Project MVTec Anomaly Detection

Report 1: exploration, data visualization and data pre-processing

Project Description

The aim of the project is to perform **anomaly detection in visual data**, in order to identify defects or irregularities in photographic representations of specific objects.

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

Dataset Overview

The dataset contains image data in .png format, divided into **15 categories**:

- bottle (bottles, standing, photographed from above into the bottle neck)
- cable (cable cross sections, containing three different isolated wire bundles)
- capsule (medicine capsules lying horizontally, imprinted with a number)
- carpet (close ups of mesh woven of grey fibres)
- grid (close ups of fine metal grids made of corrugated grey wires)
- hazelnut (close ups of hazelnuts in different orientations)
- leather (close ups of brown leather, showing the lines of the hide structure)
- metal_nut (metal nuts photographed from above, with four wings in different orientations)
- pill (oval medicine pills, white with reddish sprinkles an the letters ‘FF’ imprinted)
- screw (short metal screws with triangular head lying in different orientations)
- tile (close ups of dark grey spotty marble patterns on light grey background)
- toothbrush (oval toothbrush heads photographed from above, bristles in different colors facing up)
- transistor (single black square transistors, soldered to a conductor board with three conductors)
- wood (close ups of light brown wood with dark brown wood grain)
- zipper (close ups of closed vertical black zippers in strips of black cloth)

For each category, images are grouped into:

- train (images classified as 'good', considered not containing anomalies)
- test (images classified as 'good', as well as images containing different types of anomalies, specific to each category)
- ground_truth (one image corresponding to each anomalous original image from 'train'; all areas not containing the anomaly in the original are uniformly black, all areas depicting the anomaly in the original are uniformly white)

	category	Train Normal Images	Test Normal Images	Test Anomalous Images
0	bottle	209	20	63
1	cable	224	58	92
2	capsule	219	23	109
3	carpet	280	28	89
4	grid	264	21	57
5	hazelnut	391	40	70
6	leather	245	32	92
7	metal_nut	220	22	93
8	pill	267	26	141
9	screw	320	41	119
10	tile	230	33	84
11	toothbrush	60	12	30
12	transistor	213	60	40
13	wood	247	19	60
14	zipper	240	32	119

The above table shows the number of images in 'train', as well as the number of normal and anomalous images in 'test' for each category.

	category	number of anomaly types
0	bottle	3
1	cable	8
2	capsule	5
3	carpet	5
4	grid	5
5	hazelnut	4
6	leather	5
7	metal_nut	4
8	pill	7
9	screw	5
10	tile	5
11	toothbrush	1
12	transistor	4
13	wood	5
14	zipper	7

The above table shows the number of different anomaly types distinguished for each category.

Given the partitions described above, the following **folder structure** has to be accessed in order to load the images:

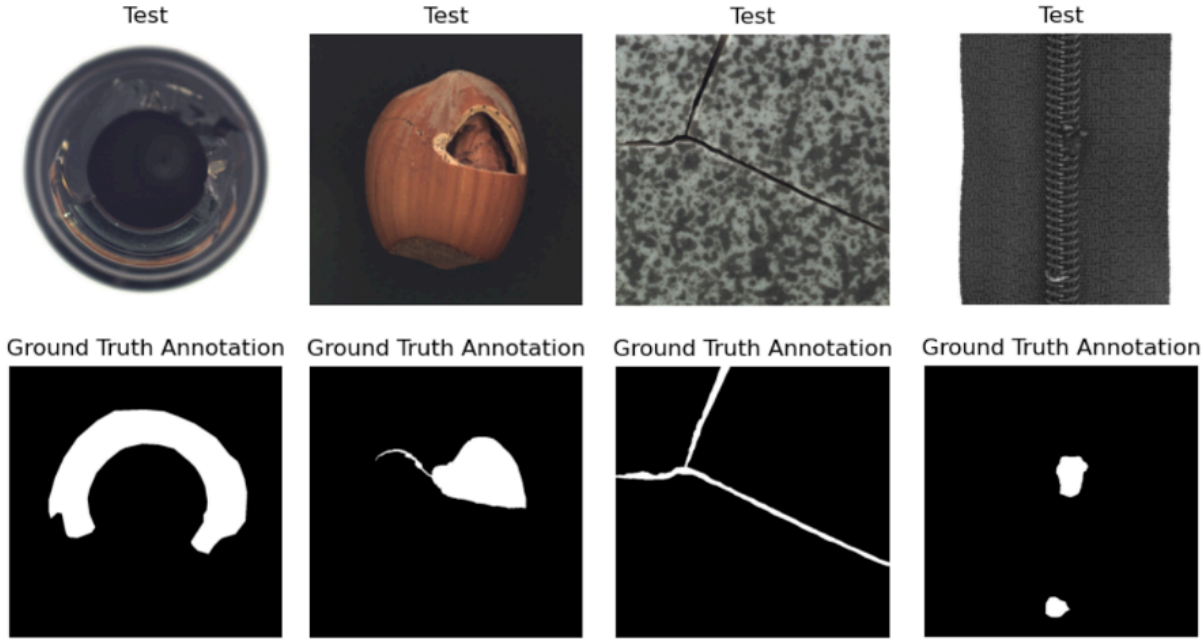
```
<dataset folder>
     <folder 'category 1'>
          <folder 'ground_truth'>
               <folder 'anomaly type 1'>
               <folder 'anomaly type 2'>
               [...]
          <folder 'test'>
               <folder 'good'>
               <folder 'anomaly type 1'>
               <folder 'anomaly type 2'>
               [...]
          <folder 'train'>
               <folder 'good'>
     [...]
     <folder category 15>
          <folder 'ground_truth'>
               <folder 'anomaly type 1'>
               <folder 'anomaly type 2'>
               [...]
          <folder 'test'>
               <folder 'good'>
               <folder 'anomaly type 1'>
               <folder 'anomaly type 2'>
               [...]
          <folder 'train'>
               <folder 'good'>
```

Each individual image is saved as an array with width and height corresponding to the number of pixels, and a depth of three layers (channels), one for color intensities of blue, green and red, respectively.

