Master Thesis

Anomaly Detection with Pre-trained CNN Features: A Comparative Study of Classical Methods

Sadia Mahjabin
Student ID: 229913

January 24, 2026

Supervisors:
Prof. Dr. Emmanuel Müller
Dr. Simon Klüttermann

Abstract

Anomaly detection aims to identify samples that deviate from normal data distributions and is particularly challenging in high-dimensional visual domains. While recent approaches often rely on fully end-to-end deep learning models, classical one-class anomaly detection methods remain attractive due to their simplicity, interpretability, and low computational cost. This thesis investigates the extent to which classical anomaly detection algorithms can be effectively combined with deep feature representations for visual anomaly detection.

A comprehensive experimental study is conducted using multiple image datasets of increasing complexity, including MNIST, Fashion-MNIST, CIFAR-100, and Tiny-ImageNet. Features are extracted from a fine-tuned ResNet50 model and used as input to classical anomaly detectors, including Principal Component Analysis, One-Class Support Vector Machine, Local Outlier Factor, and Isolation Forest. Performance is evaluated using the area under the ROC curve (AUROC). In addition to the main experiments, further analyses examine the impact of feature extraction depth, multi-layer feature aggregation, feature normalization strategies, and training-free anomaly detection using PatchCore.

The results demonstrate that deep feature representations substantially improve the effectiveness and stability of classical anomaly detection methods compared to operating on raw input representations. The optimal feature extraction layer is shown to depend on dataset complexity, with intermediate convolutional layers providing the most reliable performance on complex natural image datasets. Feature normalization plays a critical role, with standard normalization offering the most consistent results across methods and datasets. While PatchCore achieves strong performance without dataset-specific fine-tuning, feature-based methods using task-adapted representations generally provide more stable performance under constrained computational settings.

Overall, this thesis shows that the quality and structure of feature representations are the primary drivers of anomaly detection performance, enabling classical one-class methods to remain competitive and effective across diverse visual datasets.

Contents

1 Introduction

1.1 Motivation and Problem Statement

Anomaly detection is a fundamental problem in machine learning, concerned with identifying patterns or instances that deviate significantly from what is considered normal. In visual domains, anomaly detection plays a critical role in applications such as industrial inspection, medical imaging, surveillance, and autonomous systems, where abnormal events are rare, diverse, and often difficult to label exhaustively [10].

A central challenge in image-based anomaly detection is the scarcity or complete absence of labeled anomalous data. In many real-world scenarios, abnormal samples occur infrequently, evolve, or correspond to previously unseen failure modes. As a result, supervised classification approaches that rely on representative samples from all classes are often impractical. One-class and semi-supervised methods, which learn a model of normality using only normal data and detect deviations at test time, therefore provide a more realistic formulation of the anomaly detection problem [43, 10].

Classical anomaly detection algorithms, such as Principal Component Analysis (PCA), One-Class Support Vector Machines (OCSVM), Local Outlier Factor (LOF), and Isolation Forest (IForest), are among the most widely used approaches in this setting. PCA-based methods assume that normal data lies near a low-dimensional subspace and identify anomalies based on reconstruction error [25, 45]. OCSVM learns a decision boundary enclosing normal data in feature space and classifies samples outside this boundary as anomalous [43]. LOF identifies anomalies based on deviations in local data density [8], while Isolation Forest isolates anomalous samples through randomized partitioning of the feature space [30]. These methods are well understood, computationally efficient, and relatively interpretable compared to complex deep learning models.

However, when applied directly to raw pixel data, the performance of classical anomaly detectors on images is often poor. Raw pixel representations are high-dimensional, sensitive to noise, and lack semantic structure, making it difficult for simple statistical or geometric models to capture meaningful notions of normality. This limitation has motivated the use of learned feature representations that transform images into more compact and semantically meaningful spaces before anomaly detection.

Convolutional Neural Networks (CNNs) have become the dominant approach for visual representation learning due to their ability to automatically extract hierarchical features from images [27]. Lower layers of CNNs typically capture local patterns such as edges and textures, while deeper layers encode increasingly abstract and semantic information. CNN architectures pretrained on large-scale datasets such as ImageNet have been shown to learn representations that generalize well across a wide range of visual tasks [13].

Transfer learning enables these pretrained representations to be reused effectively even when labeled data in the target domain is limited [35]. In many applications, a pretrained CNN can serve as a fixed or fine-tuned feature extractor, providing a powerful embedding space for downstream tasks. While fully deep learning–based anomaly detection models exist, they often introduce significant computational overhead, architectural complexity, and reduced interpretability, which can limit their practical applicability in resource-constrained or safety-critical settings.

This thesis investigates whether classical anomaly detection methods can achieve strong and stable performance when paired with deep, pretrained CNN feature extractors. Rather than proposing new detectors or learning objectives, the focus is on understanding the role of feature representation quality and evaluating how far simple, well-established methods such as PCA, OCSVM, LOF or IForest can be pushed when operating in a semantically meaningful embedding space. By systematically evaluating performance across datasets of increasing visual complexity, this work aims to clarify the conditions under which classical methods remain effective for visual anomaly detection.

1.2 Research Questions

The following research questions guide the work in this thesis:

- How does feature extraction using pretrained CNNs affect the performance of classical one-class anomaly detection methods on image data?
- Do performance trends observed on simple datasets (e.g., MNIST) generalize to more complex and fine-grained datasets such as CIFAR-100 and Tiny-ImageNet?
- How stable is the performance of classical one-class anomaly detection methods across different normal-anomaly class-pair selections when operating on deep CNN feature representations?

1.3 Contributions

The main contributions of this thesis are summarized as follows:

- A systematic evaluation of classical one-class anomaly detection methods applied to deep CNN feature representations.
- A comparative study of feature-based versus raw-input anomaly detection across datasets of increasing visual complexity.
- An empirical analysis demonstrating that representation learning plays a dominant role in anomaly detection performance, often outweighing detector complexity.
- Evidence that simple and interpretable methods such as PCA, OCSVM, LOF, and Isolation Forest can achieve near-optimal performance when paired with pretrained, fine-tuned ResNet50 features.

1.4 Thesis Structure

The remainder of this thesis is organized as follows:

1. Chapter 2 reviews related work in image classification, transfer learning, and anomaly detection.
2. Chapter 3 introduces the necessary background on convolutional and residual neural networks.
3. Chapter 4 presents anomaly detection paradigms and the classical methods used in this work.
4. Chapter 5 describes the datasets, problem formulation, and evaluation protocol.
5. Chapter 6 reports initial experiments using a lightweight CNN.
6. Chapter 7 presents the main experiments using a ResNet-50 backbone.
7. Chapter 8 presents additional experiments that complement the main study and provide further insight into representation-related design choices in anomaly detection.
8. Chapter 9 concludes the thesis and outlines limitations and directions for future work.

2 Related Work

This chapter reviews prior research that is directly related to the methodology adopted in this thesis. The focus is on classical anomaly detection methods, their limitations on raw image data, and prior work that combines these methods with deep convolutional feature representations.

Classical Anomaly Detection Methods

Early work on anomaly detection has focused on unsupervised and one-class learning settings, where anomalous examples are unavailable or poorly defined. Chandola et al. provided a comprehensive survey of anomaly detection methods and highlighted PCA- and SVM-based approaches as foundational techniques in one-class learning scenarios [10].

Jolliffe et al. introduced Principal Component Analysis (PCA) as a linear projection method for modeling dominant variance in data [25]. Building on this, Shyu et al. applied PCA to novelty detection by modeling normal data in a low-dimensional subspace and using reconstruction error as an anomaly score [45]. Their results demonstrated that PCA can effectively detect anomalies when normal data exhibits strong global correlations.

Schölkopf et al. proposed the One-Class Support Vector Machine (OCSVM), which estimates the support of a high-dimensional distribution by learning a boundary around normal data [43]. They showed that OCSVM is well-suited for anomaly detection in cases where only normal samples are available for training. Subsequent studies reported that OCSVM performance is highly sensitive to feature representation and scaling, particularly in high-dimensional settings such as image data.

Breunig et al. introduced the Local Outlier Factor (LOF) as a density-based anomaly detection method that identifies outliers by comparing the local density of a data point to that of its nearest neighbors [8]. Unlike global methods such as PCA and OCSVM, LOF focuses on local neighborhoods and assigns higher anomaly scores to samples that reside in regions of significantly lower density relative to surrounding data. This locality-sensitive formulation enables LOF to detect subtle, context-dependent anomalies that may not be apparent under global distributional assumptions. However, subsequent studies have shown that LOF is sensitive to the choice of neighborhood size and to feature scaling, and its performance can degrade in high-dimensional spaces due to the curse of dimensionality.

Liu et al. proposed Isolation Forest (IForest) as an ensemble-based anomaly detection method that explicitly isolates anomalies rather than modeling normal data density or decision boundaries [30]. IForest operates by constructing random partitioning trees in

which anomalous samples tend to be isolated in fewer splits than normal instances. This randomized isolation mechanism allows IForest to scale efficiently to high-dimensional data and large datasets, making it attractive for practical applications. Nevertheless, like other classical detectors, its effectiveness on image data has been shown to depend strongly on the quality of the underlying feature representation, with raw pixel inputs often yielding suboptimal results.

Other widely studied approaches include statistical methods such as Gaussian mixture models [5], distance-based k-nearest neighbor outlier detection [12], and clustering-based techniques that identify anomalies as points not belonging to any cluster [24]. These methods have shown effectiveness in tabular and low-dimensional settings, but their performance often degrades in high-dimensional image spaces due to distance concentration effects and increased sensitivity to noise [2].

Although classical anomaly detection methods differ in their underlying assumptions, many studies report substantial performance degradation when these approaches are applied directly to raw image pixels. This limitation is commonly attributed to the high dimensionality of image space and the lack of explicit semantic structure, motivating the use of learned feature representations for visual anomaly detection [10].

CNN-Based Feature Representation

LeCun et al. demonstrated that convolutional neural networks (CNNs) can learn hierarchical representations from images, enabling effective feature learning directly from pixel data [27]. Their work laid the foundation for modern CNN-based visual representation learning.

He et al. introduced Residual Networks (ResNets), showing that residual connections allow very deep networks to be trained efficiently [21]. Using ImageNet benchmarks, they demonstrated that ResNet architectures achieve strong generalization and produce highly discriminative feature representations. As a result, ResNet models, including ResNet50, have since been widely adopted as pretrained feature extractors for downstream vision tasks.

Transfer Learning for Visual Tasks

Deng et al. introduced the ImageNet dataset and showed that CNNs trained on large-scale labeled data learn transferable visual representations [13]. Building on this, Pan and Yang provided a survey of transfer learning methods and demonstrated that pretrained models can be effectively reused in data-scarce target domains [35].

More recently, the success of large visual models trained on massive datasets has further reinforced the importance of representation learning in computer vision. Such models capture generic visual features that can be applied to a wide range of downstream tasks without task-specific retraining. In practical applications, including industrial inspection[4], medical imaging[44], remote sensing[55], and surveillance, pretrained CNNs are commonly used as fixed or fine-tuned feature extractors to reduce annotation cost and improve robustness.

Across these application domains, multiple studies have shown that leveraging pre-trained visual representations leads to improved performance and faster convergence compared to training models from scratch, particularly when labeled data is limited. These findings motivate the use of pre-trained CNN backbones as generic feature extractors in feature-based anomaly detection pipelines.

Feature-Based Anomaly Detection in Images

Several prior works have combined classical anomaly detection methods with CNN-derived feature representations. These studies consistently report that operating in a learned feature space substantially improves anomaly separability compared to raw pixel inputs.

In particular, PCA has been applied to CNN features for image anomaly detection, where reconstruction error in feature space provides a more meaningful anomaly score than pixel-space reconstruction. Similarly, OCSVM has been used on CNN embeddings to learn decision boundaries around normal samples, achieving improved robustness and stability compared to raw-input baselines [10].

These feature-based approaches preserve the interpretability and simplicity of classical detectors while benefiting from the representational power of deep neural networks, and they are commonly used as strong baselines in visual anomaly detection studies.

Positioning of This Work

This thesis follows the feature-based anomaly detection paradigm established by prior research. Rather than introducing new anomaly detection algorithms, the goal is to systematically evaluate a set of classical one-class methods, PCA, OCSVM, LOF, and IForest, when applied to features extracted from pretrained and fine-tuned CNNs.

By comparing performance with and without feature extraction across datasets of increasing visual complexity, this work extends prior studies by providing a controlled empirical analysis of the role of representation quality. The results aim to reinforce existing findings that deep pretrained features, such as those produced by ResNet50, are a key factor enabling reliable anomaly detection with simple and interpretable classical methods.

3 Image Classification & Deep Neural Networks

3.1 Image Classification

For human eyes, it is a primary ability to see the differences between images of different classes; for example, humans can easily see whether a picture contains a cat, a car, or a tree. However, for computers, it is not that straightforward. A computer only sees numbers that represent pixel values, so researchers in computer vision have worked for decades to build algorithms that can interpret these numbers in meaningful ways. Their research includes object detection to locate instances of semantic objects of a particular class, object recognition to determine whether image data contains a specific object, and scene understanding to parse an image into meaningful segments for analysis [17].

Image classification is the process of assigning every image in a dataset to one of several predefined categories. Typically, the dataset is split into training and testing sets. Each data instance is represented as a pair (x, y), where x denotes the input features and y denotes the class label (also known as target attribute or category, or the class the data belongs to). A classification model learns patterns from the training data, and once trained, it can assign class labels to new, unseen images

Before the rise of deep learning, the progress of image classification heavily relied on well-designed features combined with classical machine learning methods, e.g., Support Vector Machines (SVMs) [11], k-Nearest Neighbors (k-NN) [12], Random Forests [7], and Naive Bayes classifiers [31]. These methods were effective on small-sized datasets, e.g., CIFAR-10 [26] and MNIST [27], but they require significant manual feature design and often struggle with complex, real-world images.

The emergence of deep learning dramatically changed this landscape. Deep learning models, especially Convolutional Neural Networks (CNNs) [27], automatically learn meaningful features directly from raw images, eliminating the need for handcrafted descriptors. CNNs scale effectively to large, real-world datasets such as ImageNet [13] and deliver far superior accuracy.

As researchers built deeper CNNs, they encountered a new difficulty: very deep models were hard to train, and adding more layers sometimes reduced performance. Residual Networks (ResNets) [21] solved this problem by introducing skip connections, which allow information and gradients to flow through the network more easily. This innovation enabled the successful training of extremely deep models—50, 101, or even 152 layers—without suffering from degradation. Because of this breakthrough, ResNet has become one of the most influential architectures in modern computer vision.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are one of the most widely used deep learning models for image classification and feature learning. A CNN is typically composed of three main types of layers: convolutional layers, pooling layers and fully connected layers. These components work together to extract visual patterns from images and make predictions. Figure 3.1 shows a schematic illustration of the layers of a basic CNN.

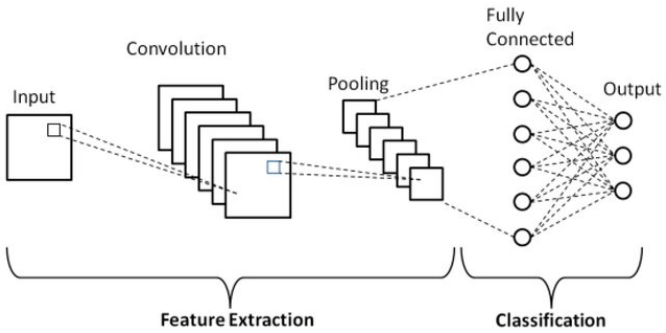

Figure 3.1: Schematic diagram of a basic convolutional neural network (CNN) architecture [37]

Convolutional Layer

The convolutional layer is the core building block of a CNN. Its main purpose is to extract features—such as edges, textures, or shapes—from the input image. A convolutional layer applies several learnable filters (kernels) over the image. Each filter slides across the image and performs an element-wise multiplication with the overlapping patch, producing a feature map. If the layer contains N filters, it will generate N feature maps [49].

A number of significant parameters constitute the convolutional layer. The size of the filter determines the area over which a convolution is carried out. Several channels are allowed in the convolutional layer, and the number of filters defines this. The stride is defined as the step length by which the filter moves in the image. The convolutional layer has the functionality to capture the feature maps using various sizes of kernels and pool the feature maps. Thus, it is able to minimize the number of connections between the convolutional and pooling layer [34].

Pooling Layer

Pooling layers reduce the spatial size of the feature maps and keep only the most important information. This helps the network become less sensitive to small image distortions and significantly reduces computational cost.

Common pooling strategies include:

- Max pooling - selects the maximum value in each region [6, 42].
- Average pooling - computes the average value [28].
- Stochastic pooling - samples a value based on local probabilities [54].
- Spatial pyramid pooling (SPP) - pools at multiple scales [20].

Among the above strategies, max pooling is widely used because it is particularly robust to local variations in images.

Fully Connected Layer

After the convolutional and pooling layers extract and condense the visual features, the fully connected (FC) layer interprets these features to perform the final prediction. In this layer, every neuron is connected to all neurons from the previous layer, enabling the network to learn complex decision boundaries [53]. The FC layer typically uses activation functions such as ReLU, Sigmoid, Softmax, or Tanh to introduce non-linearity and enable the model to generalize effectively.

The convolutional output at layer l for position j in the i-th feature map is expressed as

$$C_{i,j}^{(l)} = \sigma\left(b_j^{(l)} + \sum_{m=1}^{M} w_{m,j}^{(l)}\, x_{i+m-1,\,j}^{(l-1)}\right),$$

In which, the first index l denotes the layer, the second index, σ, is the activation term and the third index, $b_j^{(l)}$, is the bias of the feature map, as well as the weights, $w_{m,j}^{(l)}$, of the convolution kernel applied on the input feature map from the previous layer, $x^{(l-1)}$. In simpler terms, the filter multiplies its weights with a small region of the input, sums these values, adds a bias, and finally applies the activation function. This process is repeated across the entire image to build each feature map.

3.3 Residual Neural Networks

Deep neural networks generally achieve higher accuracy when they use more layers. However, one of the major problems that arises in very deep layers is the vanishing gradient problem [22]. It is a phenomenon in which the gradients become extremely small during backpropagation, which means the previous layer learn very little or not at all. That makes the training of a deep neural network very difficult. One of the solutions to this problem is a shortcut connection or residual connection, which is introduced by Residual Neural Networks [21].

A ResNet is built from residual blocks, each containing several convolutional layers and a skip connection that adds the block's input directly to its output. Instead of learning a full transformation, each block learns only the residual—the difference between the input and the desired output. This structure stabilizes training, improves gradient flow, and makes it feasible to train networks with more than 100 layers [46]. Batch

normalization and ReLU activation functions are also incorporated to ensure stable and efficient learning.

Common ResNet variants include ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, where the number refers to the total depth of the network. Among these, ResNet-50 is widely used in computer vision tasks such as image classification, object detection, and semantic segmentation [21].

