Project MVTec Anomaly Detection Report 3

Table of Contents

[Project Description 2](#_heading=h.ooqqeu39wlxy)

[**MVTec Anomaly Detection Dataset 3**](#_heading=h.2s4cm59ax5gt)

[Dataset 3](#_heading=h.ldewqmtpvcdd)

[Analysis 3](#_heading=h.2ql2623tcaps)

[**Modelling 4**](#_heading=h.vwicu3s85fex)

[Additional Datasets - synthetic and augmented 4](#_heading=h.1u1gm01q7nii)

[Feature Extraction - direct and deep 6](#_heading=h.rz7whhx0kig)

[Preprocessing 7](#_heading=h.kizejsnom2rr)

[Machine Learning 7](#_heading=h.38cq57m7jgvi)

[Data Splitting 7](#_heading=h.4g2gyntc9caq)

[Train-, Test- and Validation-Set Combinations 9](#_heading=h.nbid1hm5i8qc)

[Model 9](#_heading=h.4c0ih7t6u3qv)

[Results 11](#_heading=h.axi3xpmxqeko)

[KNN + ResNet50 11](#_heading=h.vnix4vd1hkxz)

[Model 11](#_heading=h.qmjmsbtbmbbf)

[Results 13](#_heading=h.qi5x8goc1x95)

[Deep Learning 14](#_heading=h.d4dqrmlnxemf)

[Model 15](#_heading=h.mupz4noptr0a)

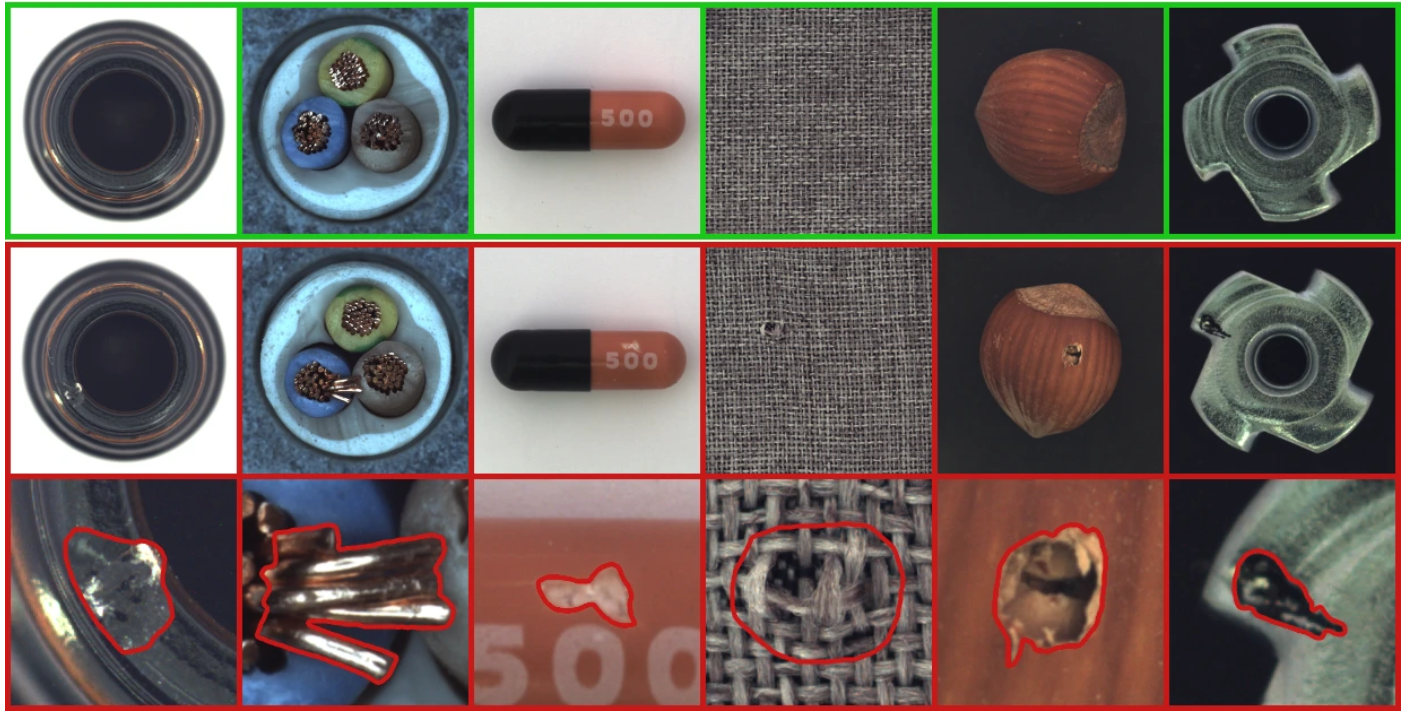
[Results 17](#_heading=h.h2i5mtz0cmge)

[Conclusion 18](#_heading=h.1t3h5sf)

[**Bibliography 18**](#_heading=h.ckkmmbz9hr92)

# 

# Project Description



Insuring production quality in industrial processes relies heavily on monitoring output in order 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.