Images are named using 3-digit consecutive numbers starting at '000', and the extension '.png'. Images in 'ground_truth' have the same number as their corresponding original, with the suffix '_mask' added to it.

As numbering starts at '000' in every folder, during data loading it has to be made sure that while images are being handled, they always are correctly tagged with their provenance, in order to avoid confusion.

Sample Images



Anomalous test images of categories 'bottle', 'hazelnut', 'tile' and 'zipper', and their corresponding ground truth image.

Dataset Preparation

Images are grouped into individual subsets for each category. As they are already sorted into ‘normal’ images without anomalies as well as images showing different types of anomalies, a common train-test-split is not necessary and could only be applied in order to change the portion of ‘normal’ images in the test-sets.
Loading is done using the PyTorch library. The `torchvision.io.read_image()` method returns a 3-dimensional PyTorch Tensor with dimensions being `(3, image_height, image_width)`, where ‘3’ is the number of color channels blue, green and red. Individual images are loaded into a custom object `ImageDataset`, which is derived from the `torch.utils.data.Dataset` class. The `torch.utils.data.DataLoader` class provides an iterable to access the images in the dataset. Thus, images can be efficiently loaded, handled and processed in batches.
The methods `torch.save()` and `torch.load()` are available to store and load whole Datasets to/from disk as .pt-files, using python’s built-in pickle module for serialization.

Dataset Analysis

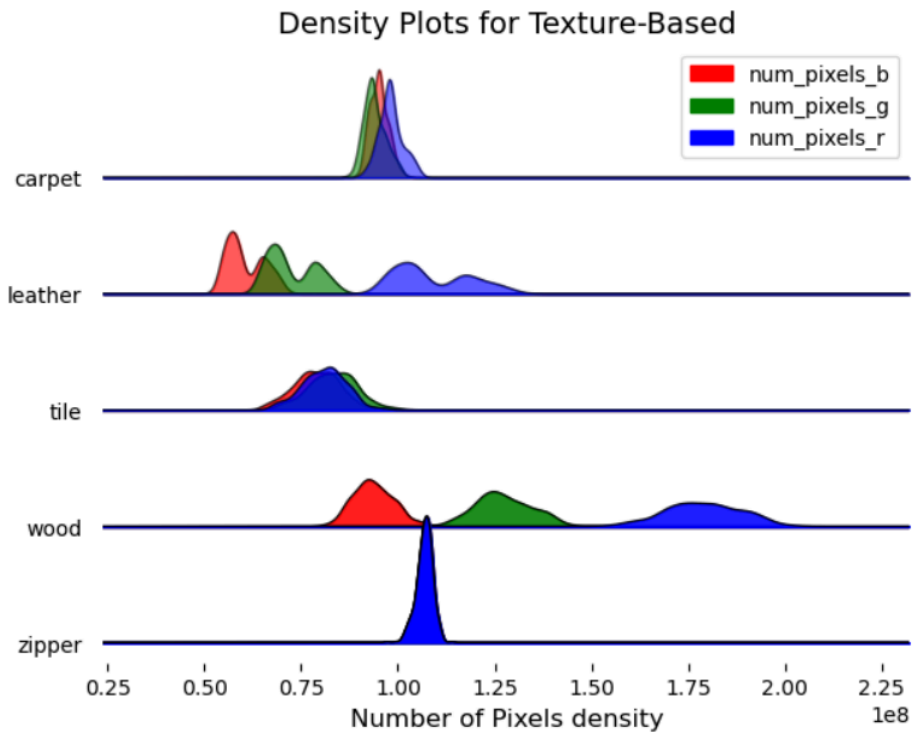
The following table summarizes various features that have been extracted across all object categories, showing image dimensions (width and height in pixel) along with average properties like absolute anomaly area, anomaly area percentage, perceived brightness, and contrast. Most categories have a standard resolution of 1024x1024, while few categories such as “metal_nut”, “pill” and “tile” have smaller dimensions. Image dimensions within the same category are constant.

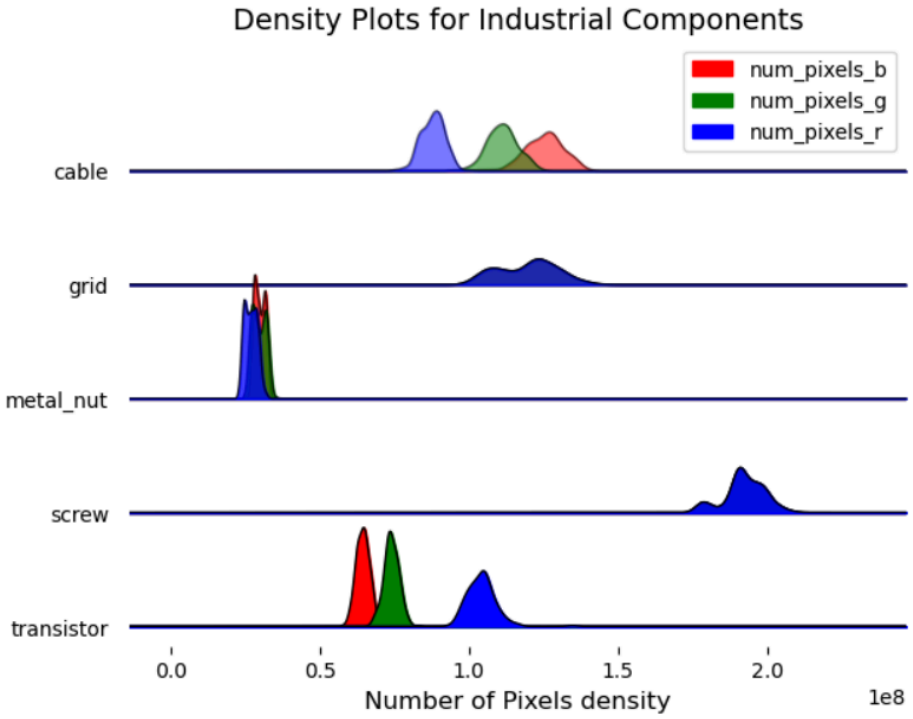
Average anomaly areas and visual features vary across categories. For example, “metal_nut“ shows the largest average anomaly area, 14.5%, while for ‘screw’ tiny anomalies of, on average, 0.3% of image size have to be detected. Contrast varies from an average 33 on ‘leather’ up to an average 275 on ‘bottle’, and perceived brightness from an average 47 on ‘toothbrush’ to an average 183 on ‘screw’.