ResNet-50 consists of 50 layers and begins with a 7×7 convolution followed by a 3×3 max-pooling layer to reduce the spatial resolution. The network is organized into four stages, each containing multiple bottleneck residual blocks. Each bottleneck block uses a sequence of 1×1, 3×3, and 1×1 convolutions designed to reduce, process, and then restore feature dimensionality efficiently. As the network progresses through the stages, the number of channels increases while the spatial resolution decreases, usually through stride-2 convolutions. Residual connections throughout the network ensure stable gradient flow and enable effective training of deep architectures. Finally, a global average pooling layer followed by a fully connected layer with 1000 output units makes the model suitable for large-scale classification tasks such as ImageNet. Figure 3.2 illustrates the ResNet-50 architecture, highlighting its stacked bottleneck residual blocks with identity shortcuts that enable stable training and effective feature extraction in deep networks.

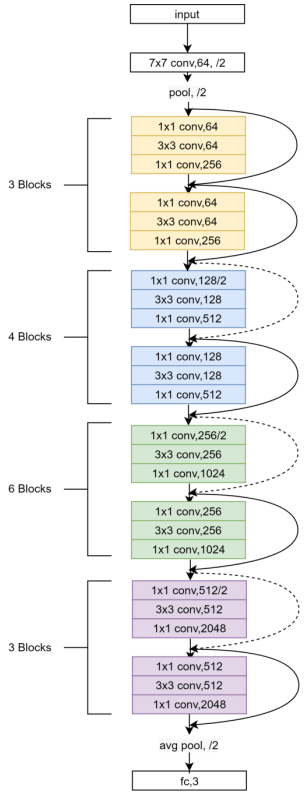

Figure 3.2: Architecture of the ResNet-50 convolutional neural network [3]

3.4 Transfer Learning

In many machine-learning applications, such as classification, regression, or clustering, models perform well only when the training and test data come from the same feature space and follow the same underlying distribution. If this distribution changes, the trained model often becomes unreliable, and rebuilding it from scratch would require collecting a new labeled dataset. In real scenarios, however, collecting and labeling large amounts of high-quality data is costly, time-consuming, and sometimes impossible. In such situations, transfer learning provides a practical solution [35].

The idea behind transfer learning originates from how humans learn. People can solve a new problem more easily when it resembles tasks they have previously encountered. In a similar way, transfer learning allows a model to use knowledge learned from one situation where data is abundant and apply it to another situation where only limited labeled data is available. To describe this formally, a domain consists of a feature space (the type of input data) and a data distribution (how that data is spread). A task consists of a label space and a predictive function that maps input features to the correct labels. The domain from which the model initially learns is called the source domain, indicated by D_S, and the task it learns there is the source task, indicated by T_S. The new situation to which the knowledge is transferred is called the target domain, indicated by D_T, with its corresponding target task, indicated by T_T. Transfer learning refers to improving the predictive function in the target domain by leveraging knowledge from the source domain, even when the feature spaces, data distributions, or tasks are not the same, e.g, when $D_S \neq D_T$ or $T_S \neq T_T$ [35]. Transfer becomes possible when the source and target domains share some meaningful relationship, either explicit or implicit, so that knowledge learned in one environment can support learning in another.

Transfer learning has been successfully applied in many areas, including text sentiment analysis [50], image classification [15], human activity recognition [18], software defect prediction [33], and multilingual text classification [38]. Several factors influence its effectiveness, such as the choice of which layers to transfer, the similarity between the source and target tasks, and the strategy used for fine-tuning. Lower layers of deep networks usually learn general features that are useful across various domains, while higher layers specialize in task-specific patterns. Although large differences between the source and target tasks may reduce the benefits of transfer learning, it generally performs better than training a model from random initialization. Fine-tuning with target-domain data further improves adaptation by adjusting the transferred knowledge to the new task requirements [52].

4 Anomaly Detection Paradigms

4.1 Anomaly Detection

With the growing use of data across nearly every sector, identifying unusual or incorrect information has become increasingly important. As a result, many algorithms and system architectures have been developed to detect such irregularities efficiently. Anomaly detection techniques are now widely used in many real-world applications, for example, detecting fraudulent transactions in finance, identifying defects in manufacturing, spotting abnormal clinical measurements in healthcare, and recognizing security breaches in cybersecurity [10, 36].

In simple terms, an anomaly is any behavior or data point that deviates from what is expected. These unusual points are often referred to as outliers [19]. An anomaly can appear when a system behaves unexpectedly, becomes disrupted, or when the data contains noise or errors. In anomaly detection, the data is divided into normal and anomalous data, where the majority of the data is considered normal and the data deviating from normal behavior is considered anomalous or abnormal. In Figure 4.1, regions N1 and N2 contain the majority of data points and are therefore considered normal, while points O1 and O2 and region O3 represent outliers relative to the dominant data distribution.

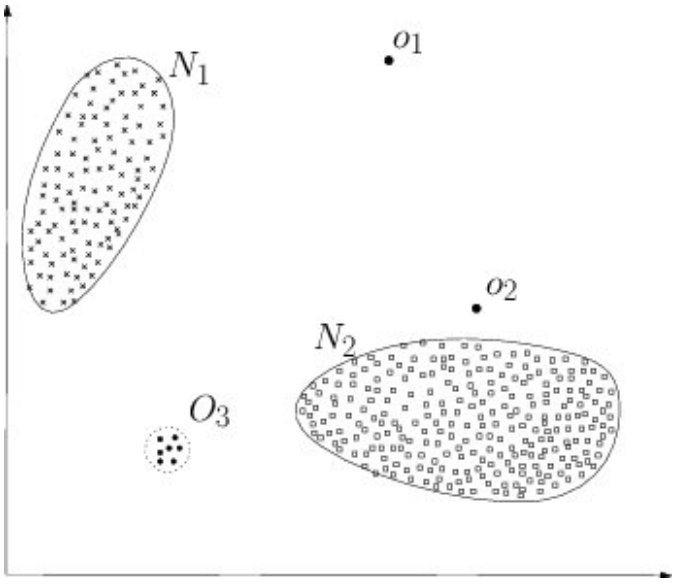

Figure 4.1: A simple example of anomalies in a 2-dimensional data set [10]

Detecting such outliers within datasets is referred to as anomaly detection. Early identification of anomalies can help prevent adverse outcomes such as financial fraud, system failures, or security breaches, and supports the reliable and stable operation of complex systems [10].

4.1.1 Types of Anomalies

Anomalies can appear in different forms depending on how the data behaves. In general, three major types of anomalies are commonly discussed in the literature, each with distinct characteristics and implications:

Point anomalies

A point anomaly occurs when a single data instance significantly deviates from the rest of the data. These anomalies often appear randomly and may not have an obvious interpretation [9]. For example, an unusually large credit card transaction compared to a customer's typical spending pattern would be considered a point anomaly.

Contextual anomalies

A contextual anomaly (also called a conditional anomaly) occurs when an instance is considered anomalous only within a specific context [47]. The same data point may appear normal in one situation but abnormal in another. This type of anomaly is common in time-series or spatial data, where the meaning of a value depends on factors such as time or location. For example, a high temperature reading might be normal in summer but anomalous in winter.

Collective anomalies

A collective anomaly refers to a group of data points that appear normal individually but become anomalous when considered together [1]. This scenario occurs when the relationship or pattern among the points is unusual. For example, in financial transaction monitoring, repeatedly performing the same transaction amount in a short period may indicate suspicious activity, even if each transaction alone is typical.

4.1.2 Challenges of Anomaly Detection

Anomaly detection presents several challenges that are particularly pronounced in visual domains. One key difficulty is the scarcity or complete absence of labeled anomalous data. In many real-world scenarios, anomalous events are rare, diverse, or poorly defined, making the collection of comprehensive labeled datasets impractical. As a result, supervised learning approaches are often unsuitable, motivating the use of one-class and unsupervised methods that model normality and detect deviations at test time.

Another important challenge lies in the choice of data representation. When anomaly detection methods are applied directly to raw image pixels, their performance is often

degraded by high dimensionality, noise, and lack of semantic structure. This can lead to unstable decision boundaries and unreliable anomaly scores, particularly for classical methods such as PCA and One-Class SVM that rely on global variance or geometric assumptions.

Noise and variability within normal data further complicate the task. Normal classes in image datasets may exhibit substantial intra-class variation, and imperfect training data can obscure the distinction between normal and abnormal samples. Without an appropriate representation, anomaly detectors may incorrectly interpret benign variations as anomalies, resulting in poor separability.

This thesis addresses these challenges by focusing on feature representation quality rather than detector complexity. By leveraging transfer learning from deep convolutional neural networks pretrained on large-scale image datasets, images are mapped into a compact and semantically meaningful feature space. Classical one-class methods are then applied to these deep features, reducing the impact of noise, mitigating the curse of dimensionality, and improving the stability of anomaly detection. Rather than attempting to solve all aspects of the anomaly detection problem, this work concentrates on scenarios with limited labeled data and visually complex inputs. The methodology and experiments presented in subsequent sections evaluate how effectively simple and interpretable anomaly detection methods can operate when paired with strong pretrained representations.

4.2 Types of Anomaly Detection Models

Based on the types of input, the deep anomaly detection model can be categorized into three categories: 1)supervised anomaly detection, 2)unsupervised anomaly detection, and 3) semi-supervised anomaly detection [9]. In the case of supervised the labels of both normal and anomalous data instances are used to train a supervised binary or multi-class classifier. Since it is not always possible to collect a huge amount of labelled data, these methods are not as popular as unsupervised or semi-supervised methods. Conversely, unsupervised anomaly detection lacks access to the precise labels of the provided dataset. It detects anomalies by using the common structures within the data instances and outlier observation that makes this method closely aligned with traditional outlier detection [23]. However, these models assume that most of the data is normal and only a small portion is abnormal. If this is not the case, they can easily mistake normal points for anomalies, leading to many false alarms [9]. Semi-supervised methods define the concept of normality and abnormality by using normal and anomalous datasets. The model trains on the normal data and finds anomalies from the dataset mixed with normal and contaminated data [10].

4.3 Methods of Anomaly Detection

This section introduces several common categories of unsupervised and semi-supervised anomaly detection methods. The focus is on approaches that can handle high-dimensional

data and that are relevant to the techniques used in this work.

Distance-based Methods

Distance-based methods rely on the idea that normal data points tend to lie close to one another, while anomalies appear farther away. Distances such as the Euclidean or Manhattan distance [14] are used to measure how far a point is from its neighbors. If a point is much farther away than expected, it is considered unusual. A typical example is the k-nearest neighbor [12] method, in which a point is marked as an anomaly when its distance to the k-th nearest neighbor is significantly larger than that of normal points. Clustering approaches such as k-means or hierarchical clustering also use distance information to detect points that do not fit well into any cluster.

These methods are intuitive and easy to interpret, which makes them popular in practice. Their main challenge is the increased computational cost when many distance calculations are required, especially in high-dimensional spaces. These methods typically require storing the entire dataset and performing extensive distance computations at inference time, which leads to high memory usage and poor scalability. Their performance is also sensitive to hyperparameter choices, such as the neighborhood size, and small changes in these parameters can significantly affect detection results.

Density-based Methods

Density-based methods evaluate how densely populated different regions of the data space are. Normal points tend to appear in dense clusters, whereas anomalies lie in sparse regions with few neighboring points. A point whose local density is much lower than that of its surroundings is likely to be an outlier. The Local Outlier Factor (LOF) [8] is a well-known example that compares the density of each point with that of its neighbors. DBSCAN [16] is another method that identifies low-density points during its clustering process. A more detailed description of LOF is provided later in the 4.4 section.

Density-based approaches are flexible and can detect anomalies that differ subtly from their local neighborhood. They can handle complex data structures that distance-based methods may overlook. Their performance can depend on parameters such as neighborhood size, but when chosen carefully, they are effective tools for many anomaly detection tasks.

Boundary-based Methods

Boundary-based methods aim to learn a boundary that encloses the majority of normal data. Anything falling outside this boundary is considered an anomaly. These approaches do not depend on local neighborhood structure but instead focus on the global shape of the normal data distribution. One-Class SVM [43] is a representative method that uses kernel functions to learn a flexible boundary around the data.

Boundary-based techniques are particularly useful in high-dimensional settings, where local density or distance relationships become less meaningful. Their success depends on appropriate kernel and parameter choices, but when configured properly, they provide strong performance for many anomaly detection problems.

Partition-based Methods

Partition-based methods work by repeatedly splitting the feature space into smaller regions. The key assumption is that anomalies are easier to isolate because they lie in sparse or irregular parts of the data space. Isolation Forest [30] is a well-established example that isolates points through random splits. Points that require only a few splits to isolate are assigned high anomaly scores.

These methods are highly efficient and scalable, making them suitable for large datasets. Their reliance on simple random partitions allows them to perform well in high-dimensional environments without requiring expensive distance or density computations.

Property-based Methods

Property-based methods detect anomalies by examining how well each data point fits the global structure or statistical properties of the dataset [23]. Instead of focusing on distances or densities, they model the underlying relationships among features and measure how much individual points deviate from these learned patterns. Principal Component Analysis (PCA) [25] is a common example that captures the main directions of variation in the data. Points that cannot be well reconstructed from the principal components are flagged as anomalies.

These methods are useful when the data exhibit clear global correlations or low-dimensional structure. They are computationally efficient once the model is learned and complement other approaches by capturing aspects of the data that distance- or density-based methods may miss.

4.4 Classical Methods Used Here

4.4.1 Principal Component Analysis

Principal Component Analysis (PCA) [25] is an unsupervised dimensionality reduction technique that identifies the main directions along which data varies. By projecting high-dimensional data onto a lower-dimensional subspace defined by these directions, PCA preserves the dominant structure of the data while discarding less informative variations. Due to its simplicity and interpretability, PCA is widely used in exploratory data analysis and anomaly detection.

Let

$$X = [x_1^\top, x_2^\top, \ldots, x_n^\top] \in \mathbb{R}^{n \times p} \tag{4.1}$$

denote a dataset of n samples, where each sample $x_i \in \mathbb{R}^p$. PCA first computes the mean of the data,

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{4.2}$$

and the corresponding covariance matrix,

$$\Sigma = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^\top. \tag{4.3}$$

The principal components are obtained by solving the eigenvalue decomposition problem

$$\Sigma u_j = \lambda_j u_j, \tag{4.4}$$

where u_j and λ_j are the eigenvectors and eigenvalues of the covariance matrix. Eigenvectors associated with the largest eigenvalues define a low-dimensional subspace that captures the dominant variance in the data.

PCA for Anomaly Detection

PCA can be applied to anomaly detection based on the assumption that normal data lies close to a low-dimensional subspace, while anomalous data deviates from this structure. In this work, PCA is trained using only normal data samples in order to learn a representation of normal behavior. Given a test sample x_t, it is projected onto the learned PCA subspace and reconstructed as

$$\hat{x}_t = \mu + UU^\top (x_t - \mu), \tag{4.5}$$

where U contains the selected principal components. The anomaly score is defined as the reconstruction error

$$s_t = \|x_t - \hat{x}_t\|_2. \tag{4.6}$$

Samples with large reconstruction errors are considered anomalous.

PCA provides a simple and efficient approach to anomaly detection that does not require labeled anomaly data. When applied to compact feature representations, it offers good scalability, interpretability, and serves as a strong baseline for detecting deviations from normal data structure.

PCA is sensitive to anomalies present during training, assumes linear structure, and depends on the number of retained components. In this work, these limitations are mitigated by training PCA exclusively on normal samples, applying it to deep feature representations instead of raw inputs, and selecting the number of components using a variance preservation criterion. These choices improve the robustness of PCA and its effectiveness as an anomaly detection method.

4.4.2 One-Class Support Vector Machine

One-Class Support Vector Machine (OCSVM) [43] is an unsupervised learning method designed specifically for anomaly detection. Unlike conventional classification algorithms, OCSVM is trained using only normal data and learns a decision boundary that encloses the region where normal samples are concentrated. Samples that fall outside this region are considered anomalous.

Let

$$X_{\text{normal}} = \{x_1, x_2, \ldots, x_n\}, \quad x_i \in \mathbb{R}^p \tag{4.7}$$

denote a set of feature vectors corresponding to normal data samples. OCSVM aims to estimate the support of the normal data distribution by finding a boundary that separates the data from the origin in a high-dimensional feature space.

This is achieved by solving the following optimization problem:

$$\min_{\mathbf{w},\rho,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \tag{4.8}$$

subject to

$$\mathbf{w}^\top \phi(x_i) \geq \rho - \xi_i, \quad \xi_i \geq 0, \tag{4.9}$$

where $\phi(\cdot)$ denotes a feature mapping induced by a kernel function, $\nu \in (0, 1]$ controls the fraction of allowed outliers, and ξ_i are slack variables.

OCSVM for Anomaly Detection

By mapping data into a high-dimensional feature space, OCSVM learns a compact region that contains most normal samples. Given a test sample x_t, the decision function

$$f(x_t) = \mathbf{w}^\top \phi(x_t) - \rho \tag{4.10}$$

indicates whether the sample lies inside or outside the learned boundary. Samples with negative decision values are considered anomalous. For ranking-based evaluation, the decision values can be used as anomaly scores, where larger deviations from the boundary correspond to higher anomaly likelihood.

OCSVM is capable of modeling complex, non-linear decision boundaries through the use of kernel functions and does not require labeled anomaly data. This flexibility makes it effective for detecting subtle deviations from normal behavior, particularly when normal data forms a compact but non-linearly separable distribution.

OCSVM is sensitive to feature scaling and hyperparameter selection, and its performance can degrade when the normal data distribution is highly overlapping with anomalous patterns. In this work, these limitations are mitigated by applying feature normalization based only on normal data and by evaluating OCSVM alongside complementary methods such as PCA. This combined evaluation provides a more reliable assessment of anomaly detection performance and highlights cases where OCSVM is most effective.

4.5 Local Outlier Factor

Local Outlier Factor (LOF) [8] is an unsupervised, density-based anomaly detection method that identifies anomalies by comparing the local density of a data point with that of its neighbors. Unlike global methods, LOF focuses on local structure, making it effective for detecting anomalies that occur in regions of varying data density.

Let

$$X = \{x_1, x_2, \ldots, x_n\}, \quad x_i \in \mathbb{R}^p \tag{4.11}$$

be a dataset of feature vectors. For each data point, LOF first computes the distance to its k nearest neighbors and estimates the local reachability density based on these distances. The local reachability density reflects how densely a point is surrounded by its neighbors.

The LOF score of a data point is defined as the ratio between the average local density of its neighbors and its own local density. A score close to one indicates that the point has a density similar to its neighbors, while larger values indicate that the point lies in a sparser region and is therefore more likely to be anomalous.

LOF for Anomaly Detection

In anomaly detection, LOF assigns higher scores to samples whose local density is significantly lower than that of their neighbors. This local comparison allows LOF to detect anomalies that may not be apparent when considering the global data distribution. Samples with high LOF scores are considered anomalous, while samples with scores close to one are considered normal.

LOF is well-suited for detecting local anomalies and performs effectively when normal data exhibits clusters with different densities. Its reliance on local neighborhoods allows it to identify subtle anomalies that global methods may miss.

LOF is sensitive to the choice of neighborhood size and can be computationally expensive for large datasets. In addition, its performance may degrade in very high-dimensional spaces where distance measures become less meaningful. In this work, LOF is applied to compact feature representations rather than raw data and is evaluated alongside complementary methods to provide a balanced assessment of anomaly detection performance.

