and testing.
The **original training data is then split** into 60% remaining training data and 40% validation data, which is further split evenly into three cross validation sets.

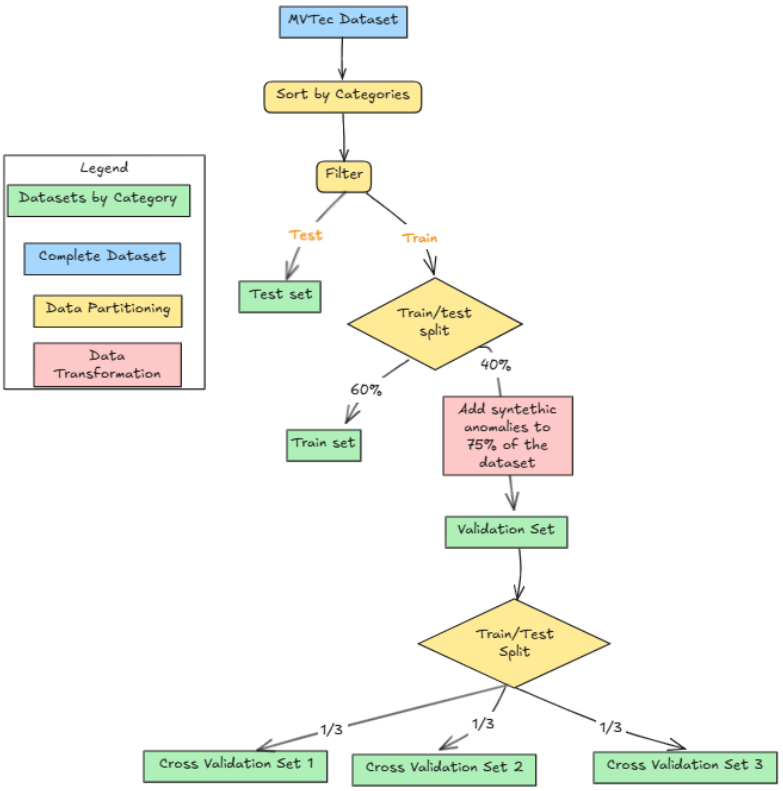


Fig. 7: process of splitting the MVTec dataset into train-, test-, and (cross-)validation-sets, adding synthetic anomalies

Train-, Test- and Validation-Set Combinations

All models are trained on a training set of normal images and tested on different combinations of test and validation sets. In order to obtain these test and validation sets, the above described three feature extraction approaches have been combined with the above described two data splitting approaches to build the following five combinations:

	feature extraction	**test- / validation- set splitting**
1.	direct	original mvtec (i)
2.	direct	original mvtec test + synthetic validation data (ii)
3.	deep (a)	original mvtec (i)
4.	deep (a)	original mvtec test + synthetic validation data (ii)
5.	deep (b)	original mvtec test + synthetic validation data (ii)

Table 1: feature extraction and data splitting methods

Model

Compared to Isolation Forest, LOF and Elliptic Envelope, the **OC-SVM** model consistently showed the best performance over most categories, especially when trained using features extracted by deep feature extraction from cfg[1] and cfg[2] blocks of ResNet50 and validated using synthetic validation data (see Table 1, option 5 and Fig. 8).

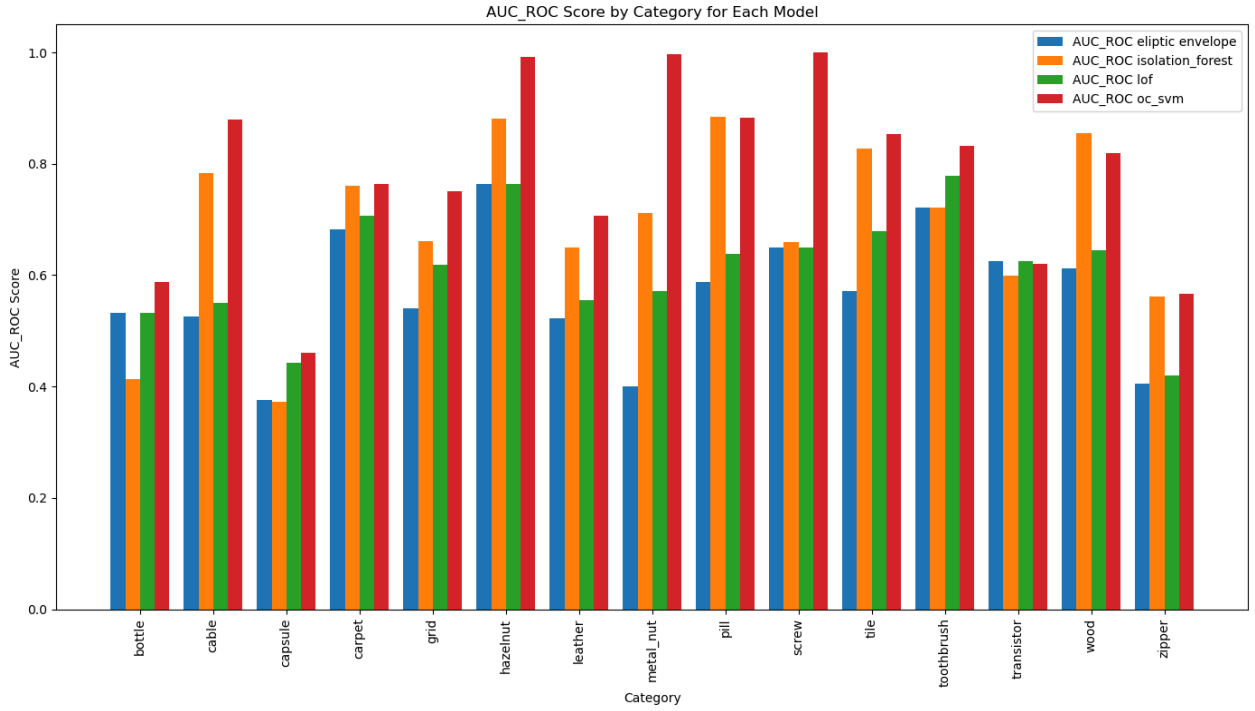


Fig.8: AUC ROC scores for each Machine Learning model across categories. OC-SVM (red) consistently shows the best performance.

For each category, the model has been trained on 60% of the original MVTec AD train set. The remaining 40% have been split into three cross validation sets, after 75% of it have been altered to contain synthetic anomalies. After performing a grid search for optimized hyperparameters during training and cross validation, the following parameter set has been chosen:

- *Upper bound for fraction of training errors and lower bound for fraction of support vectors*
  `nu = 0.001`
- *Kernel type radial basis function*
  `kernel = 'rbf'`
- *Kernel coefficient for radial basis function*
  `gamma = 'scale'`

An overview over the whole process is shown in Fig. 9.