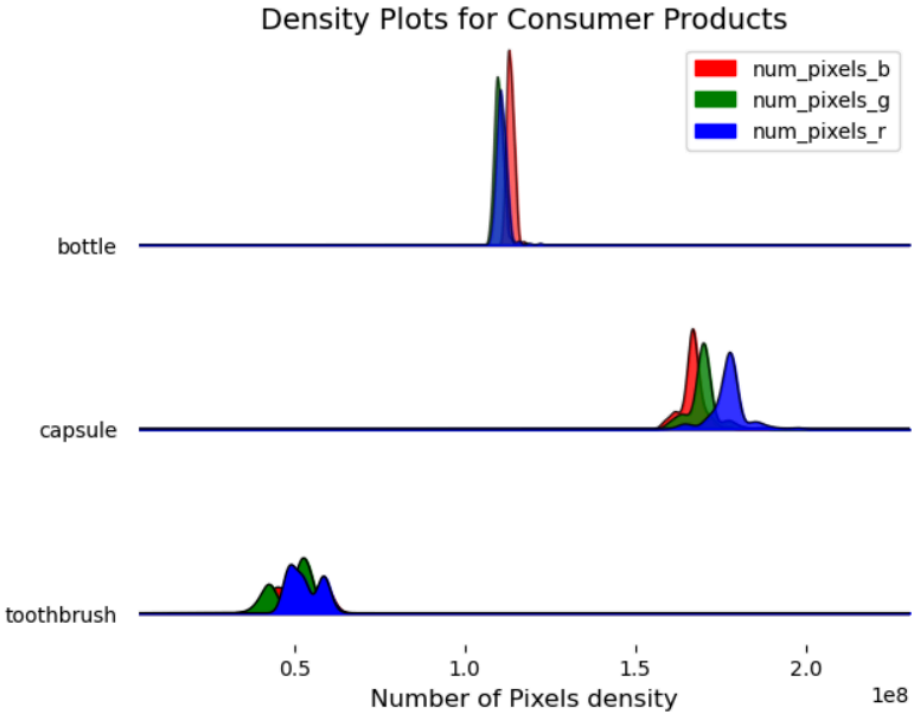
	Category	Width (pixel)	Height (pixel)	Avg_defect_area [abs]	Avg_anomaly_area_percentage %	Avg_perceived_brightness	Avg_contrast
0	bottle	900	900	61694	7.6	136	275
1	cable	1024	1024	49085	4.7	102	154
2	capsule	1000	1000	11088	1.1	171	192
3	carpet	1024	1024	22087	2.1	90	104
4	grid	1024	1024	9911	0.9	114	117
5	hazelnut	1024	1024	35175	3.4	49	71
6	leather	1024	1024	9166	0.9	75	33
7	metal_nut	700	700	70995	14.5	59	123
8	pill	800	800	25455	4.0	77	227
9	screw	1024	1024	3517	0.3	183	100
10	tile	840	840	69164	9.8	116	93
11	toothbrush	1024	1024	22449	2.1	47	160
12	transistor	1024	1024	125650	12.0	76	119
13	wood	1024	1024	53323	5.1	129	45
14	zipper	1024	1024	27515	2.6	101	246

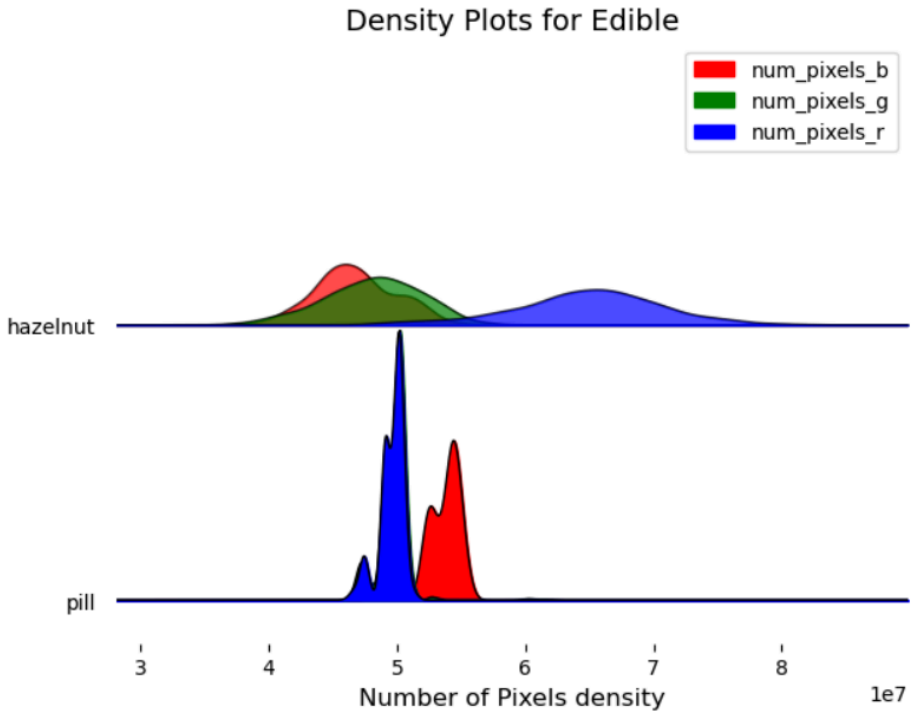
BGR Density Plots visualize the pixel intensity distributions for blue, green and red channels. The following graphs show BGR Density Plots for each category, grouped into the subcategories