In the course of 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

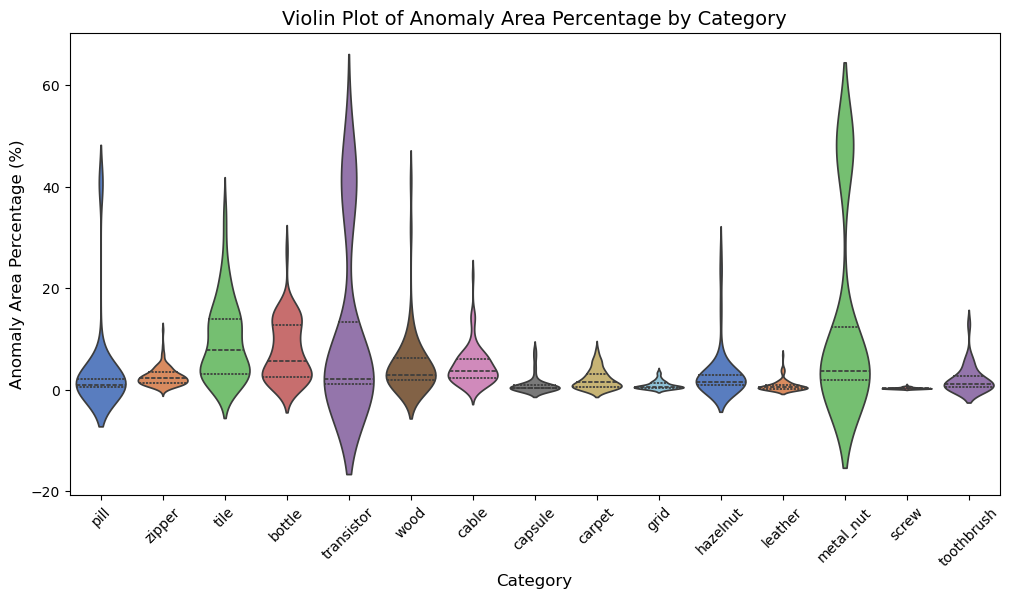
The available data is a public dataset provided by MVTec, a company specializing in software products and services for machine vision.

## Dataset

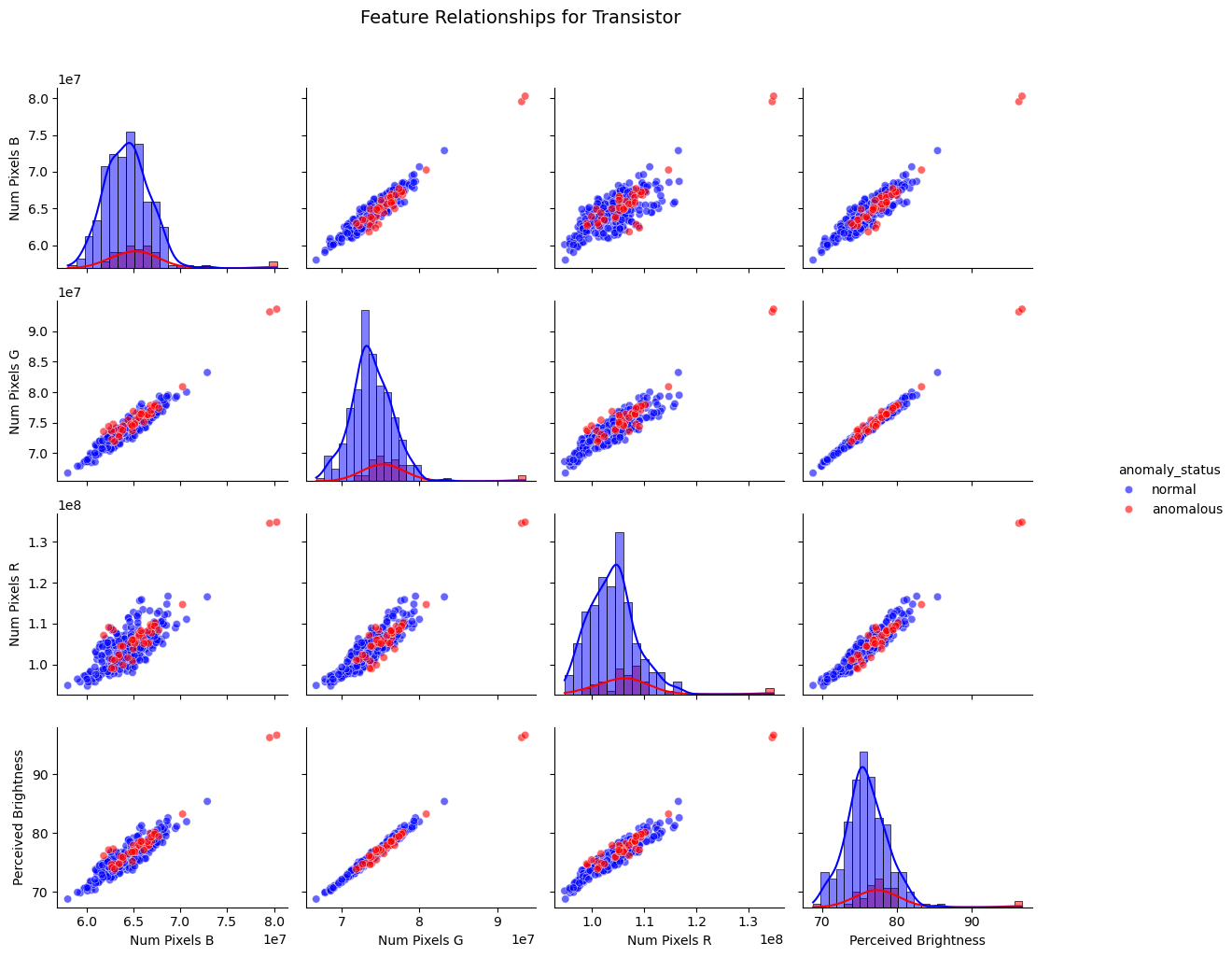
The dataset is meant for benchmarking anomaly detection methods with a focus on industrial inspection. It contains over 5000 high-resolution images divided into fifteen different object and texture categories. Each category comprises a set of defect-free training images and a test set of images with various kinds of defects as well as images without defects [1]. For each of the 15 categories, images contain a close up photograph of an object of that specific category (like cable, capsule, screw, etc.) of similar size and perspectives. In each category, images are grouped into ‘train’, ‘test’ and ‘ground\_truth’, where ‘train’ only contains images considered normal, while ‘test’ contains normal images as well as anomalous images. Anomalies are of different specific anomaly types that vary from category to category. The ‘ground\_truth’ group contains information about where in anomalous images the anomalies are located.