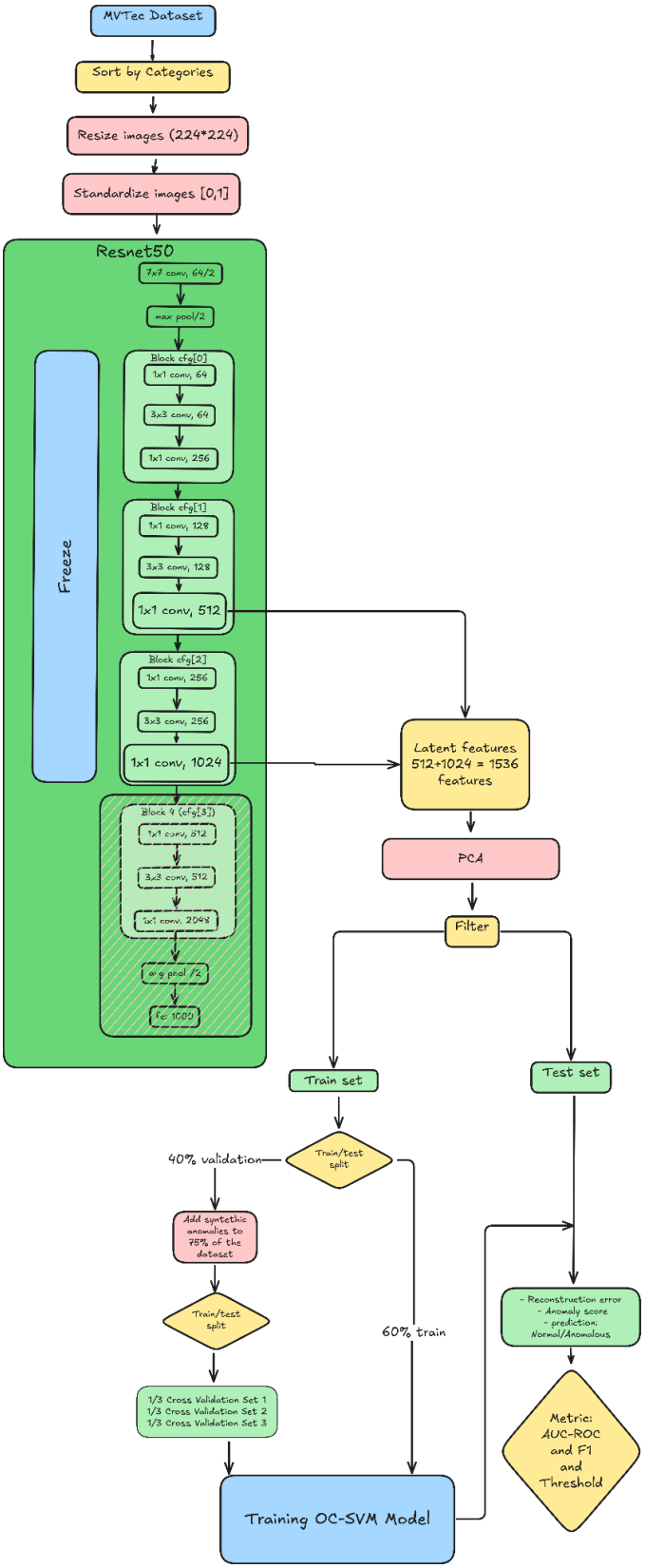


Fig. 9: Overview over the sequence of feature extraction, data splitting and model fitting for the OC-SVM model

Testing of the model on the original MVTec AD test set has yielded the following AUC ROC curves for evaluating model performance (Fig. 10).

Results

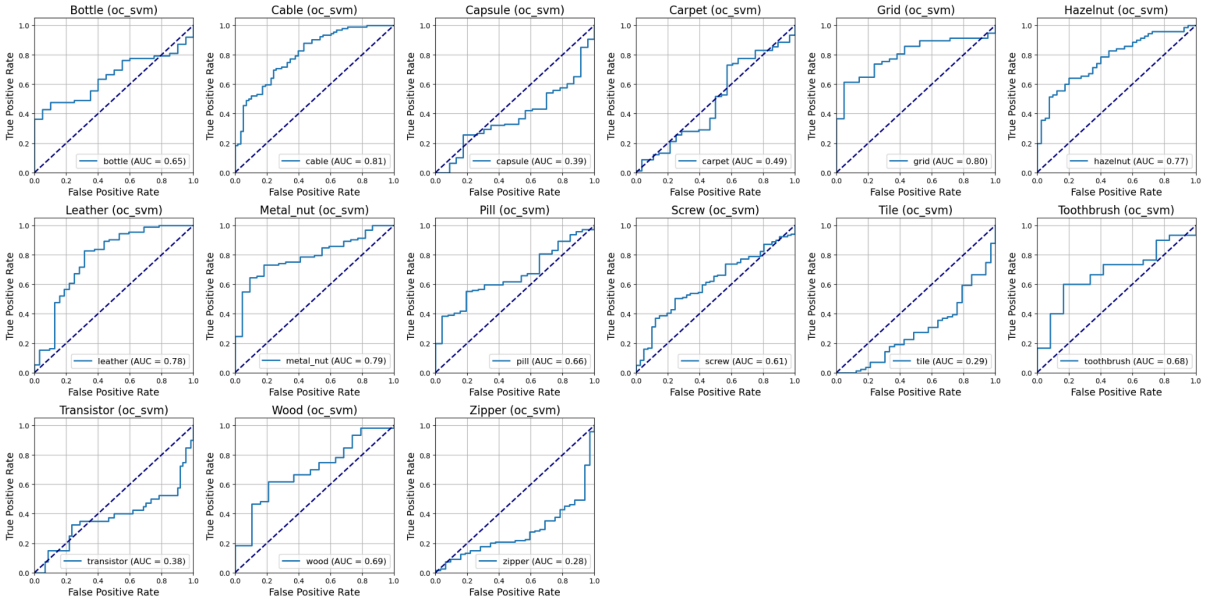


Fig. 10: AUC ROC curves for OC-SVM across all categories

OC-SVM's AUC ROC curves for the 15 categories show that, while being the best of the evaluated Machine Learning models, its performance varies widely across categories. For some, like cable, grid, hazelnut, or leather, the AUC ROC curve indicates suitability, while for others, like capsule, tile, or zipper, performance is clearly inadequate.