- texture-based (carpet, leather, tile, wood, zipper)
- industrial components (cable, grid, metal_nut, screw, transistor)
- consumer product (bottle, capsule, toothbrush)
- edible (hazelnut, pill)




Density Plots for Industrial Components
num_pixels_b
num_pixels_g
num_pixels_r
cable
grid
metal_nut
screw
transistor
0.0
0.5
1.0
1.5
2.0
Number of Pixels density
1e8

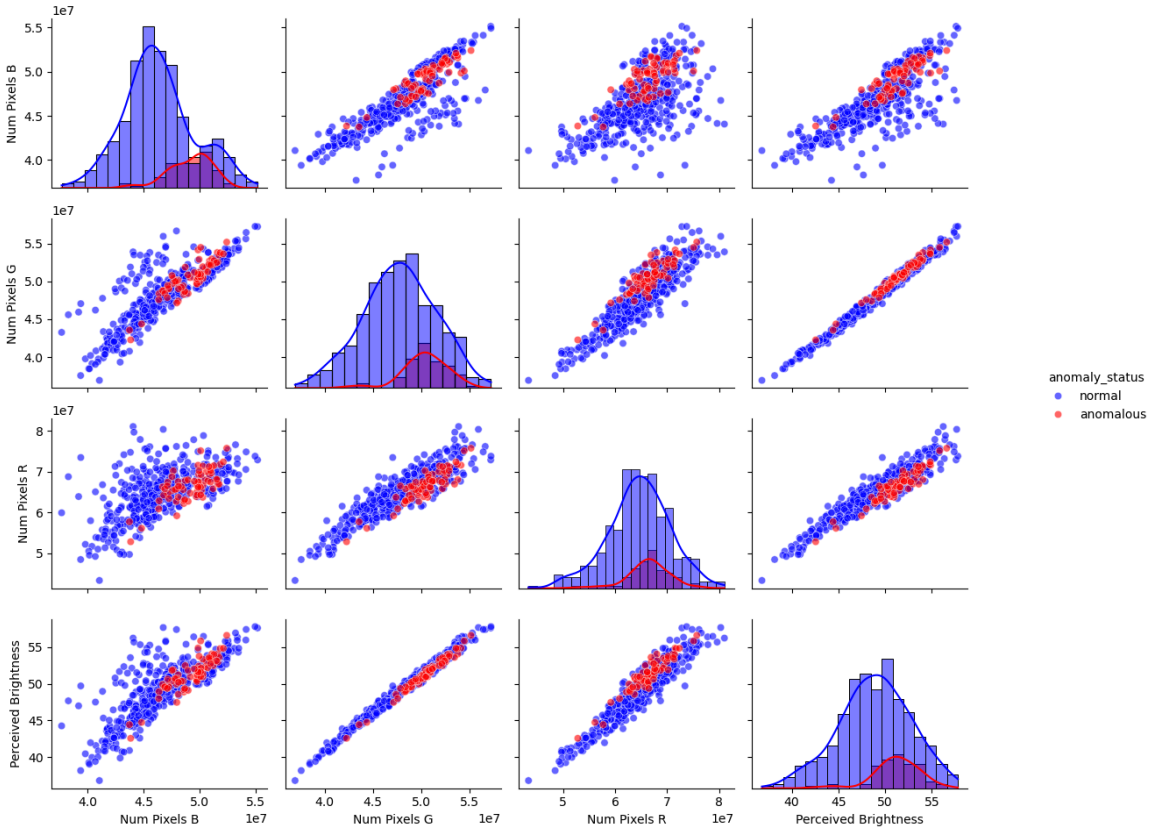

Density Plots for Consumer Products
num_pixels_b
num_pixels_g
num_pixels_r
bottle
capsule
toothbrush
0.5
1.0
1.5
2.0
Number of Pixels density
1e8