## Analysis

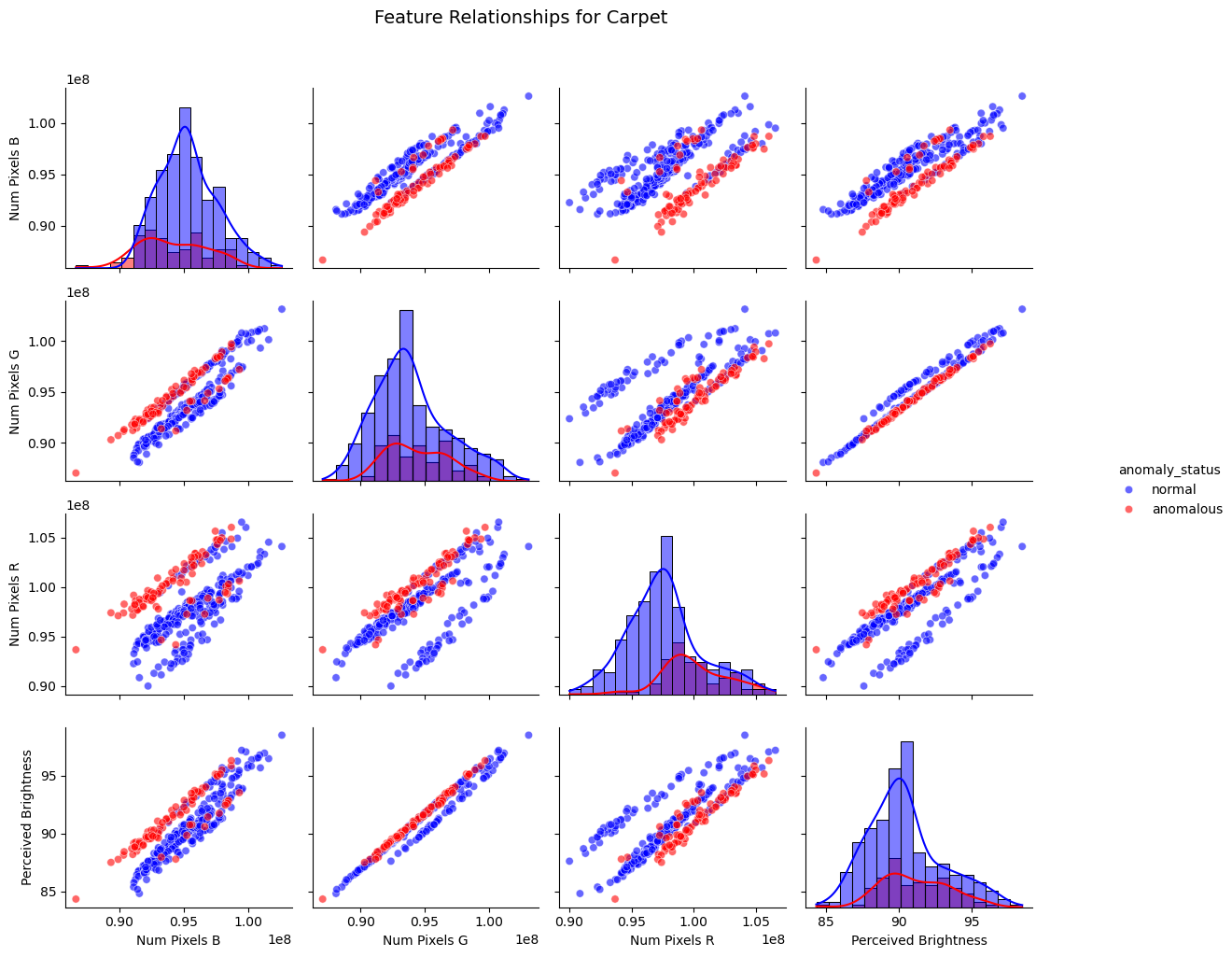
Preliminary analysis of images by categories indicate that although the size of the anomalous area in images varies significantly between categories (see Fig. 1), scatterplots show that for categories with larger anomalies, as well as such with smaller ones, there are features for which the anomalous 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. For more detailed information on dataset structure and analysis results, see project report 1.



*Fig. 1: …*



*Fig. 2: …*



*Fig. 3: …*

# Modelling

## Additional Datasets - synthetic and augmented

The fact that models have to 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 have to be split from the available test data. Therefore, two additional datasets have been artificially generated in order to serve as validation data.

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

The **augmented data** expands the existing training set to approximately 1,500 images per category. 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 224×224224 \times 224224×224 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.