The evaluated four models can therefore hardly be considered a solution to the problem at hand and can at best serve as a baseline for further model development.

KNN + ResNet50

KNN has been evaluated as one further Machine Learning model which gave much better results then the baseline models, even without the necessity to reduce the test or training sets to gain cross-validation data. In this approach, the algorithm is trained and tested on the original MVTec AD data, which proved sufficient for effective training. Fig. 11 shows an overview over the training and testing process.
Features are extracted from ResNet50's block cfg(1) and cfg(2) latent layers as described above in version (b) of deep feature extraction. The algorithm is trained using parameters

- *Number k of nearest neighbors to be considered*
  `n_neighbors = 1`
- *Euclidean distance as metric to measure distance*
  `metric = 'euclidean'`

For each test sample, distance to the nearest neighbor in the train set is computed and serves as the basis for an anomaly score.

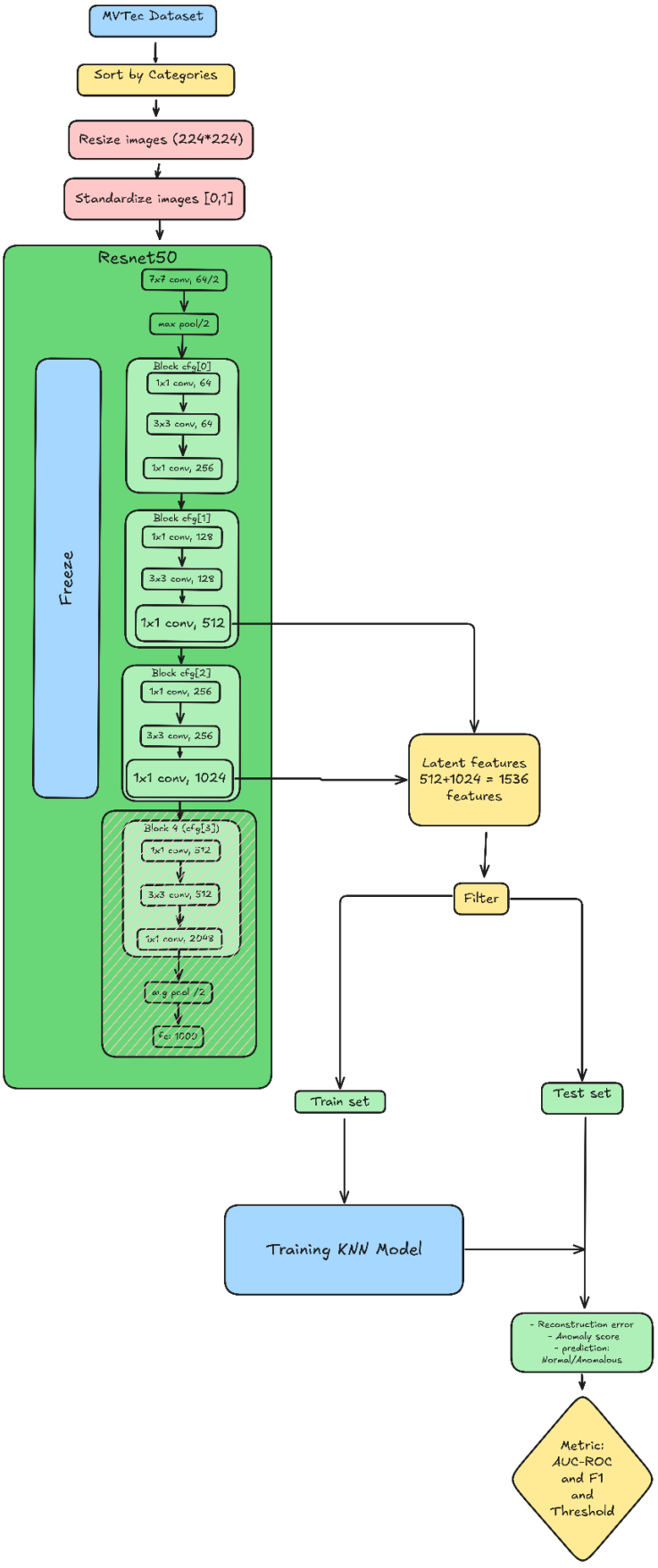


Fig. 11: Overview over the sequence of feature extraction, data splitting and model fitting for the KNN model

Results

Fig. 12 shows AUC ROC curves across categories, while Fig. 13 shows confusion matrices across categories for a threshold that balances precision and recall by optimizing the F1-score.

As evidenced by the graphs, the approach shows 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.