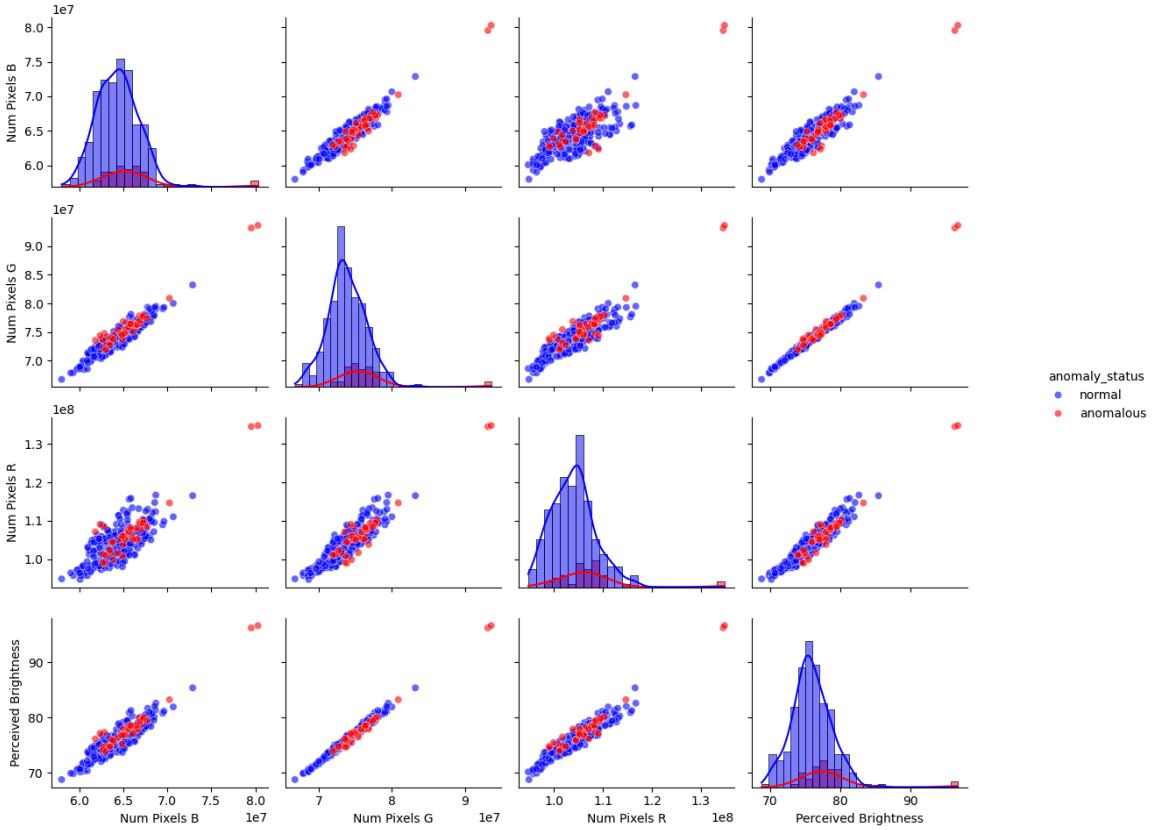
The analysis shows that the more the distributions for the different channels are overlapping, the closer the images are to grayscale, with three categories ('zipper', 'screw' and 'grid') being completely grayscale.
This hints at potential redundancies that can be used for dimensionality reduction in some of the categories.

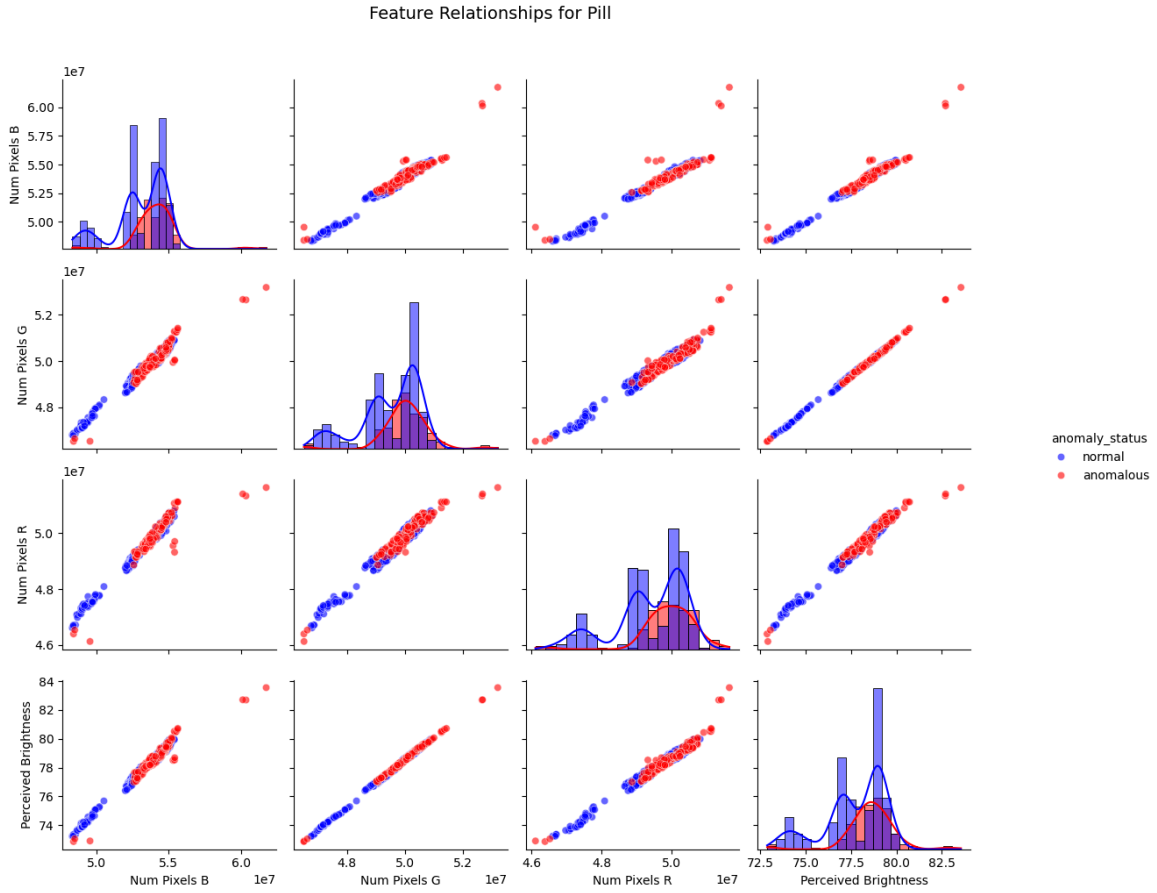
Scatter plots provide a clear visual representation of how two variables relate to each other, and are therefore effective in identifying correlations, trends and potential outliers in data.
The following graphs show **scatter plots of pixel counts for each channel and perceived brightness**. Color-coding differentiates between normal (red) and anomalous (blue) data.
The analysis shows that for several categories, a clear clustering of anomalous data can be detected.