Examples of these augmented images are shown below, demonstrating the various transformations applied (rotations, scaling etc...)

A close up of a pill

Description automatically generated

*Fig. 5: …*

A close-up of a metal object

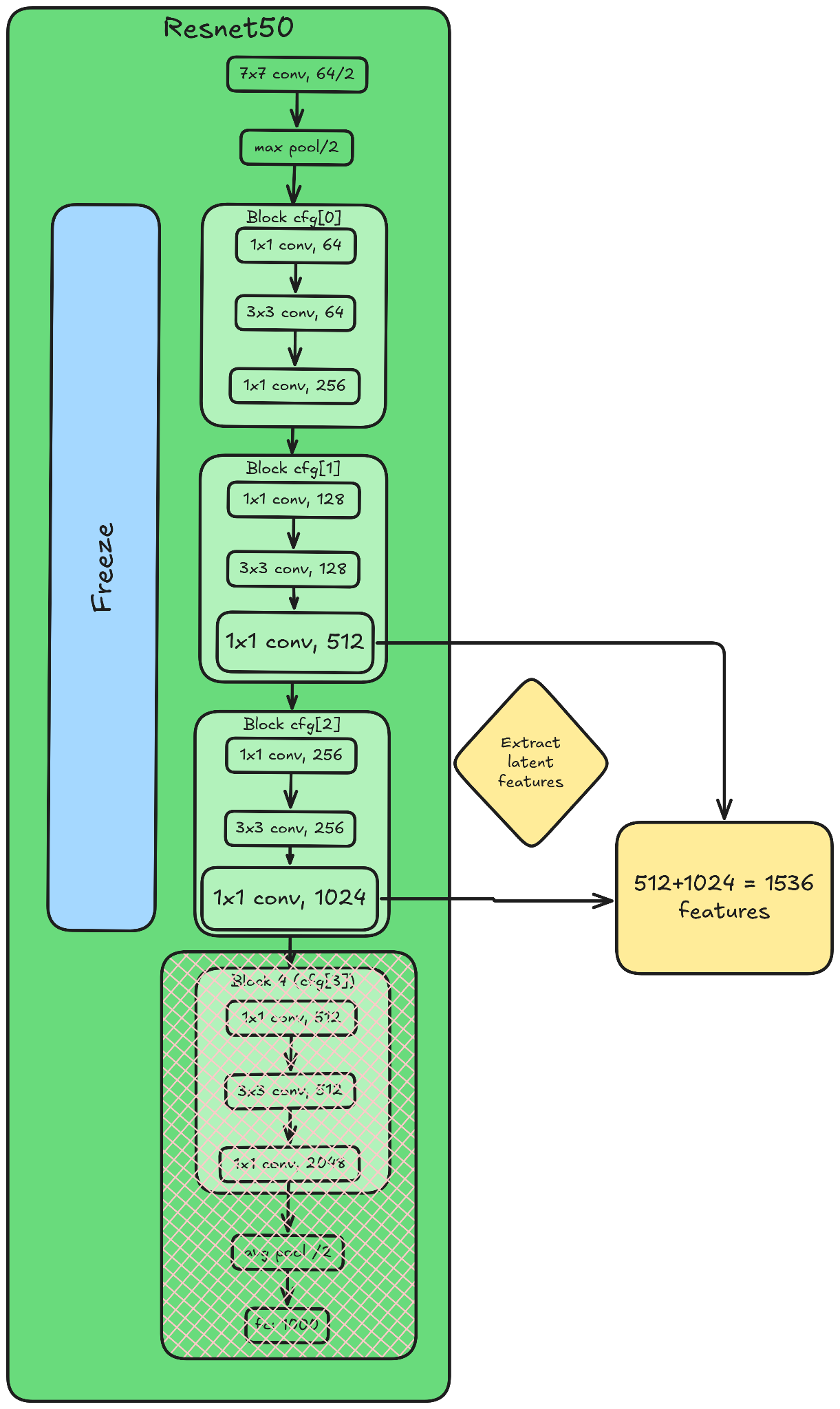
Description automatically generated

*Fig. 6: …*

## 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. For more detailed information, 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 shows the ResNet50 architecture as well as the locations after convolutional blocks 1, 2 and 3, where we extract features.



*Fig. 7:   
(a) 2048 features extracted from cfg[3] blocks (b) 1536 features extracted from cfg[1] and cfg[2] blocks*

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)

## 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

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:** 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.