4.6 Isolation Forest

Isolation Forest (iForest) [30] is an ensemble-based anomaly detection method that identifies anomalies based on their susceptibility to isolation. Unlike distance- or density-based methods, iForest explicitly exploits the idea that anomalous samples are easier to isolate than normal samples due to their rarity and distinctiveness.

Isolation Forest constructs an ensemble of binary trees, known as isolation trees, by recursively partitioning the data using randomly selected features and split values. Each

tree isolates data points by randomly dividing the feature space until each point is separated from the rest.

The path length required to isolate a data point corresponds to the number of splits needed to separate it. Anomalous points typically require fewer splits and therefore have shorter average path lengths across the ensemble.

Isolation Forest for Anomaly Detection

For a given sample x_t, the anomaly score is computed based on its average path length across all trees in the forest. Samples with shorter average path lengths are assigned higher anomaly scores, indicating a higher likelihood of being anomalous. This scoring mechanism allows iForest to detect anomalies without relying on distance or density estimation.

Isolation Forest scales efficiently to large datasets and performs well in high-dimensional settings. Its reliance on random partitioning makes it less sensitive to feature scaling and well-suited for datasets with heterogeneous feature distributions.

Isolation Forest may struggle to detect anomalies that are not easily isolated through random splits, particularly when anomalies closely resemble normal data. In this work, this limitation is addressed by applying iForest to discriminative feature representations and by comparing its performance with complementary anomaly detection methods.

5 Dataset & Problem Formulation

5.1 Datasets

The initial set of experiments is carried out on MNIST, Fashion-MNIST, and CIFAR-100. To enable a more challenging and large-scale evaluation, Tiny-ImageNet is used for the main experiment.

5.1.1 MNIST

The MNIST dataset [27] consists of $60,000$ training and $10,000$ test images of handwritten digits. Each image is a 28×28 grayscale raster representing one of ten classes $(0-9)$. For CNN architectures that require RGB input, the single channel is duplicated across three channels, and images are resized to match the model's specified input resolution.

5.1.2 Fashion-MNIST

Fashion-MNIST [51] follows the same data split and image resolution as MNIST but contains grayscale images of clothing items rather than digits. As with MNIST, images are converted to three-channel RGB format by channel replication and resized as required. Class labels are integers from 0 to 9.

5.1.3 CIFAR100

CIFAR-100 [26] comprises 50000 training and 10000 test RGB images at (32×32) resolution. In CIFAR-100, each image is annotated with both a coarse label (20 superclasses) and a fine label (100 subclasses). The fine-grained categories correspond to the 100 specific object classes (e.g., tulip, bicycle, mountain) rather than the broader superclasses. In our experiments, we adopt the standard fine labels provided by the dataset and perform classification over all 100 subclasses without using or modifying the coarse label structure. All images are resized to the model's input size during preprocessing.

5.1.4 Tiny-ImageNet-200

Tiny-ImageNet [48] comprises 200 classes of natural images at a resolution of 64×64, with predefined training, validation, and test splits. Each class is identified by a WordNet ID (WNID). We map WNIDs to consecutive integer labels based on the provided wnids.txt file and use words.txt to obtain human-readable class names. Images are resized as required by the backbone architecture.

5.2 Formal Problem Setup

Let $x \in \mathbb{R}^{H \times W \times C}$ be an input image and f_θ be a CNN backbone that produces a feature representation $z = f_\theta(x) \in \mathbb{R}^d$. The global average pooling (GAP) layer is used to obtain the feature representation z in all initial and main experiments. GAP produces a compact, fixed-dimensional embedding by aggregating spatial information across the entire feature map, thereby reducing sensitivity to local spatial variations and noise [29]. Although convolutional feature maps at different depths encode complementary information, the choice of feature extraction layer can substantially influence the behavior of classical one-class anomaly detection methods. For this reason, an additional set of experiments in Chapter 8 evaluates the feature representations extracted from multiple intermediate convolutional layers. Similar observations regarding the importance of high-level semantic features have been reported in prior work on feature-based anomaly detection using deep representations [40, 41]. Based on these empirical observations, the GAP layer is selected as the feature extraction point for all initial and main experiments to ensure robustness, consistency, and comparability across datasets. While intermediate convolutional layers may yield higher performance for specific datasets or anomaly detection methods, the GAP representation provides a stable and dataset-agnostic feature space. Consequently, layer-wise comparisons are reported separately to analyze the effect of feature depth without compromising comparability in the main experiments.

Classifier training

We train f_θ as a multiclass classifier on a subset of classes that excludes the designated normal-anomalous pair (the pair definition is described in subsection 5.3). In the standard setting, this subset contains all classes except the two reserved for anomaly detection. However, for computational efficiency, we sometimes train on a reduced subset, provided that the normal and anomalous classes are omitted. After selecting the training classes, all corresponding labels are remapped to a consecutive index set, and samples from the excluded classes are removed from both the training and test splits.

One-class model fitting

After training the classifier, we extract feature vectors $z = f_\theta(x) \in \mathbb{R}^d$ from the global average pooling layer and fit a one-class detector $\mathcal{D}$ using only normal-class samples. $\mathcal{D}$ is PCA or OCSVM or LOF or IForest. All one-class models are trained exclusively on features from the designated normal class, with anomaly samples used only at test time.

Anomaly scoring

Each detector outputs a continuous anomaly score $s(x)$, which we evaluate directly using threshold-free metrics AUROC. Since our implementation does not apply a fixed decision threshold, we do not report accuracy or F_1; the evaluation relies entirely on continuous-score metrics.

5.3 Pair Definition for "normal vs anomaly"

For each dataset, we specify one class as the *normal* class and one as the *anomalous* class. These two classes are excluded from classifier training and are used solely during the anomaly-detection stage.

- **MNIST / Fashion-MNIST:** class indices $c_N, c_A \in \{0, \ldots, 9\}$.
- **CIFAR-100:** fine-label indices $c_N, c_A \in \{0, \ldots, 99\}$.
- **Tiny-ImageNet:** WNID pairs (w_N, w_A), mapped to integer indices according to the ordering in `wnids.txt`.

For evaluation, we form a two-class subset containing only samples from the selected normal and anomalous classes. These samples are relabeled as

$$0 = \text{normal}, \qquad 1 = \text{anomalous},$$

and the one-class detectors are trained exclusively on the normal-class samples.

5.3.1 Pair Selection

To evaluate the robustness of the anomaly-detection pipeline, the normal–anomalous class pairs are intentionally chosen to be visually similar. Selecting pairs that share structural, textural, or semantic properties creates a more challenging separation problem, thereby providing a more informative test of model efficiency. Easy pairs (e.g., MNIST "0" vs. "8" or CIFAR "apple" vs. "truck") would not meaningfully stress the detector, whereas closely related classes reveal whether the learned feature space preserves subtle distinctions.

Across the experiments, the following pairs were used.

MNIST. MNIST contains handwritten digits, many of which have overlapping shapes or stroke patterns. The selected pairs capture well-known sources of ambiguity:

- Normal:8, Anomalous:9 - rounded loops and nearly identical silhouettes;
- Normal:0, Anomalous:1 - contrasting shape complexity but frequently confusable when written quickly;
- Normal:1, Anomalous:7 - similar straight-stroke structure;
- Normal:2, Anomalous:3 - highly similar curvature patterns.

Fashion-MNIST. Fashion-MNIST includes clothing categories with overlapping textures and contour shapes:

- Normal:0 (T-shirt), Anomalous:6 (Shirt) - both upper-body garments with nearly identical outlines;
- Normal:2 (Pullover), Anomalous:4 (Coat) - long-sleeved tops differing subtly in thickness and collar shape;
- Normal:3 (Dress), Anomalous:4 (Coat) - elongated vertical silhouettes;
- Normal:5 (Sandal), Anomalous:7 (Sneaker) - footwear with similar foreground–background structure.

CIFAR-100. Normal-anomaly pairs were selected from semantically related classes that share similar textures, shapes, or visual structure, making anomaly detection particularly challenging:

- Normal: 0 (beaver), Anomalous: 29 (bear) - both are brown, furry mammals with comparable body structure and texture;
- Normal: 6 (bee), Anomalous: 14 (fly) - small flying insects with similar wing–body geometry and visual appearance;
- Normal: 45 (lamp), Anomalous: 49 (clock) - household objects with round or symmetric shapes and similar contours;
- Normal: 30 (bowl), Anomalous: 95 (cup) - container-like objects that differ mainly in fine-grained details such as handles and rims;
- Normal: 29 (bear), Anomalous: 30 (bowl) - visually dissimilar semantic categories that nonetheless share coarse shape and color statistics in certain viewpoints, providing a contrasting but challenging pairing.

Tiny-ImageNet. Tiny-ImageNet contains a large number of fine-grained categories with substantial intra-class variability and many visually related classes, making anomaly detection particularly challenging at low spatial resolution. The selected normal–anomaly class pairs are defined using WordNet identifiers (WNIDs) and are chosen to reflect varying degrees of semantic and visual similarity:

- (`n02123394`, `n02124075`) - *Persian_cat vs. Egyptian_cat*; two domestic feline breeds with similar facial structure, fur texture, and overall body morphology.
- (`n02132136`, `n02125311`) - *brown_bear vs. cougar*; large quadruped mammals whose comparable stance, coarse shape, and natural coloration can lead to visual ambiguity at reduced resolution.

- (n01910747, n01950731) - *jellyfish vs. crab*; marine organisms that often appear with rounded or radially symmetric silhouettes, resulting in overlapping contour and texture cues in Tiny-ImageNet images.

- (n04251144, n03838899) - *drum vs. oboe*; musical instruments that differ primarily in structural form, with fine-grained material details largely lost at this scale.

- (n02056570, n02190166) - *king_penguin vs. tern/gull family*; seabirds sharing similar color distributions and environmental backgrounds, despite differences in body proportions and beak geometry.

- (n02099601, n02106662) - *golden_retriever vs. german_shepherd*; dog breeds with comparable body shape and fur texture, distinguished mainly by subtle differences in facial features and coloration.

- (n03637318, n04501370) - *lampshade vs. vacuum_cleaner*; household objects that can exhibit overlapping global shape or texture patterns, particularly when viewed under cluttered indoor scenes.

Visual Similarity Justification. The selected normal-anomaly class pairs are chosen to exhibit strong visual overlap, ensuring that anomaly detection is evaluated under challenging and realistic conditions.

In MNIST, ambiguity arises from overlapping digit shapes and stroke patterns (e.g., 2 vs. 3, 1 vs. 7). Fashion-MNIST pairs share similar garment outlines and textures, such as T-shirts vs. shirts or pullovers vs. coats. In CIFAR-100, several pairs involve semantically related categories with comparable textures or coarse shapes, including mammals (beaver vs. bear), insects (bee vs. fly), and household objects (lamp vs. clock, bowl vs. cup). Tiny-ImageNet further increases difficulty through fine-grained categories and low-resolution imagery, where distinctions between classes such as Persian vs. Egyptian cats or bear vs. cougar are often limited to subtle differences in structure or texture.

By focusing on visually similar and fine-grained class pairs across datasets, the evaluation tests anomaly detectors in settings where the boundary between normal and anomalous samples is inherently ambiguous, providing a stringent assessment of one-class detection performance.

5.4 Evaluation and Metrics

This work adopts a unified evaluation framework suitable for unsupervised and one-class anomaly detection. In such settings, models are trained using only samples assumed to represent normal behavior, while anomalous samples are not available during the training process. Ground-truth labels, when available, are used exclusively for evaluation purposes and do not influence model fitting, ensuring that the anomaly detection process remains unsupervised.

Anomaly detection methods typically assign a continuous anomaly score to each sample, reflecting the degree to which the sample deviates from the learned model of normality. Higher scores indicate a greater likelihood of being anomalous. Because different detection methods may produce scores on different scales and with different distributions, direct comparison using fixed decision thresholds is not appropriate. To enable a consistent and threshold-independent comparison across methods, the Area Under the Receiver Operating Characteristic Curve (ROC–AUC) is used as the primary evaluation metric. ROC-AUC measures a model's ability to rank anomalous samples above normal samples across all possible decision thresholds. This property makes it particularly well-suited for anomaly detection, where the true proportion of anomalies is often unknown and operational thresholds may vary across applications.

Formally, let s_i denote the anomaly score assigned to sample i, and let $y_i \in \{0, 1\}$ denote the corresponding ground-truth label, where $y_i = 1$ indicates an anomalous sample. The ROC-AUC score can be interpreted as the probability that a randomly chosen anomalous sample receives a higher anomaly score than a randomly chosen normal sample. A ROC–AUC value of 0.5 corresponds to random ranking performance, while a value of 1.0 indicates perfect separability.

Throughout this thesis, the term *near-optimal performance* is used to describe anomaly detection results that approach the upper bound of achievable separability under the given experimental conditions. Specifically, this refers to ROC–AUC values that are consistently close to 1.0 and comparable to the best-performing configurations observed across different feature representations, methods, and class-pair selections. The term does not imply theoretical optimality, but rather practical performance that is difficult to improve upon without substantially increasing model complexity or introducing additional supervision.

Overall, this evaluation strategy provides a consistent and methodological framework for assessing anomaly detection performance. By applying the same evaluation metric and protocol across all considered methods, observed performance differences can be attributed to the intrinsic properties of the detection algorithms and the underlying feature representations rather than to differences in evaluation methodology.

6 Initial Experiments with Simple CNN

In this section, we present the initial experiments conducted to validate the proposed anomaly-detection pipeline on small-scale datasets before applying it to the more challenging Tiny-ImageNet benchmark. These experiments assess whether (i) the learned CNN representations support effective one-class modeling, and (ii) classical anomaly detectors benefit from operating on learned features rather than on raw pixels.

Figure 6.1 provides a visual overview of the experimental pipeline: For each experiment, a dataset is first selected, and a pair of classes is designated as the normal and anomalous categories. These two classes are entirely excluded from the supervised training process. The remaining classes are used to train the CNN exclusively on the training distribution, reflecting a realistic anomaly-detection setting in which anomalous samples are unavailable during training.

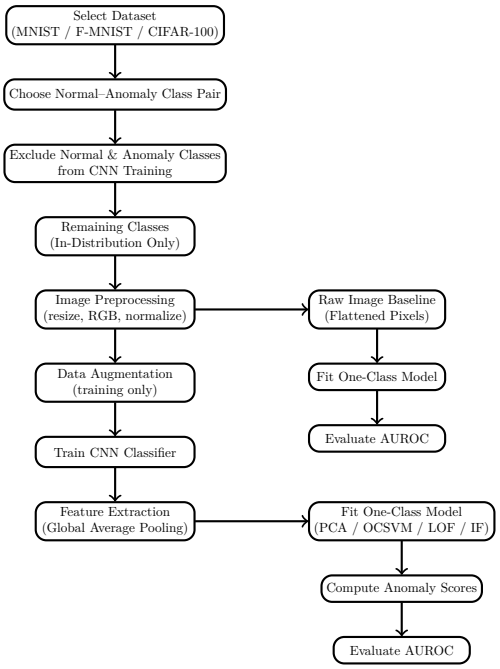

Figure 6.1: Overview of the initial experimental pipeline for CNN-based feature learning and one-class anomaly detection, including raw-image baselines.

After training, the CNN is repurposed as a feature extractor. Each image is passed through the network, and a fixed-dimensional feature representation is obtained from the global average pooling layer. These deep features encode high-level semantic information

learned from the in-distribution classes while discarding the final classification layer, which is specific to the supervised task.

The extracted features corresponding to the normal class are then used to fit a one-class anomaly detection model, PCA, One-Class SVM, Local Outlier Factor, or Isolation Forest. At test time, features from both normal and anomalous samples are provided to the trained detector, which assigns a continuous anomaly score to each sample. Higher scores indicate a greater deviation from the learned model of normality.

Finally, the anomaly detection performance is evaluated using the Area Under the Receiver Operating Characteristic Curve (AUROC), which measures the detector's ability to rank anomalous samples above normal ones in a threshold-independent manner. This unified procedure is applied consistently across all datasets and class-pair configurations to ensure a fair and comparable evaluation of the proposed pipeline.

6.1 Experimental Setup

We evaluate the proposed anomaly-detection pipeline on MNIST, Fashion-MNIST, and CIFAR-100. For each dataset, a single class is designated as the *normal* class and another as the *anomalous* class, forming a normal-anomalous pair. Both classes are completely excluded from the classifier training stage to ensure that the learned representations are not biased toward the evaluation classes. The classifier is trained only on the remaining classes. After training, feature representations are extracted from the global average pooling layer for all samples belonging to the selected normal–anomalous pair and used as input to the one-class anomaly detectors.

All images follow the same preprocessing pipeline as during classifier training: MNIST and Fashion-MNIST samples are converted to RGB and resized; CIFAR-100 images are resized using bilinear interpolation. Data augmentation is applied during the training of the classifier.

6.2 Architecture of the Sample Model

Figure 6.2 illustrates the architecture of the CNN used in the initial experiments. The network takes a $32 \times 32 \times 3$ input image after preprocessing and begins with a 3×3 convolutional layer with 32 filters, followed by batch normalization and a ReLU activation to stabilize and accelerate training. A 2×2 max-pooling layer then reduces the spatial resolution, allowing the network to focus on increasingly abstract visual patterns.

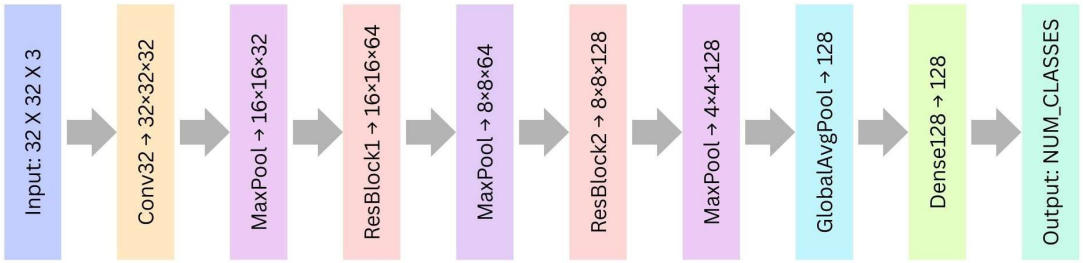

Figure 6.2: Block diagram of the lightweight CNN used in the initial experiments.

The feature extraction backbone consists of two residual blocks that progressively expand the channel dimensionality to 64 and 128, respectively. Each residual block contains two 3×3 convolutional layers with batch normalization, and a skip connection implemented via a 1×1 convolution to match feature dimensions. These shortcut connections facilitate efficient gradient flow and mitigate optimization difficulties, even in this relatively shallow setting. After each residual block, max-pooling halves the spatial resolution, resulting in compact feature maps of size $4 \times 4 \times 128$.

A global average pooling layer aggregates spatial information across each channel, producing a fixed-length 128-dimensional embedding that serves as the feature representation for subsequent anomaly detection. This representation captures high-level semantic information while remaining low-dimensional and computationally efficient. The embedding is further processed by a fully connected layer with 128 units, batch normalization, ReLU activation, and dropout for regularization, before a final softmax layer outputs predictions over the remaining in-distribution classes. Overall, the architecture balances representational capacity and efficiency, making it well-suited for evaluating the effectiveness of feature-based one-class anomaly detection in the initial experiments.