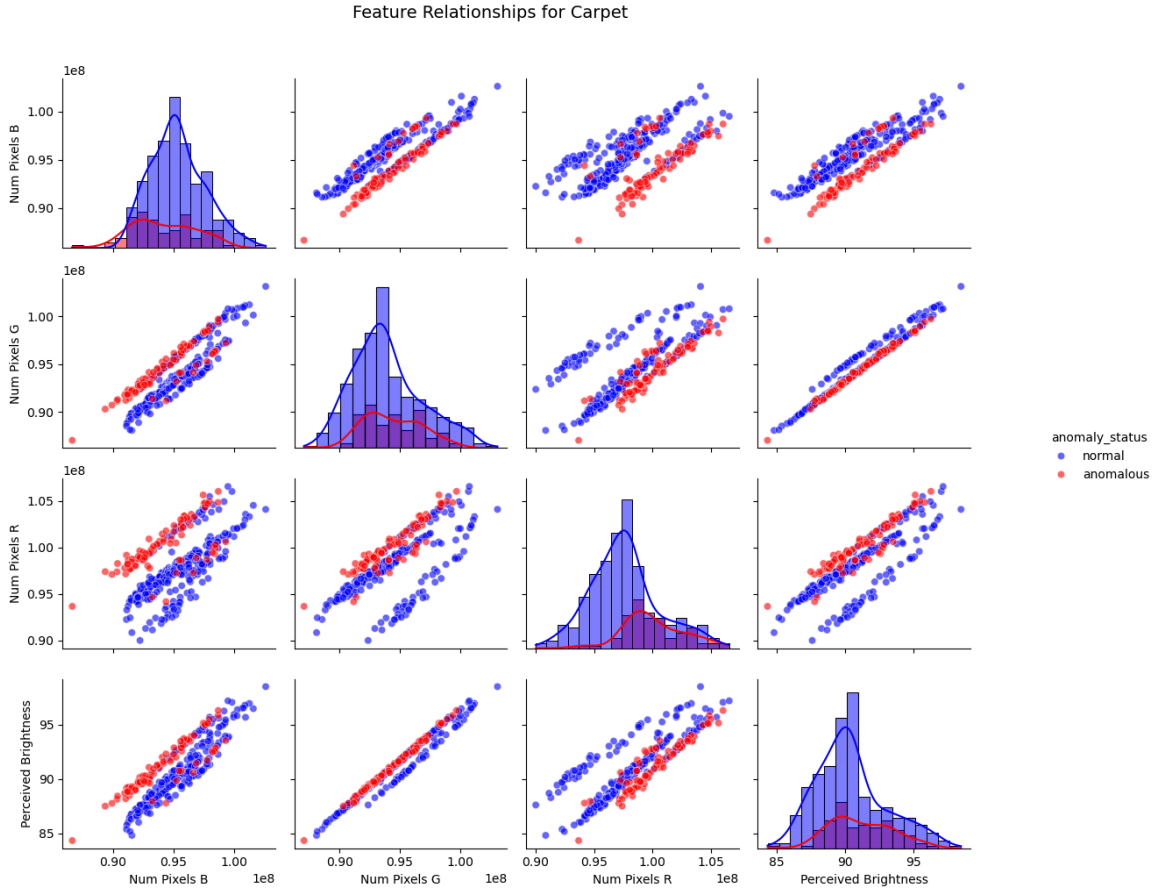
Feature Relationships for Hazelnut



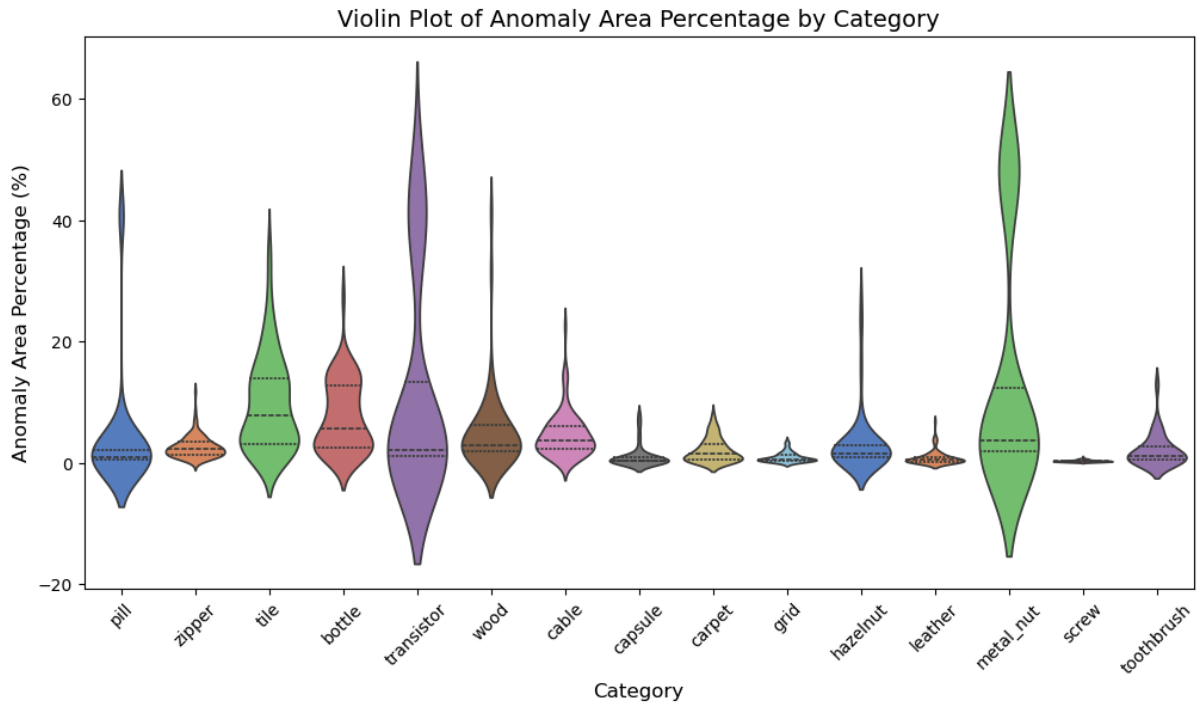
Feature Relationships for Transistor




Feature Relationships for Pill
Num Pixels B
Num Pixels G
Num Pixels R
Perceived Brightness
1e7
anomaly_status
normal
anomalous