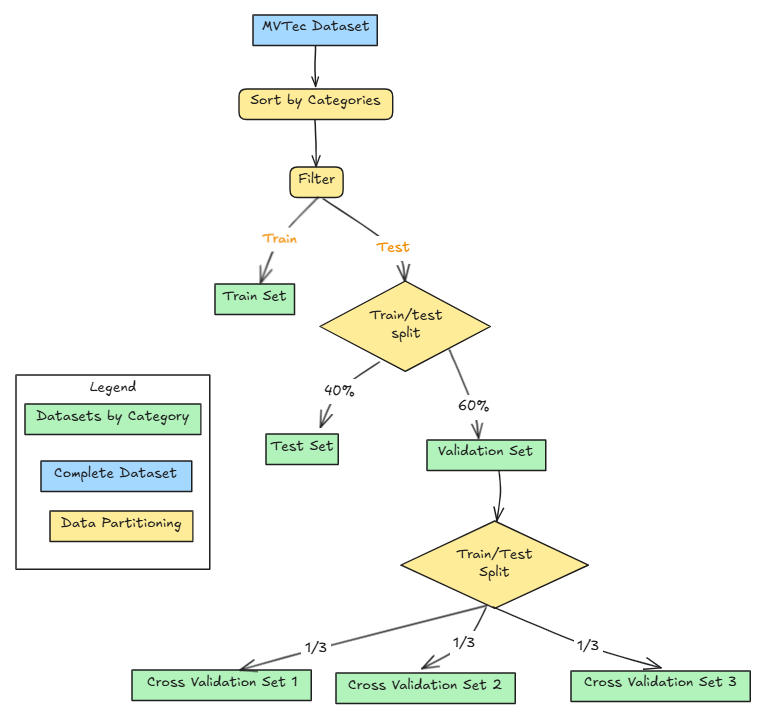
### Data Splitting

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) by splitting the original test-set, or
* (ii) by splitting the original train-set and adding synthetic anomalies to a fixed proportion.

#### (i) Original MVTec Data

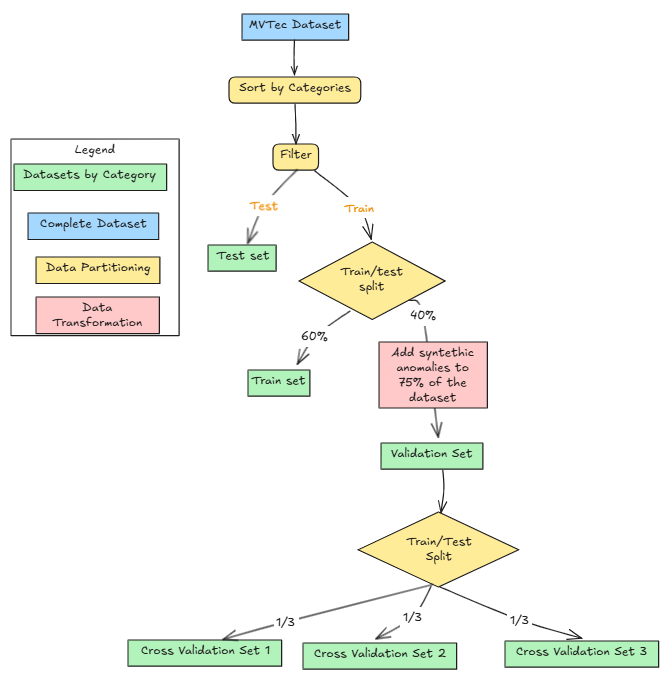
The following chart shows how data from the original MVTec Dataset is sorted by categories and then filtered into different subsets for training and testing.  
The original test data is then split into 40% remaining test data and 60% validation data, which is further split evenly into three cross validation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



*Fig. 8: …*

#### (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. 9: …*

### 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)

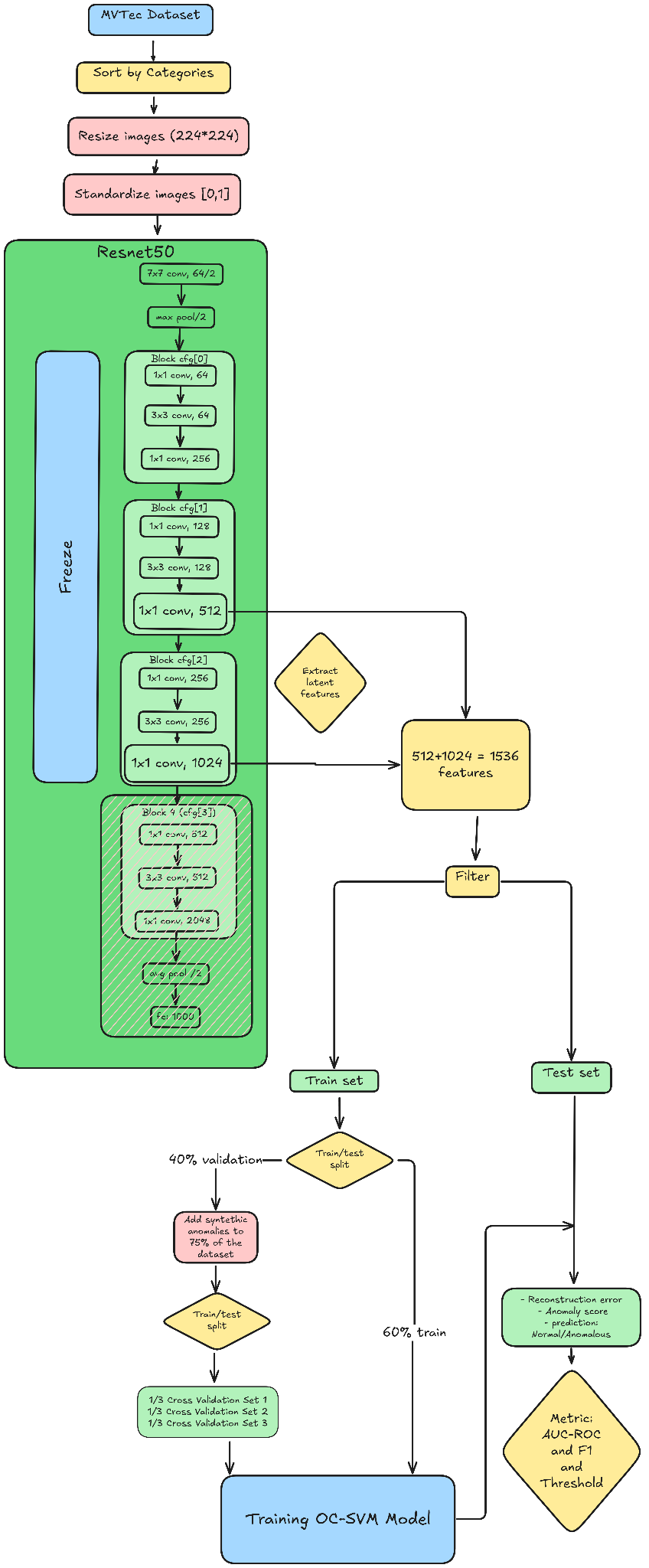
*Table 1: feature extraction and data splitting methods*