6.3 Feature Extraction

For each image x, the trained network produces a feature vector

$$z = f_\theta(x),$$

taken from the global average pooling layer. Two sets of features are extracted:

1. features from normal-class training samples, used to fit the one-class detectors;
2. features from the test set containing both normal and anomalous samples, used to compute anomaly scores and evaluate detection performance.

6.4 One-Class Detectors

We evaluate four detectors:

- **PCA**: anomaly score based on reconstruction error;
- **OCSVM**: anomaly score given by the decision function;
- **LOF**: anomaly score given by the negated local density estimate;
- **Isolation Forest**: anomaly score given by the negated isolation score.

These four detectors were selected to provide a representative coverage of classical one-class anomaly detection paradigms while maintaining interpretability and methodological simplicity. PCA represents projection-based approaches that model normality through low-dimensional structure, OCSVM exemplifies boundary-based methods that learn a decision region enclosing normal data, LOF captures density-based detection by identifying locally sparse regions, and Isolation Forest implements a partition-based strategy that isolates anomalies through random splits. Together, these methods span complementary detection principles and allow a systematic comparison of how different algorithmic assumptions interact with deep feature representations.

Moreover, all four methods operate naturally in one-class or unsupervised settings, require only normal data for training, and are computationally efficient, making them well-suited for evaluating the impact of feature representation quality without introducing architectural complexity. This design ensures that observed performance differences primarily reflect the effect of feature extraction rather than the use of specialized or black-box anomaly detection models.

All detectors are trained solely on normal-class features. Evaluation is performed on the mixed normal-anomaly set using AUROC.

6.5 Results

Table 6.1 reports AUROC scores across all evaluated class pairs. Overall, all four detectors tend to benefit from CNN-derived features, particularly on more complex datasets. One-Class SVM and Local Outlier Factor achieve strong performance across all datasets, with LOF yielding the highest average AUROC in most cases. PCA performs competitively, particularly on CIFAR-100. Isolation Forest exhibits higher variability across class pairs but still benefits noticeably from feature extraction, particularly on CIFAR-100, where raw pixel representations often lead to substantial performance degradation. Based on these observations, all four detectors are retained for the main ResNet-based experiments on Tiny-ImageNet to enable a comprehensive and fair comparison.

6.6 Comparative Analysis

6.6.1 Impact of Class-Pair Choice

For each detector and each normal-anomaly class pair, we compute

$$\Delta\text{AUROC} = \text{AUROC}_{\text{features}} - \text{AUROC}_{\text{raw}}.$$

Table 6.1: AUROC scores for the initial experiments across MNIST, Fashion-MNIST, and CIFAR-100. "With feature extraction" refers to detectors applied to CNN features. "W/O feature extraction" refers to detectors applied to raw, resized images.

Dataset / Pair	**With Feature Extraction**				**W/O Feature Extraction**			
	PCA	OCSVM	LOF	IForest	PCA	OCSVM	LOF	IForest
MNIST N:8, A:9	0.7776	0.8044	**0.9135**	0.7966	0.7891	0.8083	0.8940	0.6926
MNIST N:0, A:1	0.9985	**0.9991**	0.9964	0.9984	0.9515	0.9510	0.8528	0.9603
MNIST N:1, A:7	0.9943	**0.9970**	0.9765	0.9929	0.9930	0.9935	0.9870	0.9881
MNIST N:2, A:3	0.7766	0.8710	**0.9887**	0.8046	0.7736	0.7842	0.9067	0.8400
MNIST Average	0.8868	0.9179	**0.9688**	0.8981	0.8768	0.8843	0.9096	0.8703
F-MNIST N:0, A:6	**0.8635**	0.8614	0.7973	0.8616	0.7861	0.7875	0.6631	0.8138
F-MNIST N:2, A:4	0.7094	**0.7342**	0.7057	0.7125	0.4963	0.5400	0.5935	0.4995
F-MNIST N:3, A:4	0.8058	0.8785	0.7669	0.8598	0.9270	0.9218	0.8659	**0.9449**
F-MNIST N:5, A:7	0.4934	0.5777	**0.7204**	0.4893	0.4985	0.4676	0.4870	0.5336
F-MNIST Average	0.7180	0.7629	**0.7476**	0.7308	0.6770	0.6792	0.6524	0.6980
C100 N:0, A:29	0.9647	0.9705	0.9572	**0.9743**	0.6591	0.6985	0.7518	0.7699
C100 N:6, A:14	0.9927	0.9920	0.9846	**0.9979**	0.9481	0.9537	0.9502	0.9529
C100 N:45, A:49	0.9515	0.9511	**0.9644**	0.9424	0.6140	0.7330	0.6605	0.7022
C100 N:29, A:30	0.9137	0.9240	**0.9347**	0.8784	0.4482	0.5787	0.7320	0.6494
C100 N:30, A:95	0.5742	0.5624	0.6475	0.5963	0.6414	**0.6689**	0.6556	0.6565
CIFAR-100 Average	0.8794	0.8800	**0.8977**	0.8779	0.6622	0.7266	0.7500	0.7462

A ΔAUROC bar chart highlights how anomaly-detection performance changes when operating on CNN-derived features instead of raw pixel inputs. Positive values indicate that feature extraction improves anomaly detection performance, whereas negative values indicate cases where raw inputs perform comparably or better.

Figure 6.3 shows that the effect of feature extraction is highly dependent on both the class pair and the detector. On CIFAR-100, most class pairs exhibit clear positive gains across all methods, with especially large improvements for PCA and Isolation Forest. In particular, pairs such as C100 45 vs. 49 and C100 29 vs. 30 show substantial increases in AUROC, suggesting that CNN features provide representations that are more amenable to these detectors than raw pixel inputs.

In contrast, MNIST and Fashion-MNIST display more heterogeneous behavior. While some pairs benefit from feature extraction (e.g., MNIST 0 vs. 1 and F-MNIST 2 vs. 4),

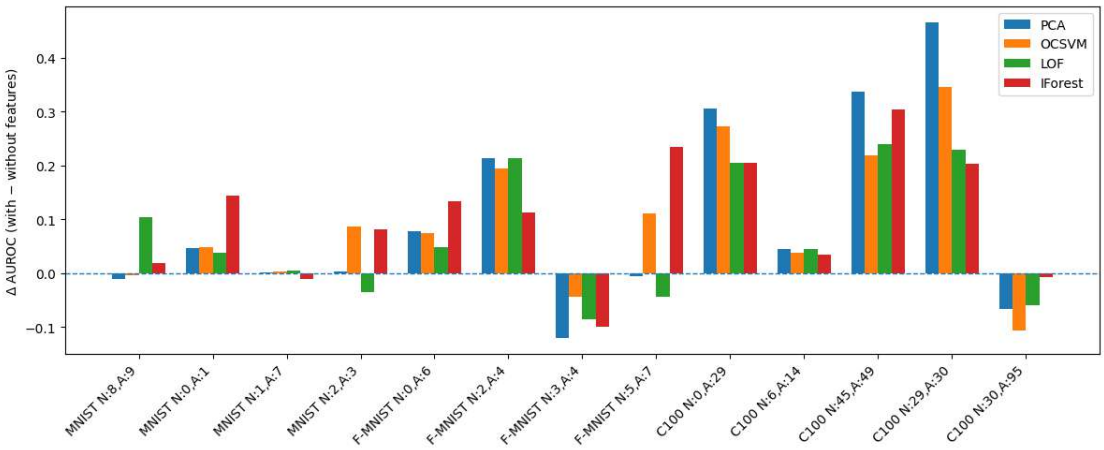

Figure 6.3: Change in AUROC (ΔAUROC) for each normal-anomaly class pair, computed as the difference between feature-based and raw-input anomaly detection. Positive values indicate that feature extraction improves anomaly detection performance.

others show only marginal changes or even slight degradations for certain detectors. For visually similar or fine-grained pairs such as MNIST 2 vs. 3 and F-MNIST 3 vs. 4, several methods exhibit near-zero or negative ΔAUROC. This suggests that when local pixel-level patterns dominate class differences, raw representations may already yield competitive performance for these pairs.

In terms of improvement over raw inputs, PCA exhibits predominantly positive ΔAUROC values, particularly on CIFAR-100, while OCSVM also shows frequent improvements on several class pairs. In contrast, Local Outlier Factor and Isolation Forest are more sensitive to the specific normal–anomaly pairing, showing both notable gains and noticeable drops depending on class similarity.

Overall, the ΔAUROC results indicate that feature-based anomaly detection is beneficial in many, but not all, settings. The magnitude and even the direction of the effect depend strongly on the degree of visual similarity between the classes and on the detector employed. These findings motivate the evaluation of multiple class pairs of varying difficulty and caution against assuming uniform benefits from feature extraction across datasets or methods.

6.6.2 Detector Comparison

Figure 6.4 summarizes feature-based AUROC scores for all detectors across every dataset and class pair. The heatmap shows that PCA, Isolation Forest, LOF, and OCSVM achieve high performance on most of the class pairs, where most entries are close to the upper end of the scale. This suggests that CNN feature representations support effective anomaly detection for many class pairs.

However, the performance patterns differ across datasets and detectors. On MNIST,

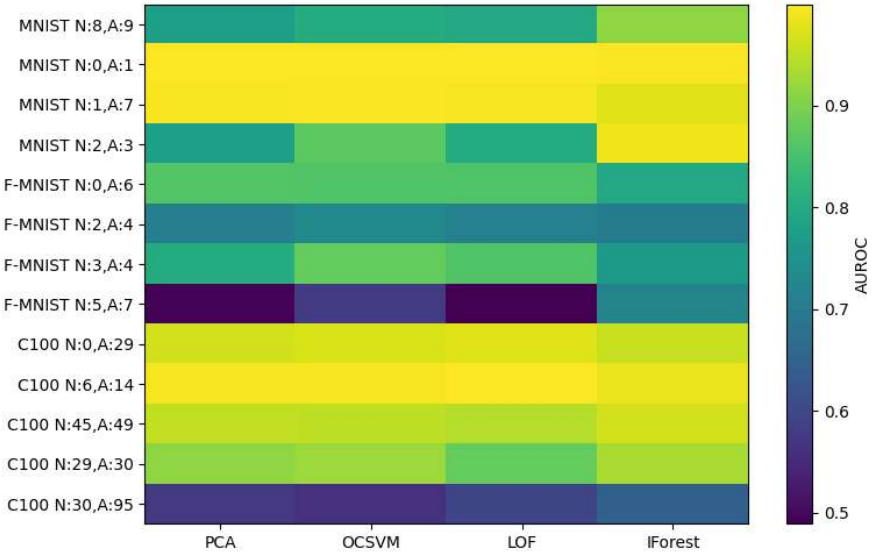

Figure 6.4: Heatmap of AUROC scores obtained using feature-based anomaly detection across all datasets and normal-anomaly class pairs. Columns correspond to the four detectors (PCA, Isolation Forest, LOF, OCSVM).

several class pairs (e.g., 0 vs. 1 and 1 vs. 7) are near-perfect, while more ambiguous pairs such as 8 vs. 9 and 2 vs. 3 show noticeably lower scores, with more pronounced drops for some methods, including PCA and LOF. Fashion-MNIST exhibits the greatest variability: while some pairs (e.g., 3 vs. 4) are handled well, others (notably 5 vs. 7) lead to markedly reduced AUROC, particularly for PCA and LOF. These results suggest that feature-based separability depends strongly on the degree of visual similarity between the class pairs.

In absolute feature-space performance, OCSVM attains high AUROC values on most class pairs and shows comparatively fewer low-performing cases, suggesting comparatively stable performance in the learned feature space. PCA also performs strongly in many settings, especially on CIFAR-100, but degrades more on difficult Fashion-MNIST pairs. In contrast, LOF and Isolation Forest exhibit greater variability across class pairs, with noticeable drops for several challenging cases, suggesting their sensitivity to local density structure and feature distribution characteristics.

Overall, the heatmap confirms that CNN-derived features provide a highly discriminative representation for anomaly detection in many scenarios, particularly for complex datasets such as CIFAR-100. At the same time, the observed variability across detectors and class pairs highlights that strong average performance does not guarantee uniform robustness, underscoring the importance of evaluating multiple datasets and normal-anomaly configurations.

6.6.3 Feature-Based vs. Raw-Input Detection

A direct comparison between feature-based and raw-input anomaly detection is shown in Figure 6.5. Each point represents the AUROC achieved by a detector on raw pixels (horizontal axis) versus CNN features (vertical axis). Points located above the diagonal, therefore, indicate cases where feature extraction leads to improved anomaly detection performance, while points below the diagonal indicate degradation

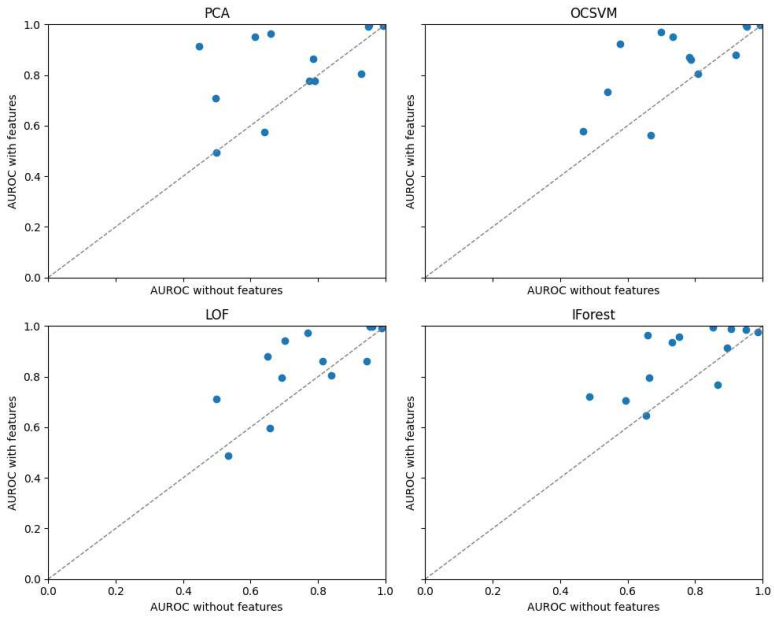

Figure 6.5: Scatter plots comparing AUROC obtained with raw inputs (horizontal axis) and CNN features (vertical axis) for each anomaly detector. Points above the diagonal indicate improved performance when using feature extraction.

The scatter plots show that a majority of points lie above the diagonal for all four detectors, indicating that feature extraction generally improves anomaly-detection performance across class pairs. For PCA, the majority of points are clearly separated above the diagonal, often with substantial vertical offsets, demonstrating frequent and sometimes large gains from operating in the learned feature space. Isolation Forest also exhibits predominantly positive shifts, although with greater dispersion, reflecting more variable improvements across class pairs.

LOF and OCSVM likewise display a clear upward trend, with most points above the diagonal. However, their gains are more heterogeneous: several points lie close to the diagonal, and a small number approach it closely, indicating that feature extraction provides only modest improvements for certain class pairs. Despite this variability, there are a few instances where performance with features is noticeably worse than with raw inputs.

Overall, the scatter plots confirm that CNN-derived features improve AUROC for

most detectors and class pairs considered. The magnitude of the benefit is detector-dependent, with PCA showing some of the largest gains across class pairs, while LOF, Isolation Forest, and OCSVM exhibit more moderate but still predominantly positive effects. These results reinforce the conclusion that learned feature representations enhance anomaly detection performance relative to raw pixel inputs, although the extent of the improvement varies with the underlying detection method.

7 Experiments with ResNet50

In this section, we present the main experiments designed to evaluate the anomaly-detection framework considered in this thesis on datasets of increasing visual complexity. Unlike the lightweight CNN employed in the initial experiments, this stage adopts a high-capacity, ImageNet-pretrained ResNet50 backbone in order to examine the effect of deeper, and commonly regarded as semantically richer, feature representations on one-class anomaly detection. The objective of these experiments is twofold: (i) to determine whether features extracted from a deep pretrained network further enhance the performance of classical one-class models, and (ii) to assess whether the performance trends observed on simpler datasets generalize to more complex and fine-grained visual domains.

Based on the results of the initial experiments, four anomaly-detection methods are considered in this section: Principal Component Analysis (PCA), One-Class Support Vector Machine (OCSVM), Local Outlier Factor (LOF), and Isolation Forest(IForest). These methods represent complementary detection paradigms and enable a comprehensive evaluation of how deep feature representations interact with different algorithmic assumptions.

7.1 Experimental Setup

All experiments follow the same anomaly-detection protocol introduced in the previous chapter. Specifically, the normal-anomaly class-pair selection strategy, class exclusion procedure, preprocessing pipeline, feature extraction process, anomaly scoring, and AUROC-based evaluation remain unchanged to ensure comparability between the initial and main experiments. An overview of this unified pipeline is provided in Figure 7.1.

Datasets and Pair Selection The experiments are conducted on four benchmark datasets: MNIST, Fashion-MNIST, CIFAR-100, and Tiny-ImageNet. A detailed description of these datasets is provided in Section 5.1. Tiny-ImageNet serves as the primary benchmark due to its high visual variability and a large number of semantically related classes, while the remaining datasets are included to evaluate generalization across simpler domains.

Similar to the initial experiment(Chapter 6), for each dataset, anomaly detection is framed as a class-exclusion task. One class is designated as the normal class and another as the anomalous class. Both are removed from supervised training and reserved exclusively for evaluation. For Tiny-ImageNet, classes are referenced using WordNet identifiers (WNIDs) [32], which uniquely define semantic categories in the ImageNet

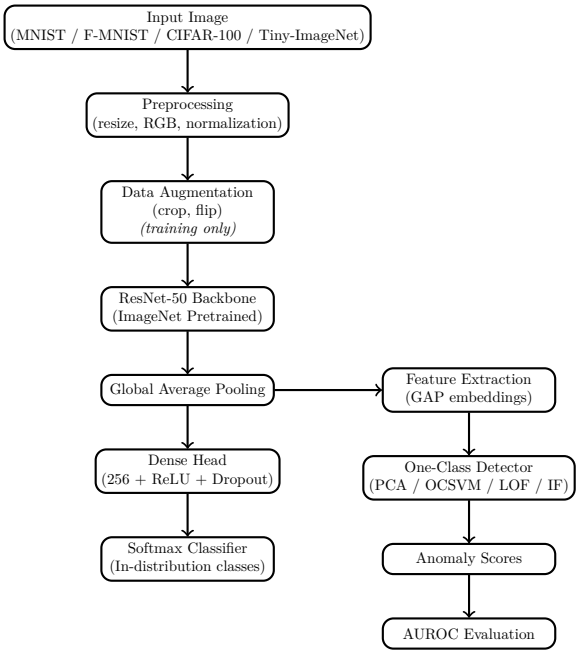

Figure 7.1: Training and anomaly-detection pipeline using a ResNet-50 backbone. The classifier is trained only on in-distribution classes, while features from the global average pooling layer are used for one-class anomaly detection.

hierarchy and provide a stable mapping between class labels and the dataset directory structure. Using WNIDs ensures reproducibility and consistency across experiments, as class indices are independent of dataset ordering or preprocessing choices. For the remaining datasets, standard label indices are used, following the same procedure as in the initial experiments.

7.2 ResNet50 Architecture and Training

The model consists of a ResNet-50 backbone pretrained on ImageNet, followed by a lightweight classification head: a global average pooling layer, a 256-unit ReLU dense layer with dropout, and a final softmax layer over the non-excluded classes. All backbone layers are set to trainable, enabling full fine-tuning.