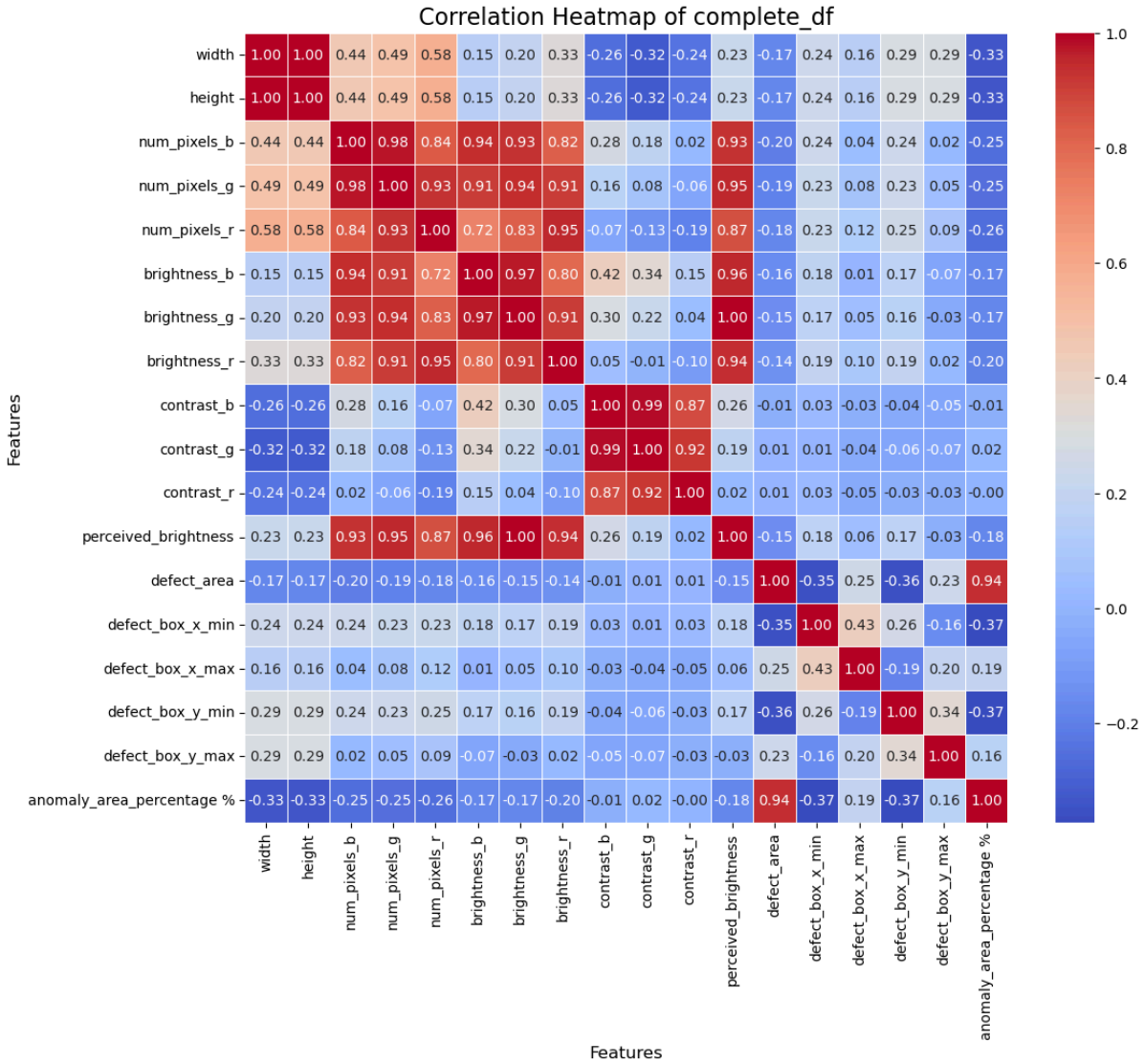

Feature Relationships for Carpet
Num Pixels B
Num Pixels G
Num Pixels R
Perceived Brightness
1e8
anomaly_status
normal
anomalous

Violin Plots of Anomaly Area Percentages show how sizes of anomalous areas within images are distributed throughout categories. This provides insights into the typical sizes of anomalies per category.



The analysis shows that some categories, like 'screw' or 'grid' have only very small anomalies with little variation, while others, especially 'metal_nut' and 'transistor' have large numbers of extreme values towards big anomaly areas. Understanding such variability in anomaly sizes is important for developing robust detection models.

Like scatterplots, a **heatmap** helps to identify correlations between different features of the dataset and provides insight into which features are most valuable and where there are redundancies.
The analysis shows that strong correlations exist among brightness values across all channels. Also, contrast across channels is negatively correlated to width and height of the image.



Roadmap for Anomaly Detection (Unsupervised model)

Step 1: Define Objectives

- Clearly define the goals of the anomaly detection project, including:
    - The objective is to determine the classifier's ability to distinguish between normal and anomalous samples effectively and robustly. Since the approach is unsupervised, the model must learn patterns of normalcy autonomously, relying solely on normal sample data without labeled anomalies.
    - Desired performance metrics (e.g., AUROC, precision-recall, F1-score). AUROC is often considered the most robust metric as it evaluates model performance across all classification thresholds, providing a comprehensive view of both sensitivity and specificity.
    - Resource constraints (e.g., computational power, time). For computational needs, either use CUDA drivers for local GPU acceleration or opt for Google Colab as a cloud-based solution to manage hardware limitations effectively.

Step 2: Data Preparation

- Collect and preprocess the data:
    - Leverage the MVTec dataset, which already provides high-quality normal samples for training. The specific preprocessing steps may depend on the model being used; for instance, models requiring deep learning often benefit from resizing and normalization, whereas traditional models may need engineered features.
    - Apply preprocessing techniques such as normalization, resizing, or augmentation.