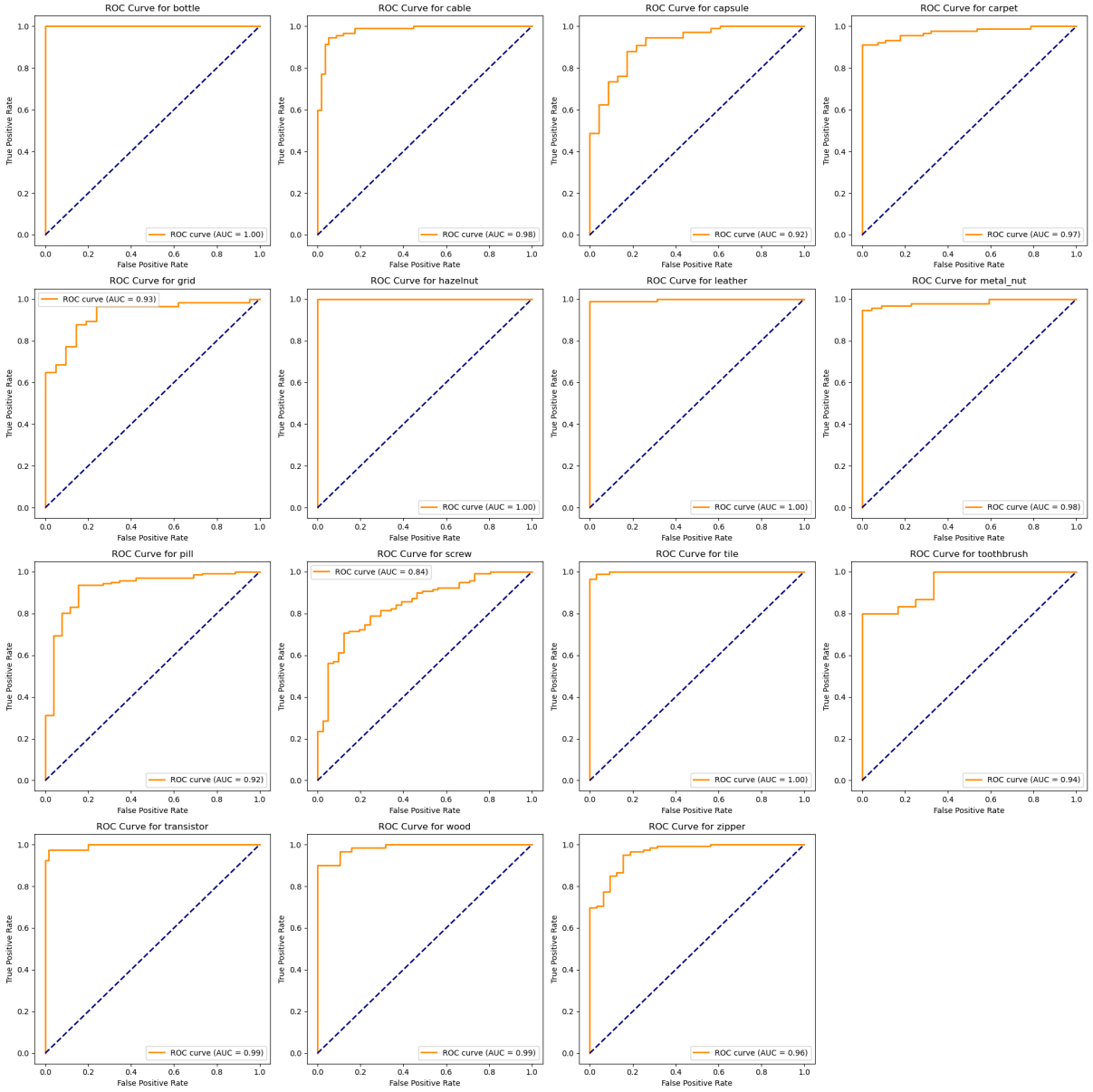


Fig. 12: AUC ROC curves for KNN across all categories

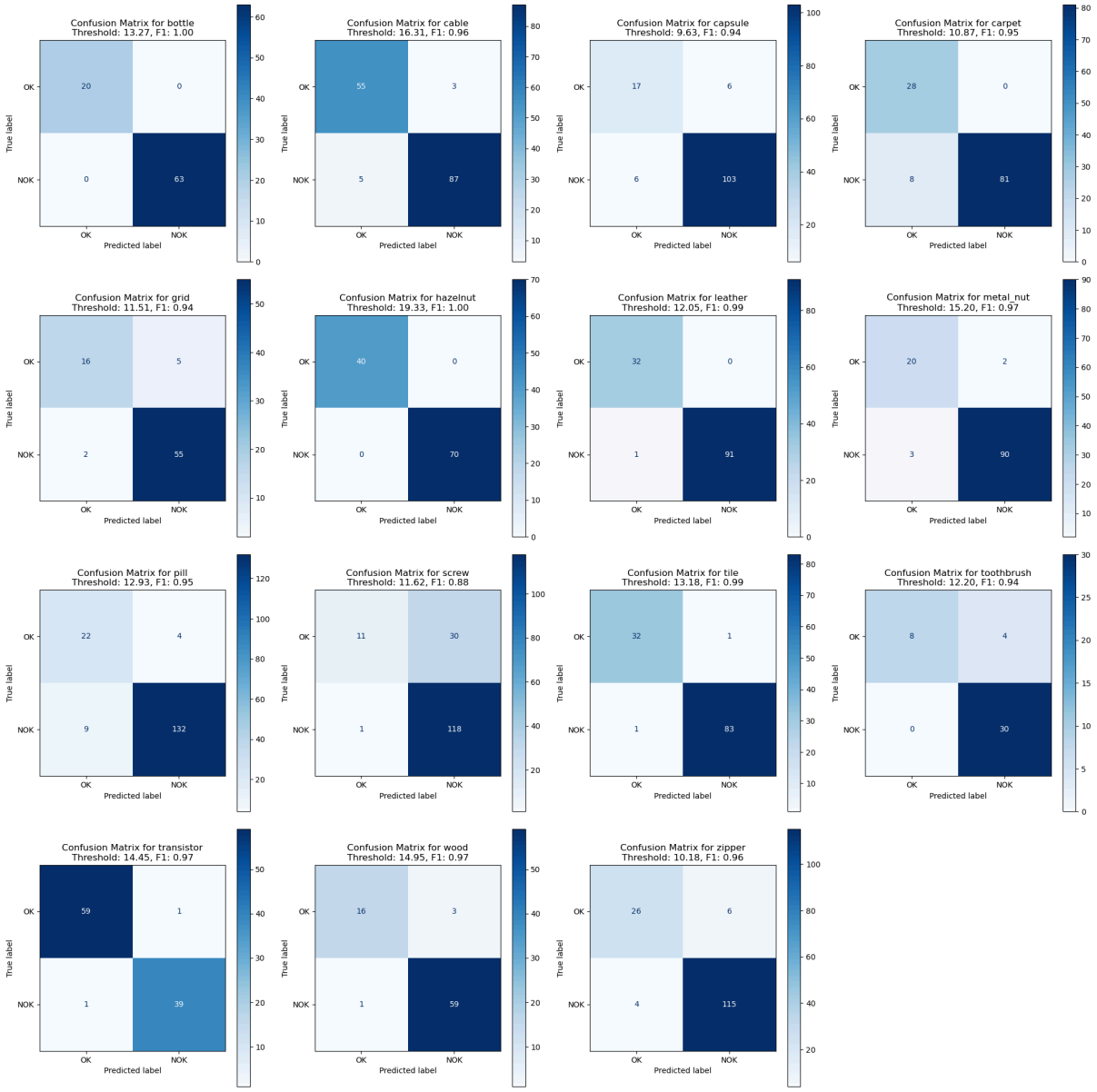


Fig. 13: confusion matrices for KNN across all categories

Anomaly Visualization

Although the difference between a test image and its nearest neighbor, that is the basis for computing the anomaly score, is not computed pixel-wise but in feature space, it can still be used to create a heatmap that indicates the location and color-codes the degree of the anomaly found in the image. Fig. 14 shows a heatmap and black-and-white image created from such a difference, as well as the original test image and its MVTec ‘ground_truth’ image.