Training uses the Adam optimizer (learning rate $3 \cdot 10^{-4}$), with

- early stopping on validation loss,
- learning-rate reduction on plateau,
- termination upon encountering numerical instabilities,
- optional model checkpointing.

The classifier is trained on the reduced label set after excluding the designated normal and anomalous classes. The resulting model is subsequently used as a feature extractor for anomaly detection.

7.3 Feature Extraction

While the global average pooling (GAP) layer is used as the primary baseline due to its stability, additional experiments (section 8) investigate features extracted from multiple intermediate convolutional layers at different depths of the network. These layers capture complementary levels of abstraction, ranging from mid-level structural information to high-level semantic cues. Two sets of features are used: normal-class training features for fitting the one-class detectors, and test-set features containing both normal and anomalous samples for computing anomaly scores and evaluating performance.

7.4 Anomaly Detection Methods

For PCA, extracted features are standardized using statistics computed from the normal-class training data and reshaped into two-dimensional vectors. PCA retains 128 principal components, which defines the dimensionality of the projected feature space used for anomaly scoring. The number of components is chosen as a compromise between expressive capacity and computational efficiency, and is kept fixed across datasets for consistency. Anomaly scores are obtained using the PyOD decision function.

OCSVM is trained on the standardized normal-class features using an RBF kernel. The parameter $\nu = 0.3$ controls the fraction of training samples permitted outside the learned decision boundary, while $\gamma = \text{scale}$ adapts the kernel width to the feature variance. Anomaly scores are computed from the signed decision function.

LOF is applied to flattened feature vectors in `novelty` mode and trained solely on normal-class samples. A fixed neighborhood size of $k = 25$ determines the number of nearest neighbors used to estimate local sample density. Decision scores are negated so that higher values indicate stronger anomaly likelihood.

IForest is trained on flattened normal-class feature vectors using an ensemble of 300 trees, which defines the size of the isolation-based scoring ensemble. The contamination parameter is set to `auto`, allowing the expected proportion of anomalies to be inferred from the data. Anomaly scores are obtained from the model's scoring function and negated such that higher values correspond to greater anomaly likelihood.

7.5 Evaluation Protocol

All the methods are fitted using only normal-class training features. Anomaly scores are then computed on the evaluation set containing both normal and anomalous samples. Performance is measured using the area under the receiver operating characteristic curve

(AUROC), which provides a threshold-independent measure of separability and is robust to class imbalance.

7.6 Results and Observations

7.6.1 Overall Performance with and without Feature Extraction

The table 7.1 summarizes the AUROC scores obtained by the four methods with and without ResNet50-based feature extraction across all evaluated datasets and class-pair configurations. The results show that the quality of the underlying feature representation has a strong influence on anomaly-detection performance.

Table 7.1: AUROC results for PCA, OCSVM, LOF, and Isolation Forest with and without ResNet50-based feature extraction (Global Average Pooling layer). Higher AUROC values indicate better anomaly separability.

Dataset / Class Pair	**With Feature Extraction (GAP)**				**Without Feature Extraction**			
	PCA	**OCSVM**	**LOF**	**IForest**	**PCA**	**OCSVM**	**LOF**	**IForest**
MNIST								
N:8, A:9	**0.9666**	0.9662	0.9476	0.9472	0.7891	0.8083	0.8940	0.6926
N:0, A:1	0.9952	**0.9958**	0.9951	0.9854	0.9515	0.9510	0.8528	0.9603
N:1, A:7	0.9718	0.9736	0.9785	0.9454	0.9930	**0.9935**	0.9870	0.9881
N:2, A:3	0.7091	0.7207	0.8430	0.6167	0.7736	0.7842	**0.9067**	0.8400
MNIST Average	0.9107	0.9141	**0.9411**	0.8737	0.8768	0.8842	0.9101	0.8702
Fashion-MNIST								
N:0, A:6	0.7602	0.7543	0.7787	0.7087	0.7861	0.7875	0.6631	**0.8138**
N:2, A:4	**0.9592**	0.9569	0.9245	0.9465	0.4963	0.5400	0.5935	0.4995
N:3, A:4	0.8713	0.8709	0.8443	0.8576	0.9270	0.9218	0.8659	**0.9449**
N:5, A:7	0.6988	0.7098	**0.7970**	0.6608	0.4985	0.4676	0.4870	0.5336
Fashion-MNIST Average	0.8224	0.8230	**0.8361**	0.7934	0.6770	0.6792	0.6524	0.6980
CIFAR-100								
N:0, A:29	0.9689	0.9765	0.9673	**0.9935**	0.6591	0.6985	0.7518	0.7699
N:6, A:14	0.7241	0.7274	0.7310	0.6502	0.9481	**0.9537**	0.9502	0.9529
N:45, A:49	0.9772	0.9772	**0.9780**	0.8676	0.6140	0.7330	0.6605	0.7022
N:30, A:95	0.6539	0.6539	0.6572	0.6390	0.6414	**0.6689**	0.6556	0.6565
N:29, A:30	0.9162	0.9165	**0.9332**	0.7153	0.4482	0.5787	0.7320	0.6494
CIFAR-100 Average	0.8481	0.8503	**0.8533**	0.7731	0.6622	0.7266	0.7500	0.7462
Tiny-ImageNet								
N:n02123394, A:n02124075	0.8960	0.8908	**0.9212**	0.8664	0.3804	0.3952	0.4096	0.4048
N:n02132136, A:n02125311	0.9876	0.9872	**0.9892**	0.9508	0.8364	0.8344	0.8128	0.8480
N:n01910747, A:n01950731	0.9964	0.9968	**0.9976**	0.9704	0.4348	0.4208	0.4208	0.4596
N:n04251144, A:n03838899	0.9784	0.9776	**0.9832**	0.9568	0.2204	0.1956	0.1468	0.2064
N:n02056570, A:n02190166	0.9196	0.9148	**0.9288**	0.8648	0.6556	0.6780	0.6816	0.6864
N:n02099601, A:n02106662	0.9496	**0.9504**	0.9496	0.9496	0.4784	0.4860	0.5552	0.4728
N:n03637318, A:n04501370	0.9552	0.9568	**0.9648**	0.9196	0.7652	0.7268	0.6396	0.7680
Tiny-ImageNet Average	0.9547	0.9535	**0.9621**	0.9255	0.5387	0.5338	0.5238	0.5494

When the four classical anomaly detectors are applied to features extracted from the fine-tuned ResNet50 model, a clear and consistent trend toward improved performance is observed across datasets. Operating in the deep feature space generally enhances separability, with PCA, OCSVM, LOF, and Isolation Forest frequently achieving AUROC values above 0.90 on Fashion-MNIST, CIFAR-100, and Tiny-ImageNet. The effect is particularly pronounced on Tiny-ImageNet, where several class pairs that are poorly distinguishable under raw pixel representations become highly separable once deep features

are employed, suggesting that the learned representations capture higher-level structure that is difficult to exploit using raw pixel inputs.

Among the evaluated detectors, PCA and OCSVM demonstrate the most stable and consistently strong performance when combined with ResNet50 features. Their AUROC scores remain high across datasets and class-pair selections, suggesting that both projection-based and boundary-based models can effectively characterize normality within the learned embedding space. LOF and Isolation Forest also benefit substantially from deep features and often achieve competitive or superior performance on specific class pairs, particularly on Tiny-ImageNet. However, their results exhibit greater variability across pairs, reflecting sensitivity to local density estimation in LOF and to stochastic partitioning in Isolation Forest.

In contrast, anomaly detection performed directly on raw input representations yields noticeably weaker and less consistent results, with performance degradation becoming more severe as dataset complexity increases. While raw pixel representations can produce reasonable AUROC values for some simple MNIST class pairs, performance drops substantially on Fashion-MNIST and CIFAR-100 with the most severe degradation occurring on Tiny-ImageNet. For several Tiny-ImageNet pairs, AUROC values without feature extraction approach 0.5, indicating performance close to random guessing and underscoring the difficulty of modeling normality in high-dimensional, unstructured pixel space.

A detector-wise comparison further reveals that LOF and Isolation Forest are particularly sensitive to the absence of feature extraction, exhibiting large performance drops and increased variance on complex datasets. PCA and OCSVM show comparatively more robust behavior on raw inputs for simpler datasets, but still experience significant degradation as visual complexity increases. These observations suggest that detector choice alone is insufficient to compensate for inadequate feature representations, particularly in complex visual domains.

Overall, the results highlight the dominant role of representation learning in one-class visual anomaly detection. When paired with deep, pretrained, and fine-tuned ResNet50 features, even relatively simple and interpretable classical detectors—such as PCA, OCSVM, LOF, and Isolation Forest—achieve strong and reliable performance across datasets of increasing complexity. In contrast, without feature extraction, all four methods become unstable and unreliable in challenging visual domains. These findings confirm that deep feature representations are a key prerequisite for robust anomaly detection and that the proposed ResNet50-based pipeline scales effectively to complex datasets.

7.6.2 Impact of Feature Extraction Across Class Pairs

Figure 7.2 provides a fine-grained, class-pair-level analysis of the impact of feature extraction on anomaly detection performance by reporting ΔAUROC for each normal-anomaly configuration and detector. By focusing on performance differences rather than absolute AUROC values, the figure largely isolates the contribution of feature representation choices and highlights how the benefit of feature extraction varies across datasets and

class-pair choices.

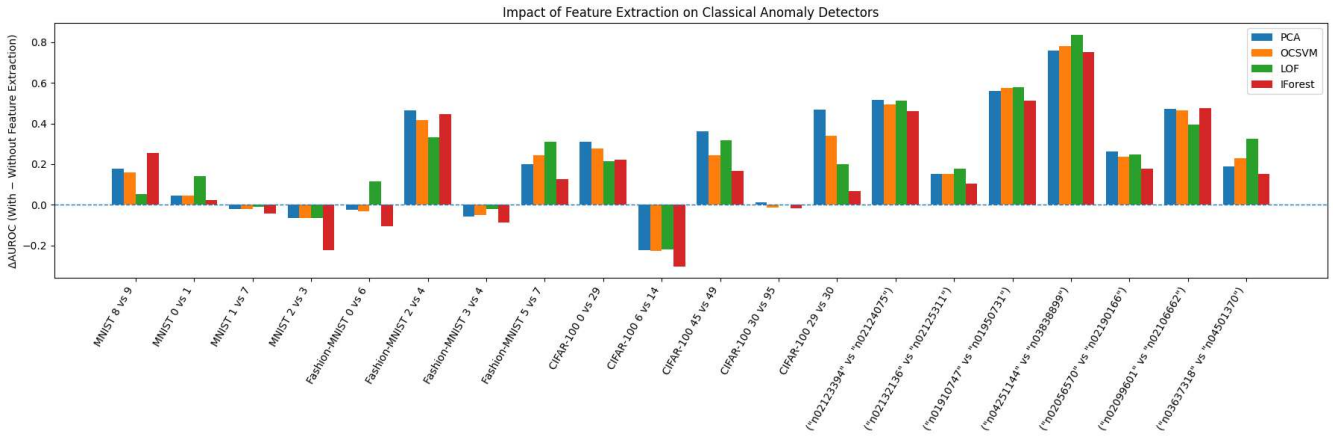

Figure 7.2: Pair-wise impact of feature extraction on anomaly detection performance. Each bar reports ΔAUROC, defined as the difference between AUROC obtained with ResNet50-based feature extraction and AUROC obtained without feature extraction. Positive values indicate improved separability due to feature extraction.

For MNIST and Fashion-MNIST, the observed ΔAUROC values are generally modest and exhibit noticeable variation across class pairs and detectors. In several MNIST configurations, particularly for OCSVM, baseline performance without feature extraction is already high, resulting in limited gains or occasionally slightly negative ΔAUROC values. This behavior reflects the low visual complexity of handwritten digit images, where simple pixel-level patterns and global intensity structure can already provide sufficient information for effective anomaly detection in certain class-pair settings.

In contrast, CIFAR-100 and Tiny-ImageNet display predominantly positive and often substantial ΔAUROC values across most class pairs. The gains are especially pronounced for Tiny-ImageNet with several class pairs exhibiting improvements exceeding +0.5 across detectors. These large performance gaps indicate that anomaly detection based on raw pixel representations becomes increasingly unreliable as visual complexity and intra-class variability grow. In such settings, raw inputs are less effective at exposing high-level structure relevant for anomaly detection, whereas deep feature representations learned by ResNet50 provide a more compact and discriminative embedding space in which normality can be modeled effectively.

Overall, the pair-wise analysis demonstrates that the advantage of feature extraction is not uniform across tasks but is strongly influenced by dataset complexity and class-pair difficulty. The consistently large gains observed for challenging CIFAR-100 and Tiny-ImageNet pairs support the inclusion of semantically diverse and fine-grained class configurations in the experimental design.

7.6.3 Detector-Wise Performance Patterns

Figure 7.3 presents a heatmap of AUROC values obtained with ResNet50-based feature extraction, enabling a systematic comparison of the four classical anomaly detectors across all evaluated normal-anomaly class pairs. By organizing results by detector and class pair, the figure highlights both shared performance trends and detector-specific behavior in the deep feature space.

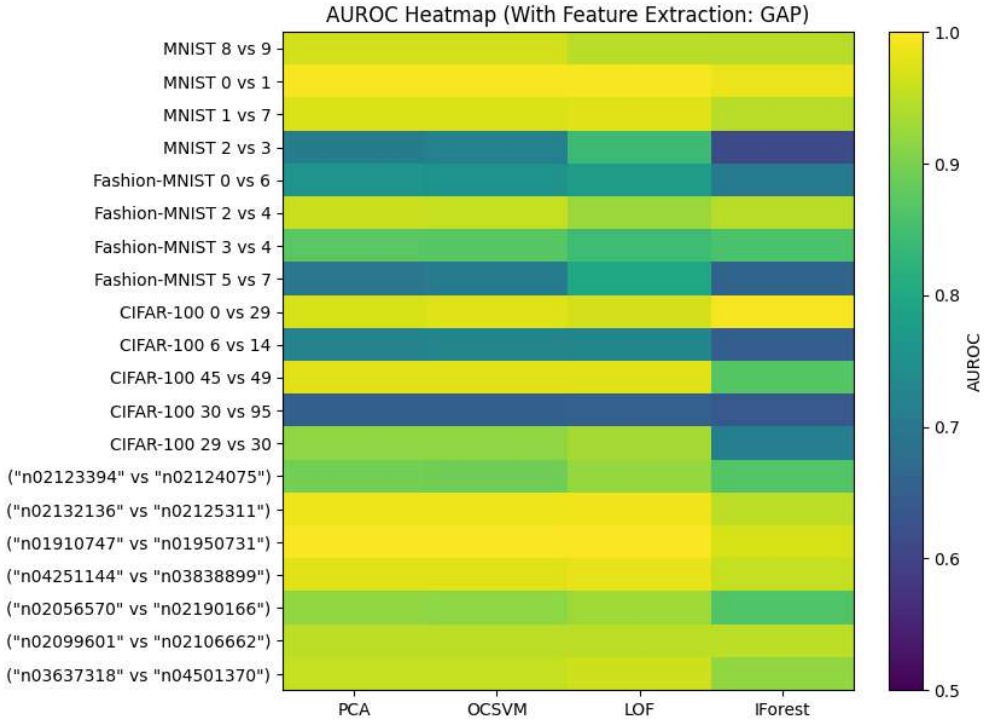

Figure 7.3: AUROC heatmap with ResNet50-based feature extraction (GAP). Each row represents a normal-anomaly class pair and each column corresponds to a classical anomaly detector. Higher values indicate better anomaly separability.

Across all datasets, PCA and OCSVM exhibit strong performance across most class pairs. This suggests that both projection-based and boundary-based methods can effectively exploit the global structure induced by deep feature representations. This behavior is consistent with the modeling assumptions of PCA, which emphasizes dominant low-dimensional variance, and OCSVM, which learns a compact decision boundary around normal samples in feature space.

LOF and Isolation Forest also achieve high AUROC values for many class pairs, especially on CIFAR-100 and Tiny-ImageNet, but their performance displays greater variability across configurations. This variability reflects the sensitivity of LOF to local density fluctuations and the stochastic nature of Isolation Forest's partitioning strategy.

Nevertheless, the overall performance of all four detectors converges when deep feature extraction is applied, indicating that a well-structured representation space substantially reduces the dependence on the specific anomaly detection algorithm.

The heatmap therefore reinforces the central role of representation quality: once informative deep features are available, multiple classical detectors are capable of achieving comparably strong anomaly detection performance.

7.6.4 Feature-Based Versus Raw Input Representations

Figure 7.4 directly contrasts feature-based and raw-input anomaly detection by plotting AUROC values obtained without feature extraction against those obtained with ResNet50-based feature representations. Each point corresponds to a normal-anomaly class pair, and the diagonal denotes equal performance under both representations, providing an intuitive visual reference for performance gains or losses.

Across all four detectors, the majority of points lie well above the diagonal, demonstrating that feature extraction consistently improves anomaly detection performance. This upward shift is particularly pronounced for CIFAR-100 and Tiny-ImageNet, where many class pairs exhibit poor separability when operating on raw inputs but achieve strong performance after feature extraction. These results illustrate the limitations of raw pixel space for one-class modeling in complex visual domains and highlight the role of deep networks in producing semantically meaningful embeddings.

While PCA and OCSVM show the most consistent and stable improvements across class pairs, LOF and Isolation Forest also benefit substantially from feature extraction, despite greater dispersion in their results. The scatter plots therefore provide direct evidence that deep feature representations fundamentally reshape the anomaly detection landscape: they enable classical detectors to operate in a structured embedding space with reduced effective dimensionality, where normality is easier to characterize and anomalies are more readily distinguishable.

Taken together, the three figures demonstrate that representation learning is the primary driver of performance in one-class visual anomaly detection. Detector choice influences performance to a secondary extent, but without feature extraction, all methods struggle to scale to visually complex datasets.