### Model

Compared to all other evaluated Machine Learning models, the **OCSVM** 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. An overview over the whole process is shown in Fig. 10.

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. Training and cross validation are performed on the following parameter set:

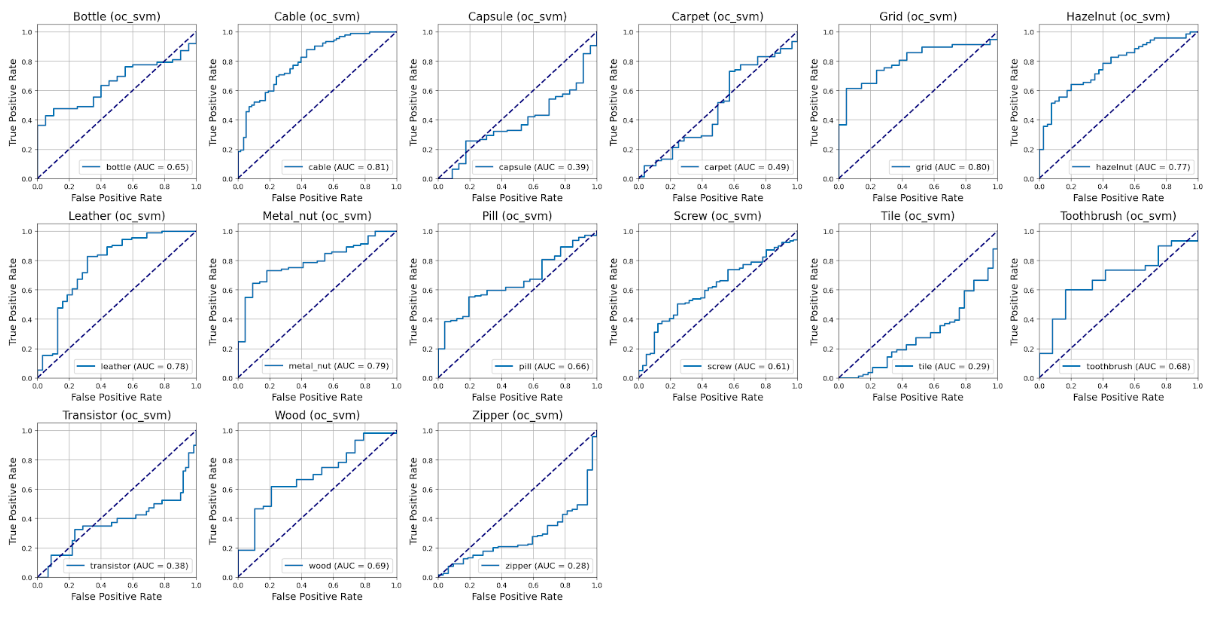
* nu = 0.001  
  *upper bound for fraction of training errors and lower bound for fraction of support vectors*
* kernel ∊ {‘rbf’, ‘linear’, ‘poly’}  
  *kernel type radial basis function, linear or polynomial*
* gamma ∊ {‘scale’, ‘auto’}  
  *kernel coefficient for ‘rbf’ and ‘poly’*
* degree ∊ {2, 3}  
  *degree of polynomial kernel*



*Fig. 10: …*

The model is tested on the original MVTec AD ‘test’ set.

### Results



*Fig. 11: …*

The 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 high suitability, while for others, like capsule, tile, or zipper, performance is clearly inadequate.

## KNN + ResNet50

### Model

In this approach, the algorithm is trained and tested on the original MVTec AD data. Expansion of the dataset using augmentation or synthetic anomalies was not necessary, as the results clearly showed that the data is sufficient for effective training.