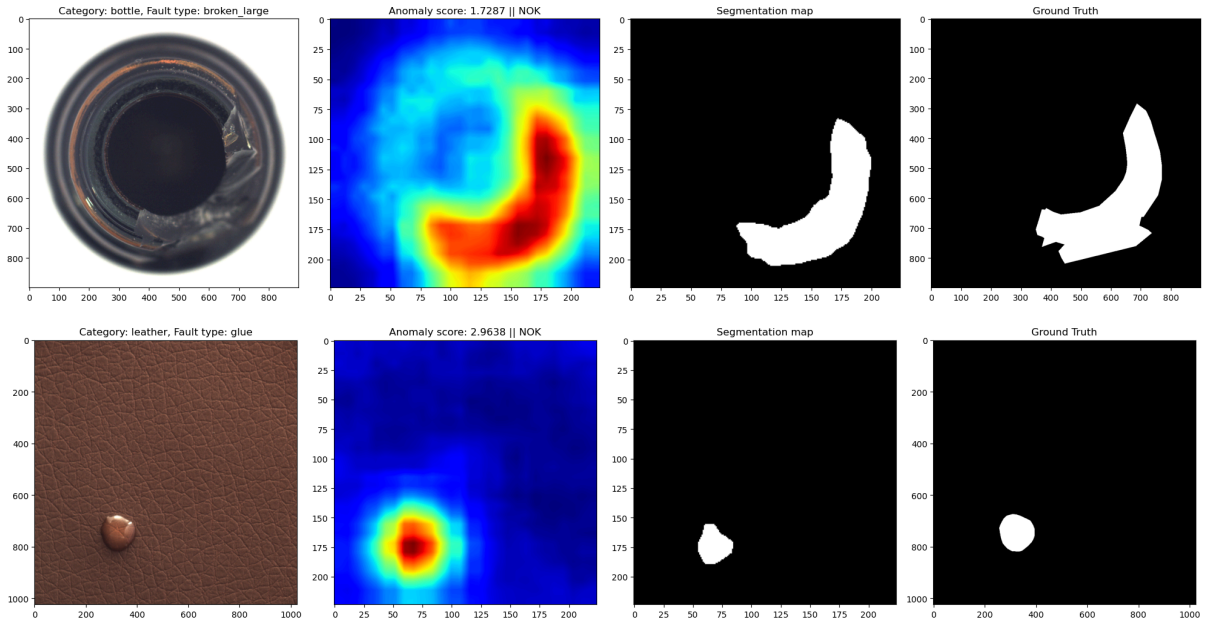


Fig. 14: from left to right: original image, KNN-difference-based heatmap, KNN-difference-based black-and-white image, original MVTec ground truth image for categories 'bottle' (upper) and 'leather' (lower)

Deep Learning

As a Deep Learning approach, a **Convolutional Autoencoder (CAE)** has been chosen.
Autoencoders are neural networks designed for unsupervised learning, primarily used for dimensionality reduction, feature learning, data compression, and anomaly detection.
An autoencoder consists of two parts, acting as an encoder and a decoder, respectively. The encoder is trained to compress the input into a lower-dimensional latent representation by extracting essential features. The decoder reconstructs the input from this latent representation in a way that meaningful information is retained while noise or redundant data are being discarded.
Convolutional Neural Networks (CNN) are specifically designed to process image data. They use convolutional layers to extract hierarchical features from input data, enabling them to detect patterns such as edges, textures, and objects.
In Convolutional Autoencoders, the encoder-decoder approach is implemented using convolutional layers to take advantage of their specific ability for image processing. This optimizes feature extraction and effective compression into latent space.

To be used for anomaly detection, an autoencoder is trained purely on the 'normal' data, so that it specifically learns to optimize compression and reconstruction of images without anomalies and minimize their reconstruction error. When presented with a different (or anomalous) image, compression and reconstruction by the trained model will result in unusually large reconstruction errors. This reconstruction error, which is the difference between original input and the output that the decoder reconstructs from the latent space representation, thus can be interpreted as an anomaly score.

Model

The architecture used to implement the Convolutional Autoencoder is as follows:

Encoder:

- Three convolutional layers progressively reduce spatial dimensions while increasing feature channels
- The first layer captures low-level features, while deeper layers extract more complex patterns.
- Batch normalization and ReLU activation are applied after each convolution

Decoder:

- Mirrors the encoder with three transposed convolutional layers to reconstruct the input
- The first decoder layer expands basic structures, while deeper layers refine details
- The final layer reconstructs the original feature map dimensions with minimized loss

Training aimed to minimize reconstruction error using Mean Squared Error (MSE) loss.
The model 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.

As for the KNN approach above, dataset expansion using augmentation or synthetic anomalies was not necessary to achieve satisfactory results and the model was trained and tested on the original MVTec AD data. Fig. 15 shows an overview over the training and testing process.

Features are extracted from ResNet50’s block cfg(1) and cfg(2) latent layers as described above in version (b) of deep feature extraction. The model was trained on 80% of the original MVTec AD train set, using the remaining 20% for validation, and then tested on the original MVTec AD test set. An anomaly score was determined for each input using mean reconstruction error plus three standard deviations, ensuring that anomalous samples were effectively identified.