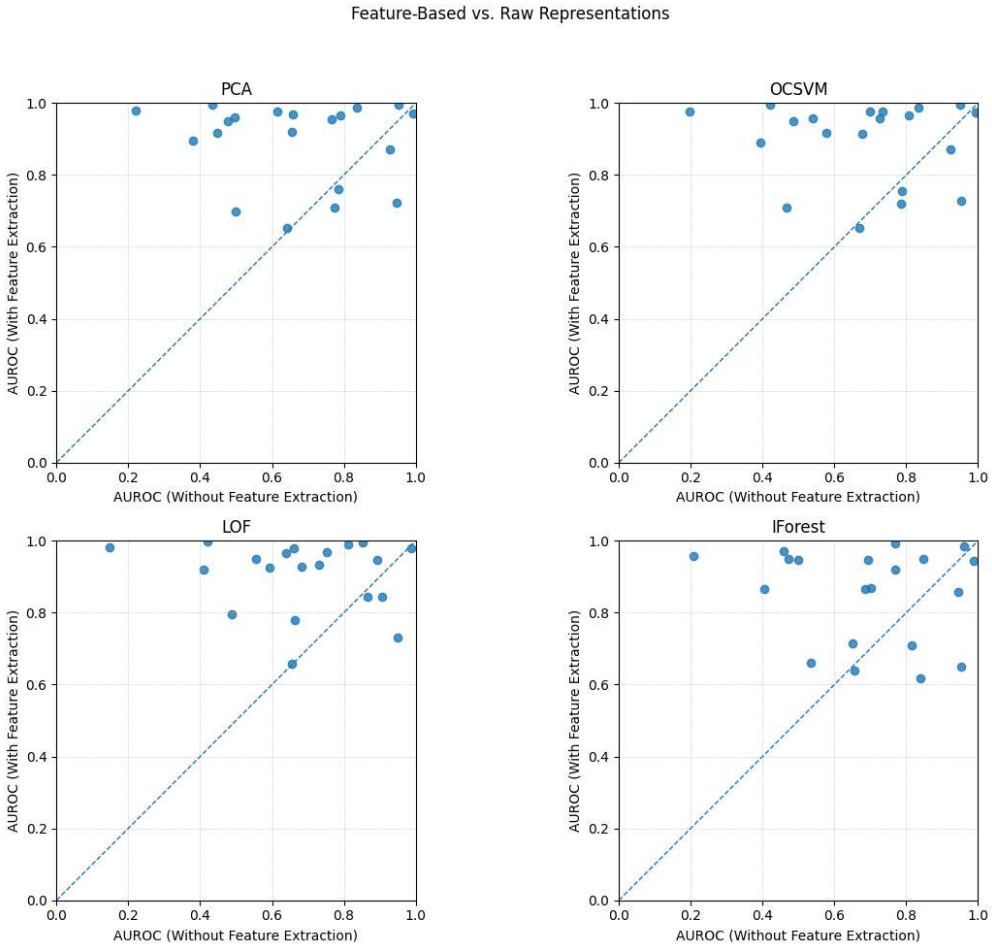

Figure 7.4: Comparison of anomaly detection performance with and without feature extraction. Each point represents a normal- anomaly class pair. Points above the diagonal indicate improved performance when using ResNet50-based feature representations.

8 Additional Experiments

This chapter presents additional experiments designed to further analyze the role of feature representation choices in one-class visual anomaly detection. While the main experiments focus on features extracted from the global average pooling layer of a fine-tuned ResNet50 model, several alternative design choices remain possible. The experiments in this chapter investigate how different feature extraction strategies influence anomaly detection performance, while keeping the anomaly detection methods and evaluation protocol unchanged.

Specifically, four additional experiments are considered: (i) feature extraction from intermediate convolutional layers, (ii) concatenation of features from multiple layers, (iii) an analysis of the effect of feature normalization, and (iv) an evaluation of zero-shot anomaly detection using PatchCore. These experiments aim to provide a deeper understanding of how representation depth, feature composition, and normalization affect the behavior of classical one-class anomaly detection methods.

8.1 Effect of Feature Extraction Layer Depth

This experiment investigates how the choice of the feature extraction layer within the ResNet50 backbone influences anomaly detection performance. While the main experiments rely on features extracted from the global average pooling layer, convolutional neural networks learn hierarchical representations, and intermediate layers may encode different levels of semantic and spatial information.

To analyze this effect, feature representations are extracted from four different points in the ResNet50 architecture:

the GlobalAveragePooling2D(GAP) layer, `conv5_block3_out`, `conv4_block6_out`, and `conv3_block4_out`. For each representation, PCA, One-Class SVM, LOF, and Isolation Forest are applied using the same training and evaluation protocol as in the main experiments.

8.1.1 Analysis and Discussion

Tables 8.1, 8.2, 8.3, and 8.4 present a systematic comparison of anomaly detection performance across different feature extraction layers and classical one-class detectors. Several consistent trends emerge across datasets of increasing complexity.

Table 8.1: Layer-wise AUROC results for PCA. GAP denotes the global average pooling layer, while C5, C4, and C3 correspond to `conv5_block3_out`, `conv4_block6_out`, and `conv3_block4_out`, respectively.

Class pair	**GAP**	**C5**	**C4**	**C3**
MNIST (N:8, A:9)	0.9666	0.9666	**0.9820**	0.9461
MNIST (N:0, A:1)	0.9952	0.9952	0.9980	**0.9995**
MNIST (N:1, A:7)	0.9718	0.9718	0.9852	**0.9882**
MNIST (N:2, A:3)	0.7091	0.7091	0.7741	**0.8514**
MNIST Average	0.9107	0.9107	0.9348	**0.9463**
Fashion-MNIST (N:0, A:6)	0.7602	0.7602	**0.7765**	0.7575
Fashion-MNIST (N:2, A:4)	**0.9592**	0.9592	0.9520	0.8824
Fashion-MNIST (N:3, A:4)	0.8713	0.8713	**0.9032**	0.8827
Fashion-MNIST (N:5, A:7)	**0.6988**	0.6988	0.6138	0.5067
Fashion-MNIST Average	**0.8224**	0.8224	0.8114	0.7573
CIFAR-100 (N:0, A:29)	0.9689	0.9689	**0.9994**	0.9790
CIFAR-100 (N:6, A:14)	0.7241	0.7241	**0.8129**	0.7848
CIFAR-100 (N:45, A:49)	0.9772	0.9772	0.9177	**0.8245**
CIFAR-100 (N:30, A:95)	0.6539	0.6539	**0.6963**	0.6845
CIFAR-100 (N:29, A:30)	**0.9162**	0.9162	0.9101	0.8423
CIFAR-100 Average	0.8481	0.8481	**0.8673**	0.8230
Tiny-ImageNet (N:n02123394, A:n02124075)	**0.8960**	0.8960	0.8796	0.8336
Tiny-ImageNet (N:n02132136, A:n02125311)	**0.9876**	0.9876	0.9404	0.8832
Tiny-ImageNet (N:n01910747, A:n01950731)	**0.9964**	0.9964	0.9852	0.8800
Tiny-ImageNet (N:n04251144, A:n03838899)	**0.9784**	0.9784	0.9696	0.9048
Tiny-ImageNet (N:n02056570, A:n02190166)	**0.9196**	0.9196	0.8364	0.7312
Tiny-ImageNet (N:n02099601, A:n02106662)	**0.9496**	0.9496	0.8996	0.7716
Tiny-ImageNet (N:n03637318, A:n04501370)	**0.9552**	0.9552	0.9704	0.9556
Tiny-ImageNet Average	**0.9547**	0.9547	0.9259	0.8514

Table 8.2: Layer-wise AUROC results for OCSVM. GAP denotes the global average pooling layer, while C5, C4, and C3 correspond to `conv5_block3_out`, `conv4_block6_out`, and `conv3_block4_out`, respectively.

Class pair	**GAP**	**C5**	**C4**	**C3**
MNIST (N:8, A:9)	0.9662	0.9662	**0.9823**	0.9433
MNIST (N:0, A:1)	0.9958	0.9958	0.9982	**0.9996**
MNIST (N:1, A:7)	0.9736	0.9736	0.9898	**0.9942**
MNIST (N:2, A:3)	0.7207	0.7207	0.8296	**0.9206**
MNIST Average	0.9141	0.9141	0.9500	**0.9644**
Fashion-MNIST (N:0, A:6)	0.7543	0.7543	**0.7649**	0.7470
Fashion-MNIST (N:2, A:4)	**0.9569**	0.9569	0.9462	0.8703
Fashion-MNIST (N:3, A:4)	0.8709	0.8709	**0.9025**	0.8815
Fashion-MNIST (N:5, A:7)	**0.7098**	0.7098	0.6460	0.5600
Fashion-MNIST Average	**0.8229**	0.8229	0.8149	0.7647
CIFAR-100 (N:0, A:29)	0.9765	0.9765	**0.9996**	0.9849
CIFAR-100 (N:6, A:14)	0.7274	0.7274	**0.8244**	0.8040
CIFAR-100 (N:45, A:49)	**0.9772**	0.9772	0.9130	0.8407
CIFAR-100 (N:30, A:95)	0.6539	0.6539	**0.7077**	0.6862
CIFAR-100 (N:29, A:30)	**0.9165**	0.9165	0.9141	0.8312
CIFAR-100 Average	0.8503	0.8503	**0.8718**	0.8294
Tiny-ImageNet (N:n02123394, A:n02124075)	**0.8908**	0.8908	0.8804	0.8232
Tiny-ImageNet (N:n02132136, A:n02125311)	**0.9872**	0.9872	0.9556	0.9120
Tiny-ImageNet (N:n01910747, A:n01950731)	**0.9968**	0.9968	0.9876	0.8976
Tiny-ImageNet (N:n04251144, A:n03838899)	**0.9776**	0.9776	0.9700	0.9032
Tiny-ImageNet (N:n02056570, A:n02190166)	**0.9148**	0.9148	0.8220	0.7412
Tiny-ImageNet (N:n02099601, A:n02106662)	**0.9504**	0.9504	0.8848	0.7568
Tiny-ImageNet (N:n03637318, A:n04501370)	**0.9568**	0.9568	0.9700	0.9552
Tiny-ImageNet Average	**0.9535**	0.9535	0.9243	0.8556

Table 8.3: Layer-wise AUROC results for LOF. GAP denotes the global average pooling layer, while C5, C4, and C3 correspond to `conv5_block3_out`, `conv4_block6_out`, and `conv3_block4_out`, respectively.

Class pair	**GAP**	**C5**	**C4**	**C3**
MNIST (N:8, A:9)	0.9476	0.9476	**0.9636**	0.9435
MNIST (N:0, A:1)	0.9951	0.9951	0.9996	**0.9999**
MNIST (N:1, A:7)	0.9785	0.9785	0.9819	**0.9816**
MNIST (N:2, A:3)	0.8430	0.8430	**0.9638**	0.9895
MNIST Average	0.9411	0.9411	0.9772	**0.9786**
Fashion-MNIST (N:0, A:6)	0.7787	0.7787	**0.7926**	0.7895
Fashion-MNIST (N:2, A:4)	**0.9245**	0.9245	0.8602	0.7789
Fashion-MNIST (N:3, A:4)	0.8443	0.8443	**0.9126**	0.8998
Fashion-MNIST (N:5, A:7)	**0.7970**	0.7970	0.7642	0.7814
Fashion-MNIST Average	**0.8361**	0.8361	0.8324	0.8124
CIFAR-100 (N:0, A:29)	0.9673	0.9673	**0.9908**	0.9412
CIFAR-100 (N:6, A:14)	0.7310	0.7310	**0.8356**	0.8441
CIFAR-100 (N:45, A:49)	**0.9780**	0.9780	0.9361	0.8777
CIFAR-100 (N:30, A:95)	0.6572	0.6572	**0.6874**	0.7254
CIFAR-100 (N:29, A:30)	**0.9332**	0.9332	0.9316	0.8693
CIFAR-100 Average	0.8533	0.8533	**0.8763**	0.8515
Tiny-ImageNet (N:n02123394, A:n02124075)	**0.9212**	0.9212	0.9360	0.8708
Tiny-ImageNet (N:n02132136, A:n02125311)	**0.9892**	0.9892	0.9740	0.9468
Tiny-ImageNet (N:n01910747, A:n01950731)	**0.9976**	0.9976	0.9784	0.9140
Tiny-ImageNet (N:n04251144, A:n03838899)	**0.9832**	0.9832	0.9744	0.9268
Tiny-ImageNet (N:n02056570, A:n02190166)	**0.9288**	0.9288	0.9020	0.7692
Tiny-ImageNet (N:n02099601, A:n02106662)	**0.9496**	0.9496	0.9156	0.8028
Tiny-ImageNet (N:n03637318, A:n04501370)	0.9648	0.9648	**0.9808**	0.9596
Tiny-ImageNet Average	**0.9621**	0.9621	0.9516	0.8843

Table 8.4: Layer-wise AUROC results for Isolation Forest. GAP denotes the global average pooling layer, while C5, C4, and C3 correspond to `conv5_block3_out`, `conv4_block6_out`, and `conv3_block4_out`, respectively.

Class pair	**GAP**	**C5**	**C4**	**C3**
MNIST (N:8, A:9)	0.9472	0.9472	**0.9840**	0.9368
MNIST (N:0, A:1)	0.9854	0.9854	0.9964	**0.9988**
MNIST (N:1, A:7)	0.9454	0.9454	0.9815	**0.9872**
MNIST (N:2, A:3)	0.6167	0.6167	0.7429	**0.8780**
MNIST Average	0.8737	0.8737	0.9262	**0.9502**
Fashion-MNIST (N:0, A:6)	0.7087	0.7087	**0.7880**	0.7629
Fashion-MNIST (N:2, A:4)	0.9465	0.9465	**0.9576**	0.8860
Fashion-MNIST (N:3, A:4)	0.8576	0.8576	**0.9037**	0.8812
Fashion-MNIST (N:5, A:7)	**0.6608**	0.6608	0.6129	0.5246
Fashion-MNIST Average	**0.7934**	0.7934	0.8156	0.7637
CIFAR-100 (N:0, A:29)	0.9935	0.9935	**0.9995**	0.9741
CIFAR-100 (N:6, A:14)	0.6502	0.6502	**0.7655**	0.7871
CIFAR-100 (N:45, A:49)	**0.8676**	0.8676	0.8641	0.8304
CIFAR-100 (N:30, A:95)	0.6390	0.6390	**0.6701**	0.6947
CIFAR-100 (N:29, A:30)	**0.7153**	0.7153	0.8444	0.7992
CIFAR-100 Average	0.7731	0.7731	**0.8287**	0.8171
Tiny-ImageNet (N:n02123394, A:n02124075)	**0.8664**	0.8664	0.8036	0.8344
Tiny-ImageNet (N:n02132136, A:n02125311)	**0.9508**	0.9508	0.9400	0.8832
Tiny-ImageNet (N:n01910747, A:n01950731)	**0.9704**	0.9704	0.9888	0.8612
Tiny-ImageNet (N:n04251144, A:n03838899)	**0.9568**	0.9568	0.9544	0.8832
Tiny-ImageNet (N:n02056570, A:n02190166)	**0.8648**	0.8648	0.8072	0.7304
Tiny-ImageNet (N:n02099601, A:n02106662)	**0.9496**	0.9496	0.8828	0.7844
Tiny-ImageNet (N:n03637318, A:n04501370)	0.9196	0.9196	**0.9708**	0.9540
Tiny-ImageNet Average	**0.9255**	0.9255	0.9068	0.8473

For simple datasets such as MNIST, shallower convolutional features (C3) generally yield the strongest or near-strongest performance across methods. This suggests that relatively low-level spatial information captured by shallow convolutional features is sufficient to characterize normality for digit-based datasets. In this setting, deeper semantic representations provide limited additional benefit and may even obscure fine-grained pixel-level differences.

In contrast, for more complex datasets such as CIFAR-100 and Tiny-ImageNet, intermediate convolutional features (C4) consistently achieve the highest AUROC scores

across PCA, OCSVM, LOF, and Isolation Forest. These layers appear to offer a favorable balance between semantic abstraction and spatial resolution, capturing object parts and textures while preserving locality. Very deep representations (GAP/C5) tend to discard spatial diversity, whereas shallow features (C3) lack sufficient semantic discriminability, resulting in degraded performance.

GAP versus Convolutional Features Global average pooled (GAP) features remain competitive on several class pairs, particularly for PCA and OCSVM, but are rarely optimal for the most challenging datasets. By collapsing spatial information into a single global descriptor, GAP representations can limit the effectiveness of local density-based methods such as LOF and partition-based approaches such as Isolation Forest. This limitation becomes more pronounced as dataset complexity increases.

Method-specific Behavior PCA and OCSVM exhibit relatively stable performance across layers and datasets, consistent with their reliance on global variance structure and margin-based separation, respectively. In contrast, LOF and Isolation Forest show greater sensitivity to feature depth, benefiting more strongly from intermediate convolutional representations where local neighborhoods remain meaningful.

Dataset-level Trends As the datasets become more complex from MNIST to Fashion-MNIST, CIFAR-100, and Tiny-ImageNet, a consistent trend can be observed. Simple digit datasets can be handled well using shallow feature representations, whereas more complex natural image datasets require deeper convolutional features to reliably separate normal and anomalous samples. However, using features that are too deep or heavily pooled can remove important spatial details, which can reduce the effectiveness of one-class anomaly detection methods.

Overall, the results demonstrate that anomaly detection performance is governed by the interaction between feature depth, dataset complexity, and detector assumptions. Intermediate convolutional layers provide the most reliable representations across detectors for complex natural image datasets, whereas shallow layers remain optimal for simpler domains. These findings highlight the importance of selecting feature representations that align with both the data distribution and the underlying mechanisms of the anomaly detection method.

8.2 Effect of Multi-layer Feature Combination

This experiment investigates whether combining feature representations from multiple convolutional layers can improve anomaly detection performance compared to using a single feature extraction layer. While individual layers capture information at a specific level of abstraction, concatenating features from different depths may provide complementary spatial and semantic cues that are beneficial for one-class anomaly detection.

To analyze this effect, feature representations are constructed by concatenating outputs from multiple convolutional layers within the ResNet50 backbone. The evaluated

combinations include `conv3_block4_out` with `conv4_block6_out`, `conv4_block6_out` with `conv5_block3_out`, and the combination of `conv3_block4_out`, `conv4_block6_out`, and `conv5_block3_out`. For each feature combination, PCA, One-Class SVM, LOF, and Isolation Forest are applied using the same training and evaluation protocol as in the main experiments.

8.2.1 Analysis and Discussion

The results of the multi-layer feature combination experiments are summarized in Tables 8.5, 8.6, 8.7, and 8.8. The following observations highlight the main trends across datasets and anomaly detection methods.

Table 8.5: AUROC results for PCA using multi-layer feature combinations. C3, C4, and C5 correspond to `conv3_block4_out`, `conv4_block6_out`, and `conv5_block3_out`, respectively.

Class pair	**C3 + C4**	**C4 + C5**	**C3 + C4 + C5**
MNIST (N:8, A:9)	**0.9802**	0.9773	0.9735
MNIST (N:0, A:1)	**0.9982**	0.9954	0.9954
MNIST (N:1, A:7)	**0.9868**	0.9801	0.9823
MNIST (N:2, A:3)	**0.8071**	0.7536	0.7742
MNIST Average	**0.9431**	0.9266	0.9314
Fashion-MNIST (N:0, A:6)	0.7769	0.7740	**0.7791**
Fashion-MNIST (N:2, A:4)	0.9438	**0.9604**	0.9593
Fashion-MNIST (N:3, A:4)	**0.9036**	0.8864	0.8900
Fashion-MNIST (N:5, A:7)	0.5832	**0.6767**	0.6582
Fashion-MNIST Average	0.8019	**0.8244**	0.8217
CIFAR-100 (N:0, A:29)	**0.9995**	0.9693	0.9693
CIFAR-100 (N:6, A:14)	**0.8335**	0.7486	0.7619
CIFAR-100 (N:45, A:49)	0.9129	0.9758	**0.9763**
CIFAR-100 (N:30, A:95)	**0.7070**	0.6672	0.6726
CIFAR-100 (N:29, A:30)	0.9115	0.9248	**0.9280**
CIFAR-100 Average	**0.8729**	0.8571	0.8616
Tiny-ImageNet (N:n02123394, A:n02124075)	0.8828	**0.9108**	0.9108
Tiny-ImageNet (N:n02132136, A:n02125311)	0.9288	**0.9872**	0.9848
Tiny-ImageNet (N:n01910747, A:n01950731)	0.9780	0.9976	**0.9980**
Tiny-ImageNet (N:n04251144, A:n03838899)	0.9596	**0.9804**	0.9804
Tiny-ImageNet (N:n02056570, A:n02190166)	0.8212	**0.9152**	0.9116
Tiny-ImageNet (N:n02099601, A:n02106662)	0.8848	**0.9408**	0.9400
Tiny-ImageNet (N:n03637318, A:n04501370)	**0.9704**	0.9628	0.9640
Tiny-ImageNet Average	0.9180	**0.9564**	0.9557

Table 8.6: AUROC results for One-Class SVM using multi-layer feature combinations. C3, C4, and C5 correspond to `conv3_block4_out`, `conv4_block6_out`, and `conv5_block3_out`, respectively.