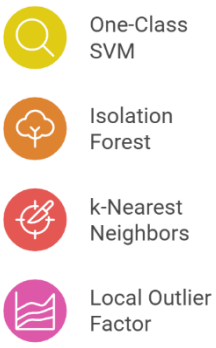
Step 3: Feature Extraction

- Choose effective feature extraction methods:
    - Start by applying custom feature extraction strategies, including brightness, contrast, and anomaly area calculations, which have already been implemented and tailored specifically to this dataset.
    - Use pre-trained models like ResNet for deep features. ResNet (Residual Network) is a widely-used deep learning architecture designed to address the vanishing gradient problem in very deep networks. It uses residual blocks, which allow the model to learn residual functions relative to the input rather than trying to learn unreferenced functions. This architecture is particularly effective for extracting hierarchical features, making it highly suitable for tasks like anomaly detection by leveraging its ability to capture both local and global patterns in data.

Step 4: Model Selection and Training

Option A: Traditional Machine Learning Models

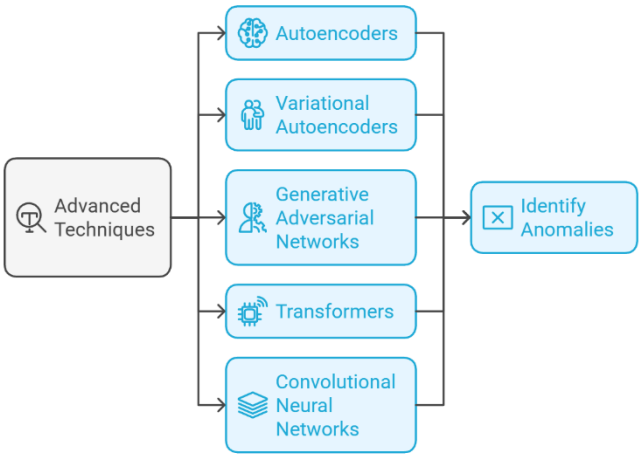
- Experiment with at least one model, with and without ResNet feature extraction:
    - **One-Class SVM (OCSVM)**: A robust model for identifying anomalies in high-dimensional feature spaces by learning a decision boundary around normal data and classifying points outside it as anomalous. Optimize hyperparameters such as kernel type, gamma, and nu to enhance performance.
    - **Isolation Forest:** Detects anomalies by isolating individual data points using random partitioning, making it efficient for large datasets and highly interpretable. Tune hyperparameters like the number of estimators and maximum features to improve detection accuracy.
    - **k-Nearest Neighbors (KNN):** A distance-based approach that identifies anomalies as points far from their nearest neighbors, well-suited for scenarios with well-defined local patterns. Adjust the number of neighbors and experiment with different distance metrics for better results.
    - **Local Outlier Factor (LOF):** A density-based method that identifies anomalies by comparing the local density of a data point to that of its neighbors. Points with significantly lower density than their neighbors are considered anomalies. Tune parameters like the number of neighbors to balance sensitivity and specificity.



Option B: Deep Learning Models

- Explore advanced techniques (at least one) with ResNet features extraction:
    - **Autoencoders:** Employ reconstruction errors to pinpoint anomalies by training the model to recreate input data, enabling the identification of unusual regions as high-error areas.
        - **MSE L2 loss without ResNet:** The mean squared error (MSE) loss measures pixel-level reconstruction differences, effectively flagging regions with unusual patterns when the autoencoder is trained without feature extraction from ResNet.

        - **Extracting ResNet latent features:** ResNet latent features can be integrated with the autoencoder to provide a richer representation of the input, enhancing the model's ability to detect subtle anomalies by leveraging hierarchical and contextual features.
    - **Variational Autoencoders (VAEs):** Extend autoencoders by modeling the data distribution through a latent space representation. This allows anomalies to be detected as samples with low likelihood under the normal data distribution. Using beta-VAE, which introduces a regularization factor to balance reconstruction and latent space disentanglement, could improve anomaly detection by enhancing feature separation and interpretability.
    - **Generative Adversarial Networks (GANs):** Leverage adversarial training where a generator produces data and a discriminator identifies anomalies based on reconstruction or classification errors.
    - **Transformers:** Utilize their capacity to capture long-range dependencies and global context for improved anomaly detection and localization.
    - **Convolutional Neural Networks (CNNs):** Extract hierarchical features from input data. When combined with autoencoders, CNNs can enhance anomaly detection by providing detailed spatial representations for more accurate reconstruction and anomaly characterization.



Step 5: Training Setup Options

- Consider different scenarios:

    - **Unsupervised Learning:** Train using normal samples only. As a rule of thumb, split the training dataset, which consists of normal pictures, into training and validation subsets. Then, test the model on a test dataset containing both normal and anomalous data to evaluate its performance.
    - **Transfer Learning:** Fine-tune pre-trained feature extractors like ResNet. ResNet is particularly advantageous for this task because it captures both local and global patterns in data using residual blocks, ensuring robust feature extraction for anomaly detection.

- **Multi-Scale Representations:** Aggregate features at multiple scales for enhanced localization.

Step 6: Evaluate and Compare Models

- Establish performance metrics:
  - **Detection Metrics:** Evaluate model performance using image-level AUROC (Area Under the Receiver Operating Characteristic curve). AUROC provides a comprehensive measure of a model's ability to distinguish between normal and anomalous data across various thresholds.
- Visualize results:
  - Generate ROC curves to illustrate the trade-offs between true positive and false positive rates at different thresholds.
  - Create heatmaps and bar charts to provide a visual comparison of model performance across datasets and metrics.
  - Highlight anomaly localization accuracy with overlays by comparing model predictions against ground truth anomaly maps, helping to assess spatial precision and accuracy.