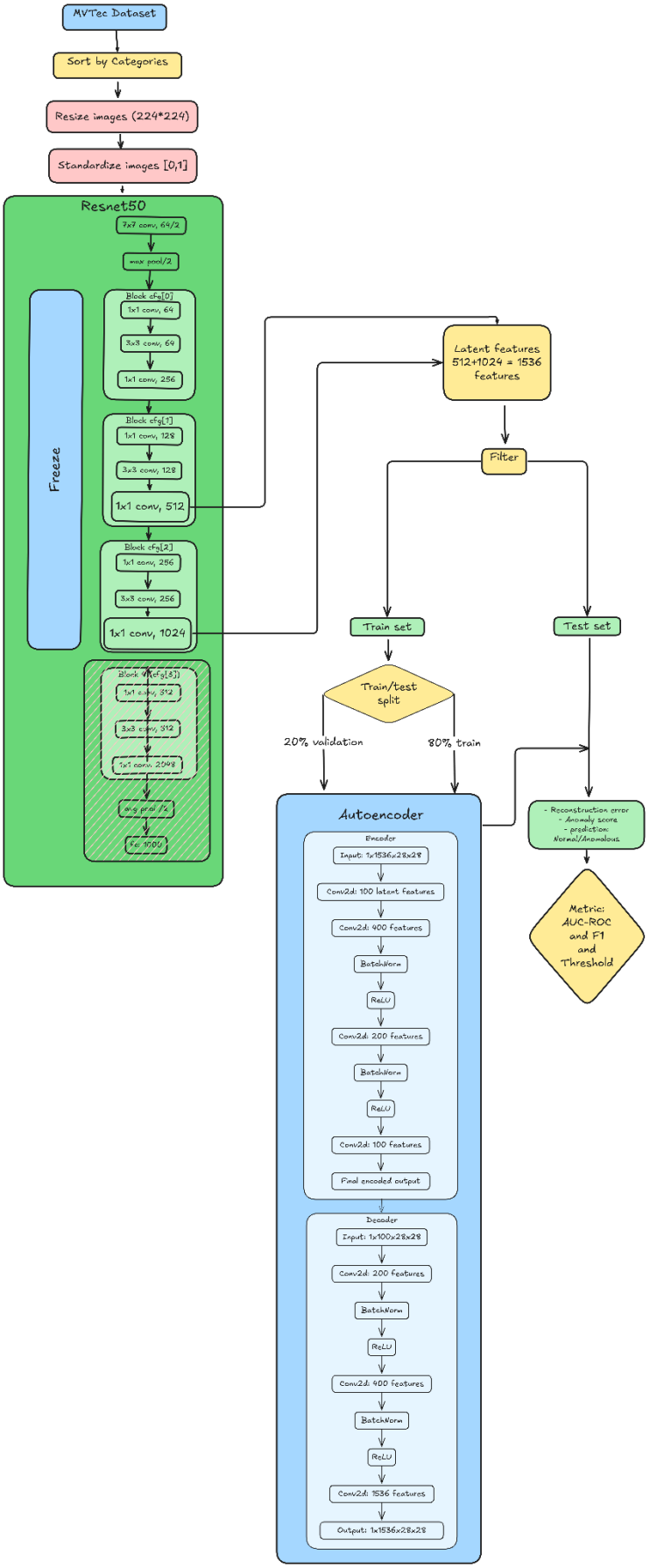


Fig. 15: Overview over the sequence of feature extraction, data splitting and model fitting for the CAE model

Results

Fig. 16 shows AUC ROC curves across categories, while Fig. 17 shows confusion matrices across categories for a F1-score-optimized threshold.

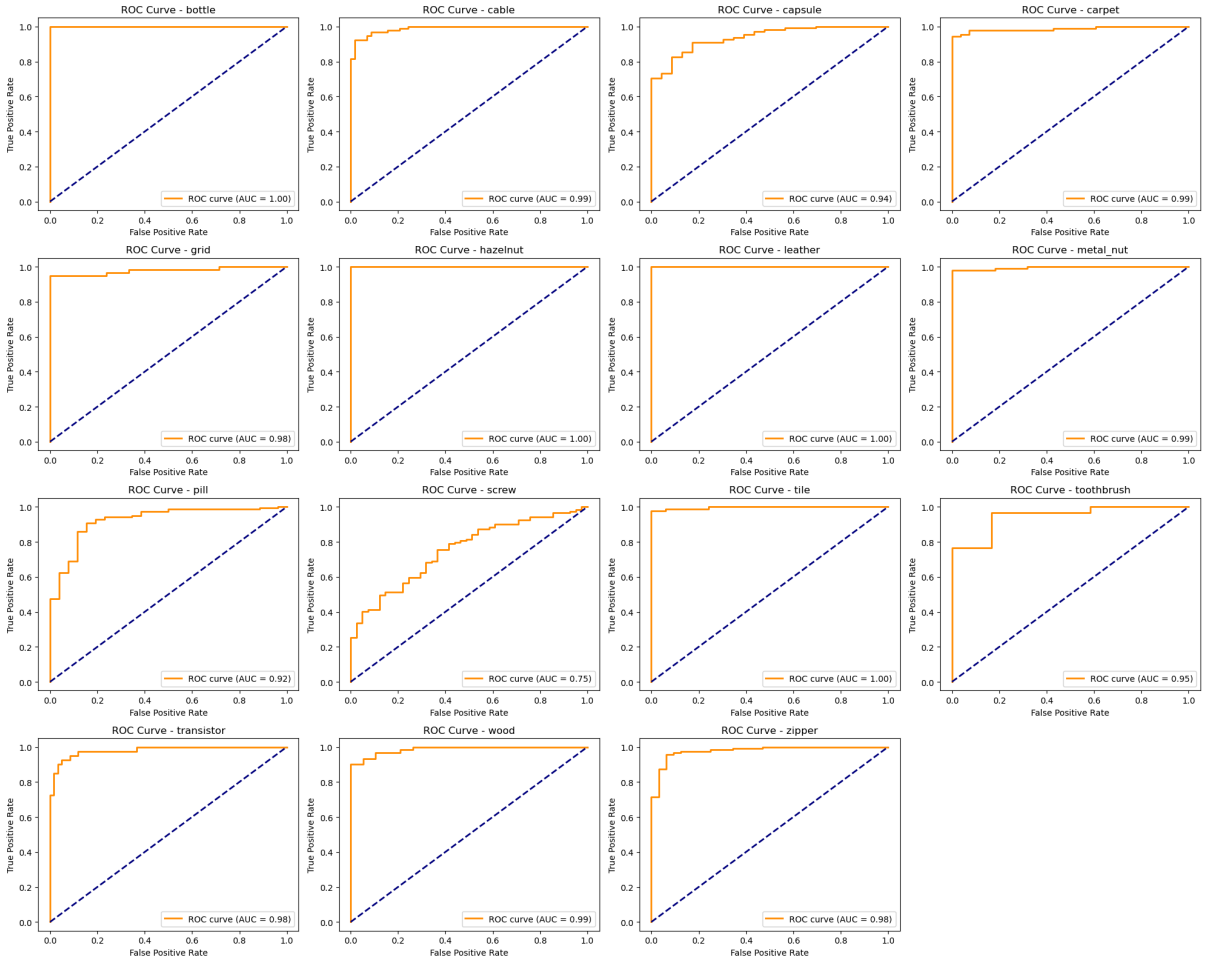


Fig. 16: AUC ROC curves for Convolutional Autoencoder across all categories

Most categories exhibit an Area Under the Curve (AUC) value close to 1.00, indicating excellent performance. Some categories such as *bottle, hazelnut, leather, and tile* have perfect AUC scores of 1.00, suggesting flawless discrimination between normal and anomalous images. Only 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. In this category it is outperformed by the KNN approach, but still significantly better then the OC-SVM model.

The consistently high AUC scores across categories suggest that the autoencoder-based feature extraction and anomaly detection pipeline generalizes well across various types of industrial components.