Class pair	**C3 + C4**	**C4 + C5**	**C3 + C4 + C5**
MNIST (N:8, A:9)	**0.9821**	0.9794	0.9766
MNIST (N:0, A:1)	**0.9984**	0.9960	0.9962
MNIST (N:1, A:7)	**0.9901**	0.9850	0.9864
MNIST (N:2, A:3)	**0.8476**	0.7984	0.8121
MNIST Average	**0.9546**	0.9397	0.9428
Fashion-MNIST (N:0, A:6)	0.7680	0.7705	**0.7723**
Fashion-MNIST (N:2, A:4)	0.9516	**0.9574**	0.9568
Fashion-MNIST (N:3, A:4)	**0.9041**	0.8897	0.8930
Fashion-MNIST (N:5, A:7)	0.6268	**0.6904**	0.6749
Fashion-MNIST Average	0.8126	**0.8270**	0.8243
CIFAR-100 (N:0, A:29)	**0.9996**	0.9769	0.9771
CIFAR-100 (N:6, A:14)	**0.8458**	0.7643	0.7812
CIFAR-100 (N:45, A:49)	0.9187	0.9776	**0.9782**
CIFAR-100 (N:30, A:95)	**0.7192**	0.6836	0.6898
CIFAR-100 (N:29, A:30)	0.9176	0.9269	**0.9310**
CIFAR-100 Average	**0.8802**	0.8659	0.8715
Tiny-ImageNet (N:n02123394, A:n02124075)	0.8856	**0.9136**	0.9132
Tiny-ImageNet (N:n02132136, A:n02125311)	0.9312	**0.9876**	0.9856
Tiny-ImageNet (N:n01910747, A:n01950731)	0.9792	0.9976	**0.9984**
Tiny-ImageNet (N:n04251144, A:n03838899)	0.9624	**0.9816**	0.9812
Tiny-ImageNet (N:n02056570, A:n02190166)	0.8268	**0.9188**	0.9156
Tiny-ImageNet (N:n02099601, A:n02106662)	0.8896	**0.9424**	0.9416
Tiny-ImageNet (N:n03637318, A:n04501370)	**0.9720**	0.9644	0.9656
Tiny-ImageNet Average	0.9209	**0.9579**	0.9573

Table 8.7: AUROC results for LOF using multi-layer feature combinations. C3, C4, and C5 correspond to `conv3_block4_out`, `conv4_block6_out`, and `conv5_block3_out`, respectively.

Class pair	**C3 + C4**	**C4 + C5**	**C3 + C4 + C5**
MNIST (N:8, A:9)	**0.9699**	0.9641	0.9696
MNIST (N:0, A:1)	**0.9998**	0.9953	0.9954
MNIST (N:1, A:7)	0.9803	**0.9828**	0.9816
MNIST (N:2, A:3)	**0.9768**	0.8931	0.9091
MNIST Average	**0.9817**	0.9588	0.9639
Fashion-MNIST (N:0, A:6)	**0.7932**	0.7862	0.7915
Fashion-MNIST (N:2, A:4)	0.8601	0.9312	**0.9348**
Fashion-MNIST (N:3, A:4)	**0.9154**	0.8737	0.8810
Fashion-MNIST (N:5, A:7)	0.7746	0.7971	**0.8019**
Fashion-MNIST Average	0.8358	0.8470	**0.8523**
CIFAR-100 (N:0, A:29)	**0.9893**	0.9677	0.9678
CIFAR-100 (N:6, A:14)	**0.8546**	0.7569	0.7698
CIFAR-100 (N:45, A:49)	0.9296	**0.9787**	0.9785
CIFAR-100 (N:30, A:95)	**0.7065**	0.6660	0.6688
CIFAR-100 (N:29, A:30)	0.9263	0.9440	**0.9460**
CIFAR-100 Average	**0.8813**	0.8627	0.8662
Tiny-ImageNet (N:n02123394, A:n02124075)	**0.9416**	0.9276	0.9316
Tiny-ImageNet (N:n02132136, A:n02125311)	0.9748	0.9900	**0.9912**
Tiny-ImageNet (N:n01910747, A:n01950731)	0.9812	**0.9980**	0.9976
Tiny-ImageNet (N:n04251144, A:n03838899)	0.9724	**0.9832**	0.9824
Tiny-ImageNet (N:n02056570, A:n02190166)	0.8636	**0.9372**	0.9348
Tiny-ImageNet (N:n02099601, A:n02106662)	0.9040	**0.9512**	**0.9512**
Tiny-ImageNet (N:n03637318, A:n04501370)	**0.9816**	0.9688	0.9720
Tiny-ImageNet Average	0.9456	0.9651	**0.9658**

Table 8.8: AUROC results for Isolation Forest using multi-layer feature combinations. C3, C4, and C5 correspond to `conv3_block4_out`, `conv4_block6_out`, and `conv5_block3_out`, respectively.

Class pair	**C3 + C4**	**C4 + C5**	**C3 + C4 + C5**
MNIST (N:8, A:9)	**0.9794**	0.9749	0.9769
MNIST (N:0, A:1)	**0.9976**	0.9956	0.9914
MNIST (N:1, A:7)	**0.9834**	0.9812	0.9766
MNIST (N:2, A:3)	**0.8183**	0.6804	0.7532
MNIST Average	**0.9447**	0.9080	0.9245
Fashion-MNIST (N:0, A:6)	**0.7820**	0.7589	0.7705
Fashion-MNIST (N:2, A:4)	0.9398	**0.9653**	0.9466
Fashion-MNIST (N:3, A:4)	0.8922	0.8740	**0.9049**
Fashion-MNIST (N:5, A:7)	0.5694	**0.6558**	0.6245
Fashion-MNIST Average	0.7958	**0.8135**	0.8116
CIFAR-100 (N:0, A:29)	0.9967	**0.9995**	0.9991
CIFAR-100 (N:6, A:14)	**0.8136**	0.6922	0.7503
CIFAR-100 (N:45, A:49)	0.8497	0.8809	**0.8874**
CIFAR-100 (N:30, A:95)	**0.7049**	0.6612	0.6718
CIFAR-100 (N:29, A:30)	**0.8378**	0.8188	0.8052
CIFAR-100 Average	**0.8405**	0.8105	0.8228
Tiny-ImageNet (N:n02123394, A:n02124075)	0.8436	**0.9168**	0.8824
Tiny-ImageNet (N:n02132136, A:n02125311)	0.9356	0.9596	**0.9652**
Tiny-ImageNet (N:n01910747, A:n01950731)	0.9788	0.9788	**0.9944**
Tiny-ImageNet (N:n04251144, A:n03838899)	0.9376	**0.9596**	0.9572
Tiny-ImageNet (N:n02056570, A:n02190166)	0.8124	**0.9020**	0.8796
Tiny-ImageNet (N:n02099601, A:n02106662)	0.8380	**0.9232**	0.9192
Tiny-ImageNet (N:n03637318, A:n04501370)	**0.9628**	0.9416	0.9488
Tiny-ImageNet Average	0.9013	**0.9402**	0.9353

Performance on Simple Datasets For more complex datasets, including CIFAR-100 and Tiny-ImageNet, multi-layer feature combinations can improve or stabilize anomaly detection performance. By aggregating features from different depths, these representations may combine low-level spatial cues with higher-level semantic information, resulting in a richer feature space that supports more reliable separation between normal and anomalous samples.

Performance on Complex Datasets For more complex datasets, including CIFAR-100 and Tiny-ImageNet, multi-layer feature combinations often improve or stabilize anomaly detection performance. By aggregating features from different depths, these representations may combine low-level spatial cues with higher-level semantic information, aggregated representations form a richer feature space that supports more reliable separation between normal and anomalous samples.

Method-specific Behavior Differences across anomaly detection methods are also observed. LOF benefits most consistently from feature aggregation, reflecting its reliance on meaningful local neighborhood structures in the feature space. PCA and One-Class SVM exhibit moderate and generally consistent improvements across feature combinations. In contrast, Isolation Forest shows more variable performance, suggesting increased sensitivity to higher feature dimensionality.

Overall, these results indicate that multi-layer feature combination can be advantageous for complex natural image datasets by leveraging complementary information across feature depths. However, the performance gains are not uniform across datasets or methods, and the increased dimensionality may introduce redundancy or instability for certain detectors. Consequently, while feature aggregation can enhance performance in some scenarios, carefully selected single-layer representations remain a competitive and more parsimonious choice for general-purpose anomaly detection.

8.3 Effect of Feature Normalization

This experiment investigates the impact of feature normalization on anomaly detection performance. In all main experiments, standard normalization is applied to the extracted feature representations, and the corresponding results are reported in the main evaluation tables. In this additional analysis, two alternative normalization strategies are considered: standard normalization followed by ℓ_2 normalization, and no normalization. The goal is to assess how different normalization choices influence the behavior of classical one-class anomaly detection methods.

For this comparison, anomaly detection is performed using the same feature representations, class pairs, and evaluation protocol as in the main experiments.

8.3.1 Analysis and Discussion

The results of the normalization study are summarized in Table 8.9,8.10,8.11 and 8.12. Several consistent patterns emerge across datasets and anomaly detection methods.

Table 8.9: AUROC results for **PCA** under different feature normalizations.

Dataset / Pair	**Standard**	**Standard + ℓ_2**	**No Norm.**
MNIST(N:8 A:9)	**0.9666**	0.9641	**0.9666**
MNIST(N:0 A:1)	**0.9952**	0.9842	**0.9952**
MNIST(N:1 A:7)	**0.9718**	0.9735	**0.9718**
MNIST(N:2 A:3)	**0.7091**	0.6894	**0.7091**
Avg. (MNIST)	**0.9107**	0.9028	**0.9107**
Fashion-MNIST(N:0 A:6)	**0.7602**	0.7928	**0.7602**
Fashion-MNIST(N:2 A:4)	**0.9592**	0.9465	**0.9592**
Fashion-MNIST(N:3 A:4)	**0.8713**	0.8263	**0.8713**
Fashion-MNIST(N:5 A:7)	**0.6988**	0.6945	**0.6988**
Avg. (Fashion-MNIST)	**0.8224**	0.8150	**0.8224**
CIFAR-100(N:0 A:29)	**0.9689**	0.9352	**0.9689**
CIFAR-100(N:6 A:14)	**0.7241**	0.6367	**0.7241**
CIFAR-100(N:45 A:49)	**0.9772**	0.9690	**0.9772**
CIFAR-100(N:30 A:95)	**0.6539**	0.6382	**0.6539**
CIFAR-100(N:29 A:30)	**0.9162**	0.8895	**0.9162**
Avg. (CIFAR-100)	**0.8481**	0.8137	**0.8481**
Tiny-ImageNet (N:n02123394, A:n02124075)	**0.8960**	0.9020	**0.8960**
Tiny-ImageNet (N:n02132136, A:n02125311)	**0.9876**	0.9464	**0.9876**
Tiny-ImageNet (N:n01910747, A:n01950731)	**0.9964**	0.9764	**0.9964**
Tiny-ImageNet (N:n04251144, A:n03838899)	**0.9784**	0.9604	**0.9784**
Tiny-ImageNet (N:n02056570, A:n02190166)	**0.9196**	0.8832	**0.9196**
Tiny-ImageNet (N:n02099601, A:n02106662)	**0.9496**	0.9512	**0.9496**
Tiny-ImageNet (N:n03637318, A:n04501370)	**0.9552**	0.9376	**0.9552**
Avg. (Tiny-ImageNet)	**0.9547**	0.9367	**0.9547**

Table 8.10: AUROC results for **OCSVM** under different feature normalizations.

Dataset / Pair	**Standard**	**Standard + ℓ_2**	**No Norm.**
MNIST(N:8 A:9)	**0.9662**	0.9453	0.9230
MNIST(N:0 A:1)	**0.9958**	0.9917	0.9711
MNIST(N:1 A:7)	**0.9736**	0.9714	0.9641
MNIST(N:2 A:3)	0.7207	**0.8584**	0.6301
Avg. (MNIST)	0.9141	**0.9417**	0.8721
Fashion-MNIST(N:0 A:6)	**0.7543**	0.6405	0.7236
Fashion-MNIST(N:2 A:4)	**0.9569**	0.9053	0.9405
Fashion-MNIST(N:3 A:4)	**0.8709**	0.8028	0.8626
Fashion-MNIST(N:5 A:7)	0.7098	**0.7258**	0.7105
Avg. (Fashion-MNIST)	**0.8230**	0.7686	0.8093
CIFAR-100(N:0 A:29)	0.9765	0.9692	**0.9766**
CIFAR-100(N:6 A:14)	0.7274	**0.7522**	0.6382
CIFAR-100(N:45 A:49)	**0.9772**	0.9734	0.6198
CIFAR-100(N:30 A:95)	0.6539	**0.6938**	0.5954
CIFAR-100(N:29 A:30)	0.9165	**0.9364**	0.7134
Avg. (CIFAR-100)	0.8503	**0.8650**	0.7087
Tiny-ImageNet (N:n02123394, A:n02124075)	**0.8908**	0.8480	0.6480
Tiny-ImageNet (N:n02132136, A:n02125311)	**0.9872**	0.9388	0.9080
Tiny-ImageNet (N:n01910747, A:n01950731)	**0.9968**	0.9860	0.9556
Tiny-ImageNet (N:n04251144, A:n03838899)	**0.9776**	0.9436	0.9136
Tiny-ImageNet (N:n02056570, A:n02190166)	**0.9148**	0.8084	0.6852
Tiny-ImageNet (N:n02099601, A:n02106662)	**0.9504**	0.9036	0.8828
Tiny-ImageNet (N:n03637318, A:n04501370)	0.9568	**0.9640**	0.8224
Avg. (Tiny-ImageNet)	**0.9535**	0.9132	0.8308

Table 8.11: AUROC results for **LOF** under different feature normalizations.

Dataset / Pair	**Standard**	**Standard + ℓ_2**	**No Norm.**
MNIST(N:8 A:9)	**0.9476**	0.7644	0.7928
MNIST(N:0 A:1)	**0.9951**	0.9692	0.9944
MNIST(N:1 A:7)	**0.9785**	0.8718	0.9286
MNIST(N:2 A:3)	0.8430	0.8615	**0.9317**
Avg. (MNIST)	**0.9410**	0.8667	0.9119
Fashion-MNIST(N:0 A:6)	**0.7787**	0.6257	0.7578
Fashion-MNIST(N:2 A:4)	**0.9245**	0.8514	0.9177
Fashion-MNIST(N:3 A:4)	0.8443	0.7134	**0.8977**
Fashion-MNIST(N:5 A:7)	**0.7970**	0.7897	0.7937
Avg. (Fashion-MNIST)	**0.8361**	0.7450	0.8417
CIFAR-100(N:0 A:29)	**0.9673**	0.7509	0.9279
CIFAR-100(N:6 A:14)	**0.7310**	0.7246	0.6986
CIFAR-100(N:45 A:49)	**0.9780**	0.8855	0.7584
CIFAR-100(N:30 A:95)	0.6572	**0.6687**	0.6662
CIFAR-100(N:29 A:30)	**0.9332**	0.8447	0.8637
Avg. (CIFAR-100)	**0.8533**	0.7749	0.7830
Tiny-ImageNet (N:n02123394, A:n02124075)	**0.9212**	0.7056	0.8056
Tiny-ImageNet (N:n02132136, A:n02125311)	**0.9892**	0.7772	0.9508
Tiny-ImageNet (N:n01910747, A:n01950731)	**0.9976**	0.6156	0.9852
Tiny-ImageNet (N:n04251144, A:n03838899)	**0.9832**	0.7596	0.9480
Tiny-ImageNet (N:n02056570, A:n02190166)	**0.9288**	0.7514	0.7500
Tiny-ImageNet (N:n02099601, A:n02106662)	**0.9496**	0.5296	0.9288
Tiny-ImageNet (N:n03637318, A:n04501370)	**0.9648**	0.7260	0.8448
Avg. (Tiny-ImageNet)	**0.9621**	0.6950	0.8876

Table 8.12: AUROC results for **Isolation Forest** under different feature normalizations.

Dataset / Pair	Standard	Standard + ℓ_2	No Norm.
MNIST(N:8 A:9)	**0.9472**	0.9295	**0.9472**
MNIST(N:0 A:1)	**0.9854**	0.9653	**0.9854**
MNIST(N:1 A:7)	**0.9454**	0.9215	**0.9454**
MNIST(N:2 A:3)	**0.6167**	0.4922	**0.6167**
Avg. (MNIST)	**0.8737**	0.8271	**0.8737**
Fashion-MNIST(N:0 A:6)	**0.7087**	0.6029	**0.7087**
Fashion-MNIST(N:2 A:4)	**0.9465**	0.9117	**0.9465**
Fashion-MNIST(N:3 A:4)	**0.8576**	0.7524	**0.8576**
Fashion-MNIST(N:5 A:7)	**0.6608**	0.5536	**0.6608**
Avg. (Fashion-MNIST)	**0.7934**	0.7051	**0.7934**
CIFAR-100(N:0 A:29)	**0.9935**	0.8649	**0.9935**
CIFAR-100(N:6 A:14)	**0.6502**	0.4979	**0.6502**
CIFAR-100(N:45 A:49)	**0.8676**	0.7123	**0.8676**
CIFAR-100(N:30 A:95)	**0.6390**	0.4824	**0.6390**
CIFAR-100(N:29 A:30)	**0.7153**	0.4479	**0.7153**
Avg. (CIFAR-100)	**0.7731**	0.6011	**0.7731**
Tiny-ImageNet (N:n02123394, A:n02124075)	**0.8664**	0.7248	**0.8664**
Tiny-ImageNet (N:n02132136, A:n02125311)	**0.9508**	0.6332	**0.9508**
Tiny-ImageNet (N:n01910747, A:n01950731)	**0.9704**	0.7036	**0.9704**
Tiny-ImageNet (N:n04251144, A:n03838899)	**0.9568**	0.5884	**0.9568**
Tiny-ImageNet (N:n02056570, A:n02190166)	**0.8648**	0.5324	**0.8648**
Tiny-ImageNet (N:n02099601, A:n02106662)	**0.9496**	0.8272	**0.9496**
Tiny-ImageNet (N:n03637318, A:n04501370)	**0.9196**	0.8096	**0.9196**
Avg. (Tiny-ImageNet)	**0.9255**	0.6885	**0.9255**

Effect of Standard Normalization Standard normalization yields consistently strong and stable performance across datasets and anomaly detection methods. For PCA and Isolation Forest, performance under standard normalization is numerically identical to that obtained without normalization across all datasets, indicating invariance to feature scaling in these settings. For OCSVM and LOF, standard normalization provides the most reliable average performance, particularly on CIFAR-100 and Tiny-ImageNet. These results justify the use of standard normalization as the default preprocessing strategy in the main experiments.