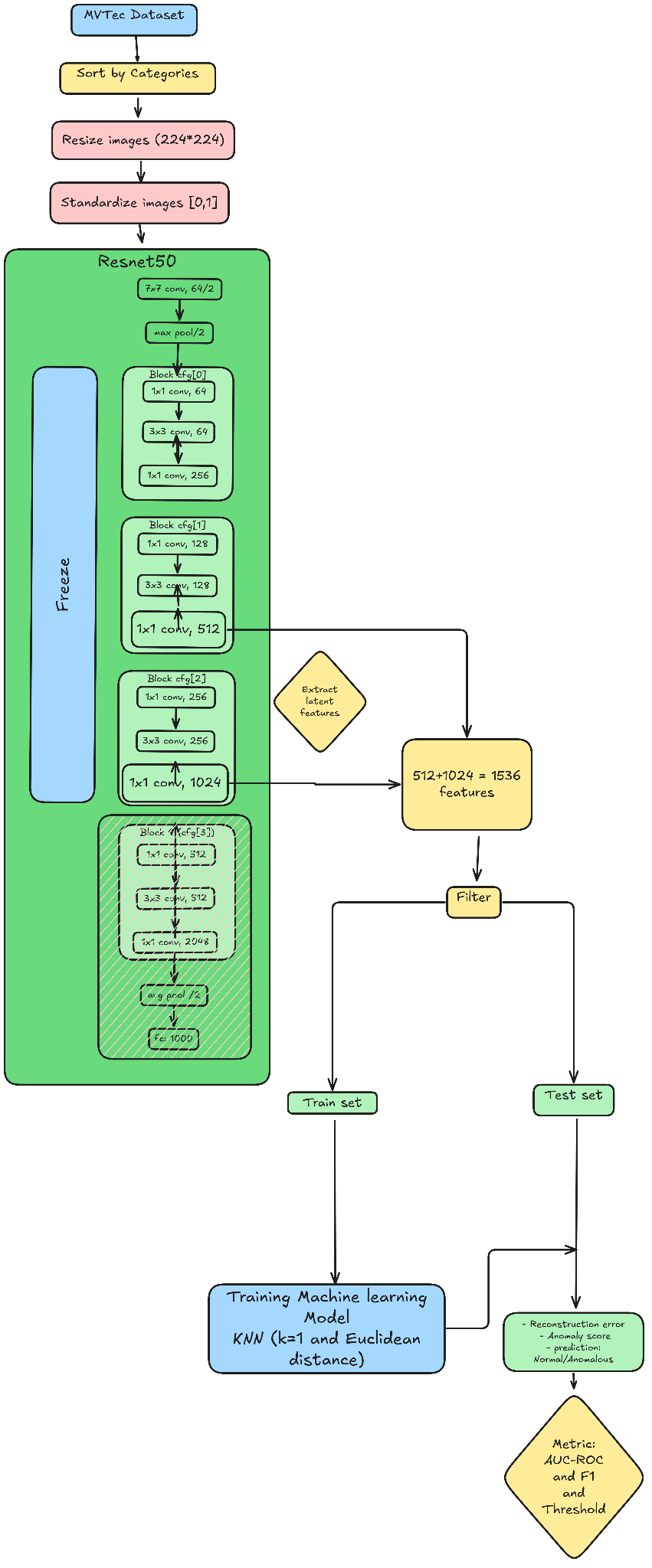
Fig. 12 shows an overview over the training and testing process.

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

* n\_neighbors = 5  
  *k = 5*
* metric = ‘euclidean’  
  *euclidean distance*

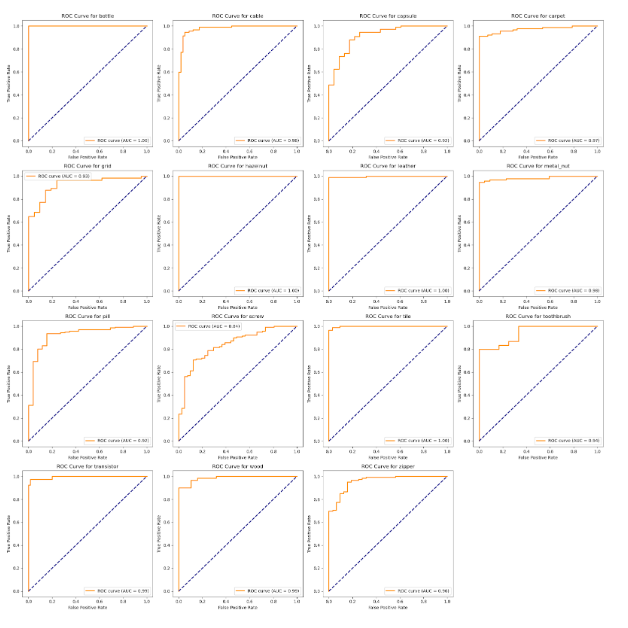
For each test sample, distances to the k nearest neighbors are computed and serve as the basis for an anomaly score.

Fig. 13 shows AUC ROC curves across categories, while Fig. 14 shows confusion matrices across categories for a threshold that balances precision and recall by optimizing the F1-score.

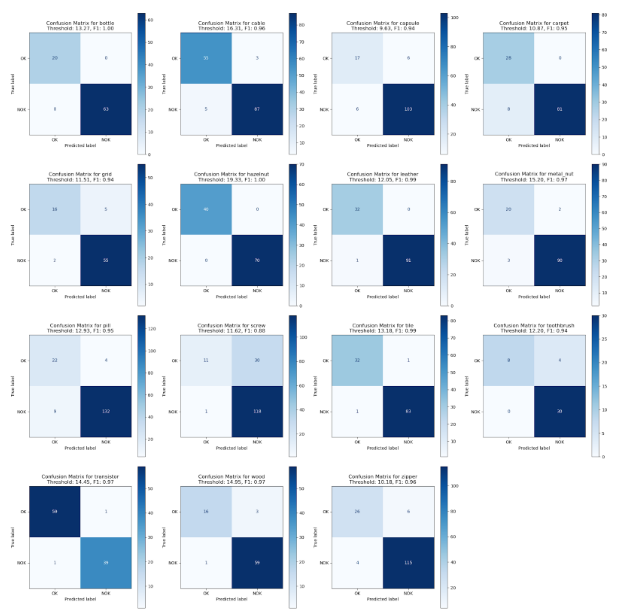


*Fig. 12: …*

### Results



*Fig. 13: …*



*Fig. 14: …*

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.