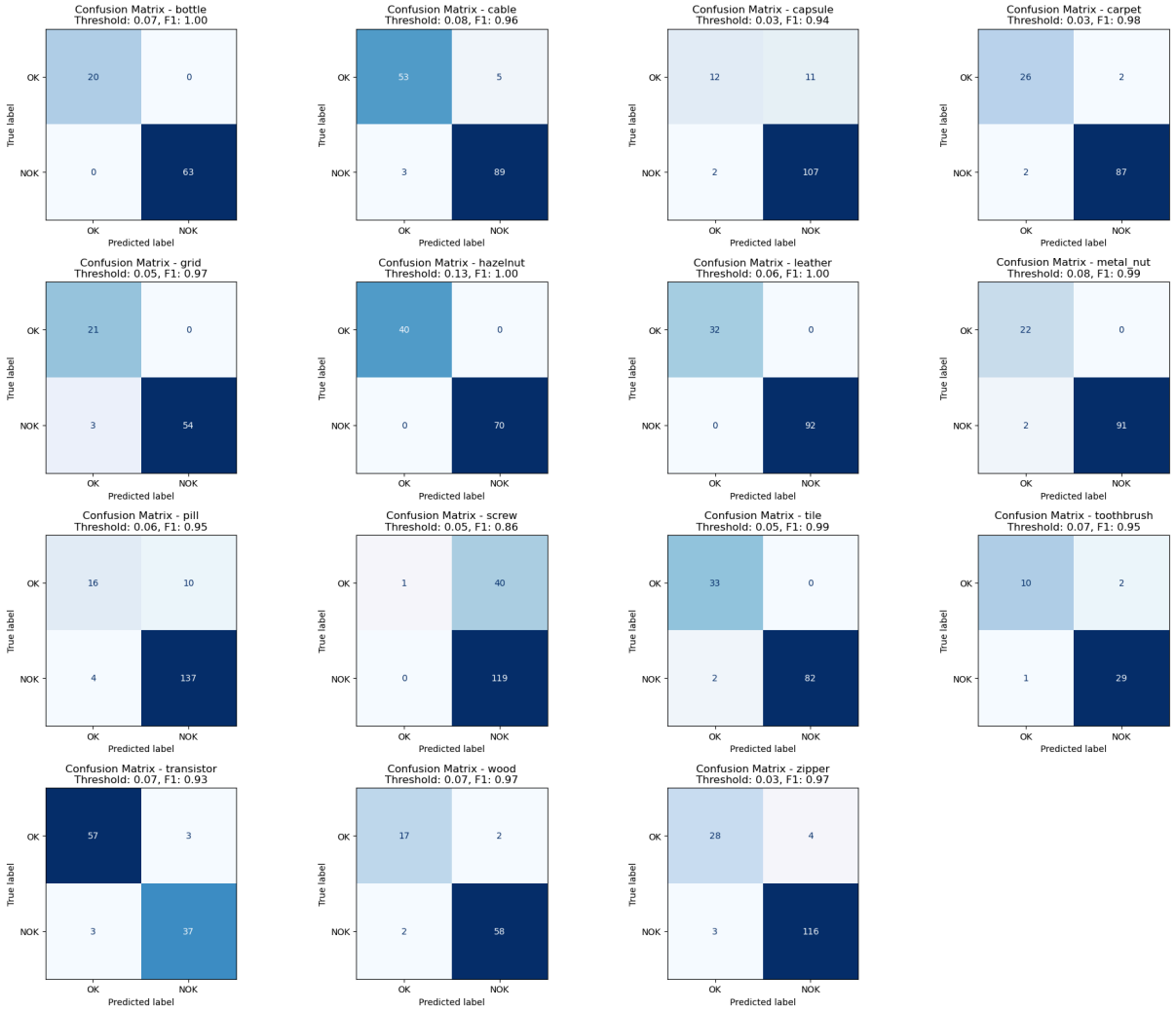


Fig. 17: confusion matrices for Convolutional Autoencoder across all categories

The confusion matrices in the image provide a detailed evaluation of the model's classification performance for different categories. As already observed in the AUC-ROC curves, the *bottle, hazelnut, leather,* and *tile* categories exhibit perfect classification with an F1-score of 1.00, meaning there were no misclassifications. Other categories such as *metal nut, carpet, and transistor* also achieved high F1-scores (>0.95), indicating strong generalization. The *capsule* category has a relatively higher number of false positives (11) compared to the other categories, impacting its F1-score (0.94). As above, the *screw* category has the lowest performance (F1-score 0.86), showing more significant misclassification errors, particularly with false negatives.

Anomaly Visualization

Autoencoder identify anomalies by computing the difference between the original input test image and the reconstruction of its projection into the latent space. This exploits the fact that the model that has been fitted to ‘normal’ data will perform poorly on unexpected (anomalous) input. As with the KNN approach above, this difference can be used to create a heatmap that indicates the location and color-codes the degree of the anomaly found in the image. Fig. 18 shows a heatmap and black-and-white image created from such a difference, as well as the original test image and its MVTec ‘ground_truth’ image.