## Deep Learning

As a Deep Learning approach, a **Convolutional Autoencoder** 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. 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 in order 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

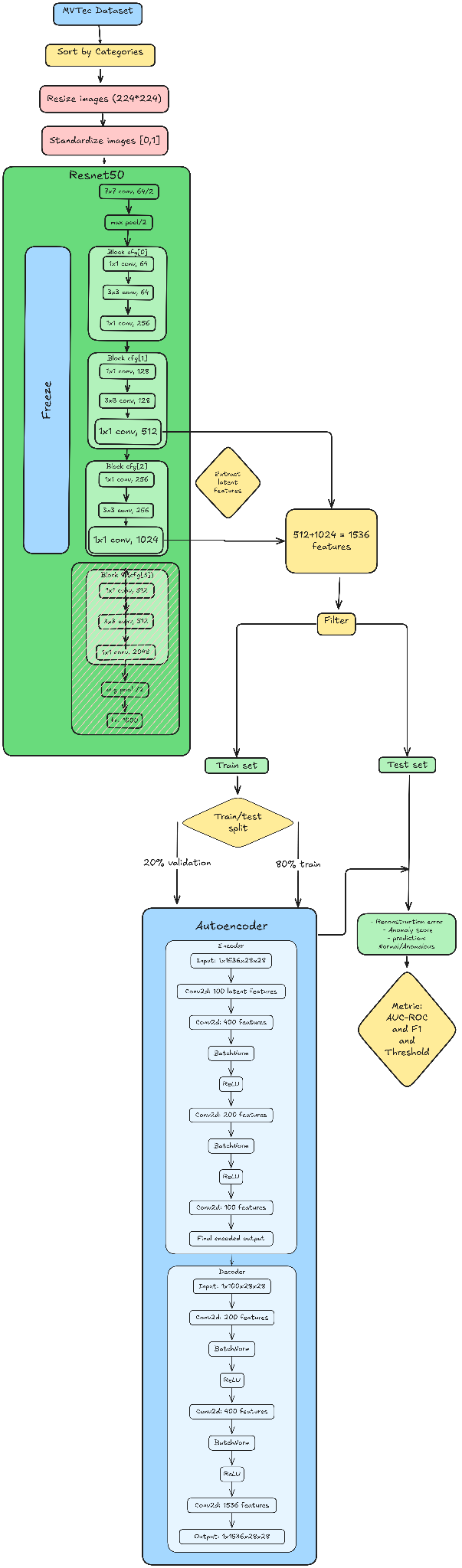
Training aimed to minimize reconstruction error using Mean Squared Error (MSE) loss.

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 satisfying 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.

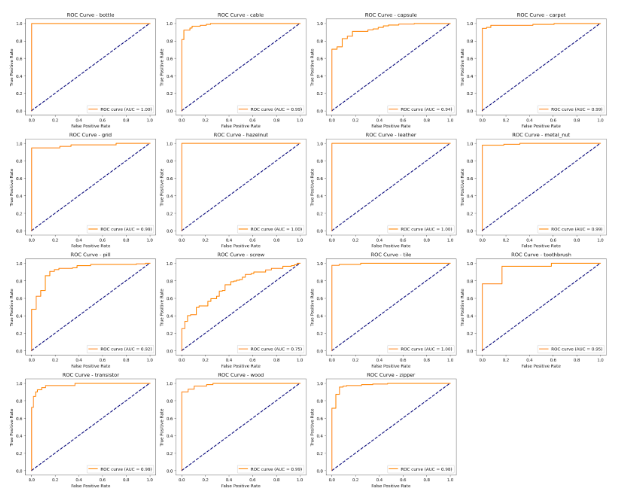
Features are extracted from ResNet50’s block cfg(1) and cfg(2) latent layers as described above in version (b) of deep feature extraction. The model was trained on 80% of the original MVTec AD ‘train’ set, using the remaining 20% for validation, and then tested on the original MVTec AD ‘test’ set. An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



*Fig. 15: …*

### Results

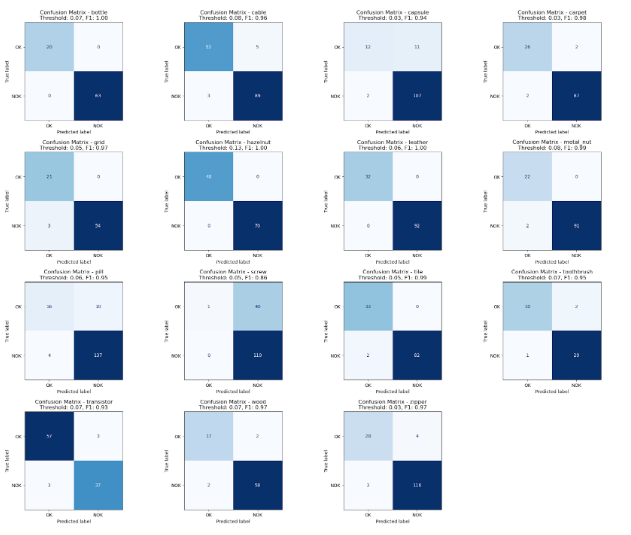
Fig. 16 shows AUC ROC curves across categories, while Fig. 17 shows confusion matrices across categories for a F1-score-optimized threshold.



*Fig. 16: …*

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 OCSVM 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: …*

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.

# Conclusion

[...]

# Bibliography

[1] *The MVTec anomaly detection dataset* (2025, February 10) <https://www.mvtec.com/company/research/datasets/mvtec-ad>