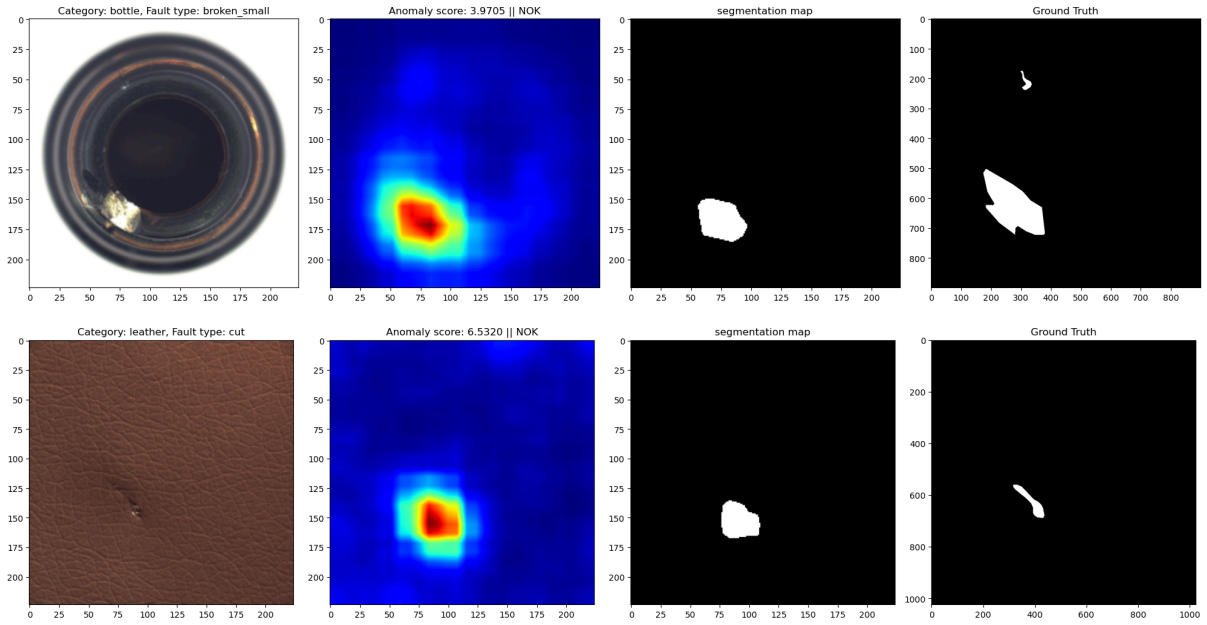


Fig. 18: from left to right: original image, KNN-difference-based heatmap, KNN-difference-based black-and-white image, original MVTec ground truth image for categories 'bottle' (upper) and 'leather' (lower)

Potential Improvement

The described CAE was one of two versions evaluated in this project and was chosen for this final report due to its superior performance. The other one was an autoencoder combined with a custom Resnet3 for feature extraction (for more detail, please see report 2). In that approach, an additional **augmented dataset** has been created to provide a wider database for training, to improve performance. The augmented data expands the existing training dataset to approximately 1,500 images per category (examples see Fig. 19 and 20). The augmentation pipeline introduces controlled variations to enhance the model's generalization ability. The transformations applied include:

- **Resizing**: Images are resized to a fixed shape of 224×224 pixels.
- **Random Horizontal Flip**: A 50% probability of flipping the image horizontally.
- **Random Scaling & Translation**: A 75% probability of applying random affine transformations with slight translation and scaling variations.
- **Rotation**: A 75% probability of rotating images by -90°, 90°, or 180°.
- **Gaussian Blur**: A slight Gaussian blur with a kernel size of 3 and a randomly chosen sigma between 0.01 and 0.05.
- **Tensor Conversion**: The final step ensures that all images are converted into tensors suitable for deep learning models.

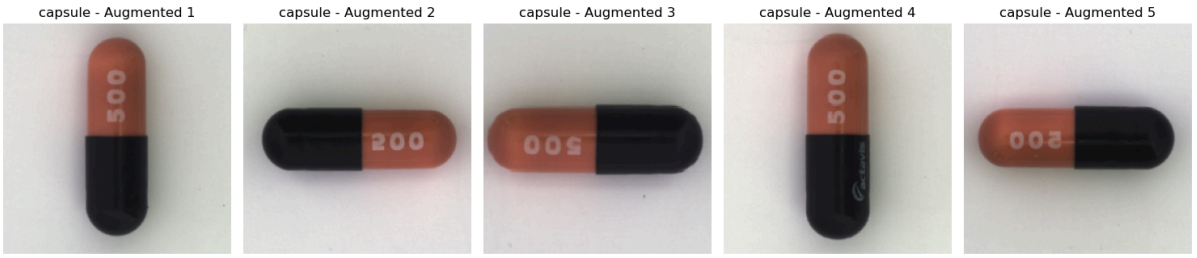


Fig. 19: augmented images of category 'capsule'

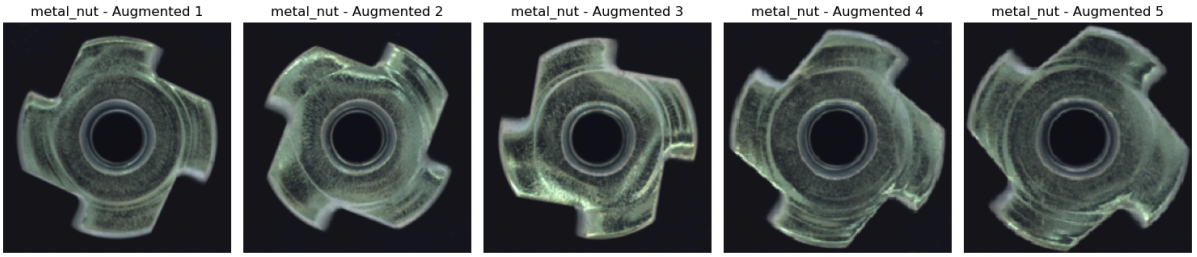


Fig. 20: augmented images of category 'metal_nut'

This expanded training base did improve model performance for the autoencoder with Resnet3, but the CAE presented above shows even better performance, even without augmented training data.
Due to the scope of the project and the limited time available, an approach combining the more successful CAE with training on the augmented database has not yet been completed but proves definitely promising.

Conclusion

Throughout the project several Machine Learning and Deep Learning models have been applied to the available image data. We compared the different models, as well as different approaches in data preparation to optimize anomaly detection performance.
The evaluation shows that the **Convolutional Autoencoder** and the **KNN approach** significantly outperform the other models, and both yield better results using the **deep feature extraction** approach.

Consequently, our **key takeaways** from this project are:

- Deep feature extraction significantly outperforms manual features, as seen in the consistently better performance of ResNet50-based methods.

- Of the two evaluated deep feature extraction approaches, one using ResNet50 blocks 1 and 2, the other one using block 3, both seem to be similarly suitable on average over all categories, but within single categories, often one showed significantly better performance over the other.

- The KNN approach leverages memory-based similarity comparisons, providing a robust alternative to parametric models.

- Deep learning using an autoencoder provides strong reconstruction-based anomaly detection, ensuring comprehensive feature learning.

- Synthetic validation data introduced diversity and improved cross validation, but also posed challenges, as some anomalies may not perfectly mimic real-world defects.

While the project results are promising, we propose the following additions or changes for **potential future improvements**:

- Data Augmentation Refinement: Exploring more advanced augmentation techniques, such as generative models (e.g., GANs), could enhance training diversity.

- Model Ensembling: Combining multiple anomaly detection models could improve robustness and generalization.

- Hyperparameter Optimization: Fine-tuning model hyperparameters further, using techniques like Bayesian optimization, could boost performance.

- Alternative Architectures: Exploring transformer-based architectures or vision encoders like ViTs for anomaly detection could yield even better results.

- Better Synthetic Data: Refining the process of synthetic anomaly generation to better align with real-world defects.

Overall, the introduced approaches provide a solid foundation for industrial anomaly detection tasks. Further optimizations and explorations in feature extraction and model selection could push performance even higher in future studies.