Effect of Standard + ℓ_2 Normalization Applying ℓ_2 normalization after standard normalization leads to method-dependent performance changes. PCA consistently exhibits slight performance degradation, while LOF shows substantial drops in average AUROC,

especially on CIFAR-100 and Tiny-ImageNet. OCSVM achieves modest improvements for some class pairs but does not show consistent gains overall. Isolation Forest does not benefit systematically from ℓ_2 normalization, with performance remaining comparable or slightly reduced. These results suggest that enforcing unit-length feature vectors may suppress magnitude-related information that is relevant for distance- and density-based anomaly detection, although the precise impact depends on the detector and dataset.

Effect of No Normalization Omitting normalization entirely results in mixed behavior across methods. For PCA and Isolation Forest, average performance without normalization is identical to that obtained with standard normalization across all datasets, indicating robustness to feature scaling in these settings. In contrast, OCSVM and LOF exhibit noticeable performance degradation and increased variability without normalization, particularly on more complex datasets such as CIFAR-100 and Tiny-ImageNet. This sensitivity reflects their reliance on meaningful distance relationships in the feature space, which can be distorted when feature scales differ substantially.

8.4 Zero-Shot Anomaly Detection with PatchCore

In addition to the feature-based anomaly detection experiments presented earlier, this section evaluates PatchCore [39] as a representative zero-shot anomaly detection method. Zero-shot anomaly detection aims to identify anomalous samples without task-specific training or fine-tuning on the target dataset. Instead, such methods rely on feature representations learned from large-scale datasets, typically ImageNet, and apply distance-based or memory-based scoring mechanisms to model normality.

8.4.1 Methodology

In this experiment, PatchCore is implemented using a ResNet50 backbone pretrained on ImageNet. The network is kept fully frozen to preserve the zero-shot setting. Feature maps are extracted from the `conv5_block3_out` layer, which provides high-level semantic representations while retaining spatial resolution required for patch-level analysis.

Input images are resized to a fixed resolution independent of the classification models used in earlier experiments and are preprocessed using the standard ResNet50 preprocessing pipeline. For grayscale datasets such as MNIST and Fashion-MNIST, images are converted to three-channel RGB format prior to feature extraction.

Patch-level features are obtained by flattening the spatial dimensions of the extracted feature maps. To reduce memory consumption and computational cost, principal component analysis (PCA) is applied to the patch features, followed by ℓ_2 normalization. A subset of the resulting features is then selected using random coreset sampling to construct the memory bank of normal patches.

At inference time, patch-level distances between test samples and the memory bank are computed using a k-nearest neighbor search. Patch anomaly scores are aggregated into an image-level anomaly score using a top-k mean strategy, which emphasizes the most

anomalous regions while remaining robust to noise. Anomaly detection performance is evaluated using the area under the ROC curve (AUROC). Importantly, no anomaly samples are used during memory construction, and no dataset-specific training or fine-tuning is performed.

8.4.2 Results and Comparison

The results of the PatchCore experiment are summarized in Table 8.13. PatchCore achieves strong performance on several class pairs, demonstrating that pretrained deep features can effectively characterize normality even without task-specific adaptation, particularly when normal and anomalous classes exhibit clear semantic or visual differences.

Table 8.13: Comparison of zero-shot PatchCore and classical one-class anomaly detection methods using ResNet50-based GAP features (AUROC). Classical methods are trained on normal samples, while PatchCore operates in a zero-shot setting with a frozen ImageNet-pretrained backbone.

Dataset / Class Pair	**PCA**	**OCSVM**	**LOF**	**IForest**	**PatchCore**
MNIST (N:8, A:9)	**0.9666**	0.9662	0.9476	0.9472	0.9324
MNIST (N:0, A:1)	0.9952	0.9958	0.9951	0.9854	**0.9962**
MNIST (N:1, A:7)	0.9718	0.9736	0.9785	0.9454	**0.9877**
MNIST (N:2, A:3)	0.7091	0.7207	**0.8430**	0.6167	0.7534
MNIST Average	0.9107	0.9141	**0.9410**	0.8737	0.9174
Fashion-MNIST (N:0, A:6)	0.7602	0.7543	**0.7787**	0.7087	0.7605
Fashion-MNIST (N:2, A:4)	**0.9592**	0.9569	0.9245	0.9465	0.6069
Fashion-MNIST (N:3, A:4)	0.8713	0.8709	0.8443	0.8576	**0.8781**
Fashion-MNIST (N:5, A:7)	0.6988	0.7098	**0.7970**	0.6608	0.6064
Fashion-MNIST Average	0.8224	0.8230	**0.8361**	0.7934	0.7129
CIFAR-100 (N:0, A:29)	0.9689	0.9765	0.9673	**0.9935**	0.9502
CIFAR-100 (N:6, A:14)	0.7241	0.7274	0.7310	0.6502	**0.8234**
CIFAR-100 (N:45, A:49)	0.9772	0.9772	**0.9780**	0.8676	0.6174
CIFAR-100 (N:30, A:95)	0.6539	0.6539	**0.6572**	0.6390	0.5359
CIFAR-100 (N:29, A:30)	0.9162	0.9165	**0.9332**	0.7153	0.7913
CIFAR-100 Average	0.8481	0.8503	**0.8533**	0.7731	0.7436
Tiny-ImageNet (N:n02123394, A:n02124075)	0.8960	0.8908	**0.9212**	0.8664	0.8475
Tiny-ImageNet (N:n02132136, A:n02125311)	0.9876	0.9872	**0.9892**	0.9508	0.8792
Tiny-ImageNet (N:n01910747, A:n01950731)	0.9964	0.9968	0.9976	0.9704	**0.9991**
Tiny-ImageNet (N:n04251144, A:n03838899)	0.9784	0.9776	**0.9832**	0.9568	0.7990
Tiny-ImageNet (N:n02056570, A:n02190166)	0.9196	0.9148	**0.9288**	0.8648	0.8168
Tiny-ImageNet (N:n02099601, A:n02106662)	0.9496	**0.9504**	0.9496	0.9496	0.6100
Tiny-ImageNet (N:n03637318, A:n04501370)	0.9552	0.9568	**0.9648**	0.9196	0.5292
Tiny-ImageNet Average	0.9547	0.9535	**0.9621**	0.9255	0.7830

PatchCore achieves competitive performance on selected class pairs and serves as a strong zero-shot baseline for anomaly detection. However, under the experimental conditions considered in this thesis, it is less effective than the proposed feature-based anomaly detection approach overall. Across all datasets, classical one-class detectors operating on GAP features consistently achieve higher average AUROC scores and exhibit greater stability across different normal–anomaly class pairs.

On MNIST, PatchCore performs comparably to feature-based methods but is surpassed by PCA, OCSVM, and LOF on average, indicating that for low-complexity data, compact global feature representations are sufficient for effective anomaly detection. For more complex datasets such as CIFAR-100 and Tiny-ImageNet, PatchCore shows strong performance on a small number of class pairs but suffers from substantial performance drops on others, resulting in lower average performance and higher variance compared to feature-based methods.

Overall, these results indicate that while PatchCore provides a valuable training-free reference point, feature-based anomaly detection methods leveraging task-adapted global representations offer superior average performance and robustness under constrained computational settings. This confirms that representation quality, rather than patch-level modeling alone, plays a central role in achieving reliable anomaly detection performance across diverse datasets.

9 Conclusion and Future Work

9.1 Summary of Findings

This thesis investigated the effectiveness of classical one-class anomaly detection methods when combined with deep, pretrained convolutional neural network features. Through a systematic evaluation on MNIST, Fashion-MNIST, CIFAR-100, and Tiny-ImageNet, the study examined how representation quality influences anomaly detection performance across datasets of increasing visual complexity.

The experimental results demonstrate that classical anomaly detection algorithms such as Principal Component Analysis (PCA), One-Class Support Vector Machine (OCSVM), Local Outlier Factor (LOF), and Isolation Forest perform significantly better when applied to features extracted from a deep convolutional neural network rather than directly on raw input representations. In particular, features obtained from a fine-tuned ResNet50 model provided a structured and discriminative embedding space, enabling classical detectors to model normal data distributions effectively.

Across all evaluated datasets, feature-based anomaly detection consistently achieved higher and more stable AUROC scores compared to raw pixel-based approaches. While raw representations occasionally produced acceptable results on simple datasets such as MNIST, their performance degraded substantially as dataset complexity increased. On Fashion-MNIST, CIFAR-100, and especially Tiny-ImageNet, raw input representations resulted in unstable and often poor anomaly detection performance. In contrast, deep feature representations maintained strong performance even on complex and fine-grained class pairs.

Among the evaluated anomaly detection methods, PCA and OCSVM demonstrated the most consistent and robust performance across datasets and class pairs. LOF and Isolation Forest achieved competitive results but showed greater sensitivity to dataset characteristics and feature scaling. These observations indicate that no single anomaly detection algorithm is universally optimal; instead, performance depends strongly on the quality of the underlying feature representation.

The additional experiments further emphasize the central role of feature representation choices in one-class visual anomaly detection. The results show that the optimal feature extraction depth depends on dataset complexity, with shallow convolutional features being sufficient for simple datasets such as MNIST, while intermediate convolutional layers provide more reliable performance on complex datasets such as CIFAR-100 and Tiny-ImageNet. Multi-layer feature aggregation can improve or stabilize performance in complex settings by combining complementary spatial and semantic information, although the benefits are not uniform across detectors and class pairs. The normalization anal-

ysis confirms that standard normalization is the most consistent preprocessing choice, whereas additional ℓ_2 normalization yields mixed effects and can degrade performance on complex natural images. Additionally, under the experimental conditions considered in this thesis, the proposed feature-based anomaly detection approach achieves higher accuracy and more robust performance than PatchCore.

9.2 Limitations

Despite its contributions, this work has several limitations. First, the study focuses on image datasets and does not consider other data modalities such as video, time series, or multimodal inputs. Second, the evaluation is restricted to a limited set of classical detectors and does not include recent end-to-end deep anomaly detection architectures. Third, although multiple normal–anomaly class pairs are evaluated, the selection is not exhaustive and may not capture all forms of intra-class variability.

Additionally, the use of fixed hyperparameters for OCSVM and iForest prioritizes comparability over optimal tuning and may underestimate its best achievable performance in certain settings.

9.3 Future Work

Several promising directions emerge from this study. Future work could extend the analysis to additional data modalities and real-world industrial datasets, where anomaly detection is often mission-critical. Incorporating temporal or contextual information, particularly for video-based anomaly detection, is another natural extension.

From a methodological perspective, future research could explore hybrid approaches that combine classical detectors with lightweight deep adaptation layers, balancing interpretability and performance. Investigating self-supervised or contrastive pretraining strategies may further improve feature quality without relying on labeled data.

Finally, a deeper theoretical analysis of why certain feature spaces favor specific anomaly detection mechanisms could provide valuable insights and guide the design of more principled and robust anomaly detection pipelines.

In conclusion, this thesis demonstrates that classical anomaly detection methods remain highly competitive when paired with strong deep feature extractors. Despite their simplicity, lower computational cost, and higher interpretability, these methods can achieve performance comparable to more complex deep anomaly detection frameworks when operating in a well-structured feature space.

Bibliography

[1] C. C. Aggarwal. 2017. *Outlier Analysis* (2 ed.). Springer. doi:10.1007/978-1-4614-6396-2

[2] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. 2001. On the Surprising Behavior of Distance Metrics in High Dimensional Space. *Proceedings of the International Conference on Database Theory (ICDT)* (2001), 420–434.

[3] N. Al-Humaidan and M. Prince. 2021. A Classification of Arab Ethnicity Based on Face Image Using Deep Learning Approach. *IEEE Access* PP (03 2021), 1–1. doi:10.1109/ACCESS.2021.3069022

[4] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger. 2019. MVTec AD—A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] C. M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer, New York, NY.

[6] Y.-L. Boureau, J. Ponce, and Y. LeCun. 2010. A Theoretical Analysis of Feature Pooling in Visual Recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*. 111–118. https://icml.cc/Conferences/2010/papers/638.pdf

[7] L. Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. 2000. LOF: Identifying Density-Based Local Outliers. *ACM SIGMOD Record* 29, 2 (2000), 93–104.

[9] R. Chalapathy and S. Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).

[10] V. Chandola, A. Banerjee, and V. Kumar. 2009. Anomaly detection: A survey. *Comput. Surveys* 41, 3 (2009). doi:10.1145/1541880.1541882

[11] C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine Learning* 20 (1995), 273–297.

[12] T. M. Cover and P. E. Hart. 1967. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory* 13, 1 (1967), 21–27. doi:10.1109/TIT.1967.1053964

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 248–255.

[14] M. M. Deza and E. Deza. 2009. *Encyclopedia of Distances*. Springer. doi:10.1007/978-3-642-00234-2

[15] L. Duan, D. Xu, and I. W. Tsang. 2012. Learning with Augmented Features for Heterogeneous Domain Adaptation. *CoRR* abs/1206.4660 (2012). arXiv:1206.4660 http://arxiv.org/abs/1206.4660

[16] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of KDD* (1996), 226–231.

[17] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li. 2019. Computer vision algorithms and hardware implementations: A survey. *Integration* 69 (2019), 309–320. doi:10.1016/j.vlsi.2019.07.005

[18] M. Harel and S. Mannor. 2011. Learning from multiple outlooks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*. 401–408.

[19] D. M. Hawkins. 1980. *Identification of Outliers*. Chapman and Hall.

[20] K. He, X. Zhang, S. Ren, and J. Sun. 2015. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 346–361.

[21] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. doi:10.1109/CVPR.2016.90

[22] M. Heidari, L. Ding, M. Kheshti, W. Bao, X. Zhao, M. Popov, and V. Terzija. 2024. A review on application of machine learning-based methods for power system inertia monitoring. *International Journal of Electrical Power & Energy Systems* 162 (2024), 110279. doi:10.1016/j.ijepes.2024.110279

[23] C. Huang. 2018. *Featured anomaly detection methods and applications*. Ph. D. Dissertation. University of Exeter.

[24] A. K. Jain, M. N. Murty, and P. J. Flynn. 1999. Data Clustering: A Review. *Comput. Surveys* 31, 3 (1999), 264–323.

[25] I. T. Jolliffe. 2002. *Principal Component Analysis* (2nd ed.). Springer, New York.

[26] A. Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto. (Introduces the CIFAR-10 and CIFAR-100 datasets).

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. doi:10.1109/5.726791 (Introduces the MNIST dataset).

[28] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen. 2018. Deep learning for remote sensing image classification: A review. *WIREs Data Mining and Knowledge Discovery* (2018).

[29] M. Lin, Q. Chen, and S. Yan. 2014. Network In Network. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[30] F. T. Liu, K. M. Ting, and Z.-H. Zhou. 2008. Isolation Forest. *Proceedings of the IEEE International Conference on Data Mining (ICDM)* (2008), 413–422.

[31] A. McCallum and K. Nigam. 1998. A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI Workshop on Learning for Text Categorization*.

[32] G. A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[33] J. Nam and S. Kim. 2015. Heterogeneous Defect Prediction. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (FSE 2015)*. 508–519. doi:10.1145/2786805.2786814

[34] H. F. Nweke, Y. W. Teh, M. A. Al-garadi, and U. R. Alo. 2018. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications* 105 (2018), 233–261. doi:10.1016/j.eswa.2018.03.056

[35] S. J. Pan and Q. Yang. 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2010), 1345–1359. doi:10.1109/TKDE.2009.191

[36] A. Patcha and J.-M. Park. 2007. An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends. *Computer Networks* 51, 12 (2007), 3448–3470.

[37] V. Phung and E. Rhee. 2019. A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences* 9 (10 2019), 4500. doi:10.3390/app9214500

[38] P. Prettenhofer and B. Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*. 1118–1127.

[39] K. Roth, L. Pemula, J. Zepeda, B. Schölkopf, T. Brox, and P. Gehler. 2022. PatchCore: Towards Total Recall in Industrial Anomaly Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[40] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, K.-R. Müller, and M. Kloft. 2018. Deep One-Class Classification. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[41] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli. 2018. Adversarially Learned One-Class Classifier for Novelty Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[42] D. Scherer, A. C. Müller, and S. Behnke. 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Artificial Neural Networks – ICANN 2010*. Springer, 92–101. doi:10.1007/978-3-642-15825-4_10

[43] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. 2001. Estimating the Support of a High-Dimensional Distribution. In *Advances in Neural Information Processing Systems*, Vol. 13.

[44] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. 2016. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging* (2016).

[45] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. 2003. A Novel Anomaly Detection Scheme Based on Principal Component Classifier. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 353–365.

[46] K. R. Singh and S. Dash. 2024. Exploring machine learning techniques for feature extraction and classification of diabetes related medical data: A comprehensive review. In *Internet of Things and Machine Learning for Type I and Type II Diabetes*. 153–175. Abstract only.

[47] X. Song, M. Wu, C. Jermaine, and S. Ranka. 2007. Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering* 19, 5 (2007), 631–645. doi:10.1109/TKDE.2007.1009

[48] Stanford Vision Lab. [n. d.]. Tiny ImageNet Challenge (CS231n course materials). https://cs231n.stanford.edu/reports/2017/pdfs/930.pdf. Accessed 2025.

[49] S. Survarachakan, P. J. R. Prasad, R. Naseem, J. Pérez de Frutos, R. P. Kumar, T. Langø, F. Alaya Cheikh, O. J. Elle, and F. Lindseth. 2022. Deep learning for image-based liver analysis — A comprehensive review focusing on malignant lesions. *Artificial Intelligence in Medicine* 130 (2022), 102331. doi:10.1016/j.artmed.2022.102331

[50] C. Wang and S. Mahadevan. 2011. Heterogeneous Domain Adaptation Using Manifold Alignment. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. 2. 541–546.

[51] H. Xiao, K. Rasul, and R. Vollgraf. 2017. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[52] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. 2014. How Transferable Are Features in Deep Neural Networks?. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[53] S. Zahia, M. B. Garcia Zapirain, X. Sevillano, A. González, P. J. Kim, and A. Elmaghraby. 2020. Pressure injury image analysis with machine learning techniques: A systematic review on previous and possible future methods. *Artificial Intelligence in Medicine* 102 (2020), 101742. doi:10.1016/j.artmed.2019.101742

[54] M. D. Zeiler and R. Fergus. 2013. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1301.3557

[55] X. X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer. 2017. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine* (2017).

Eidesstattliche Versicherung

(Affidavit)

Mahjabin, Sadia

Name, Vorname
(surname, first name)

229913

Matrikelnummer
(student ID number)

☐ Bachelorarbeit
(Bachelor's thesis)

■ Masterarbeit
(Master's thesis)

Titel
(Title)

Anomaly Detection with Pre-trained CNN Features: A Comparative Study of Classical Methods

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund, 22.01.2016

Ort, Datum
(place, date)

Sadia Mahjabin

Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund, 22.01.2026

Ort, Datum
(place, date)

Sadia Mahjabin

Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**

Ergänzung zur Eidesstattlichen Versicherung (Affidavit) hinsichtlich des Einsatzes von generativen IT-/KI-gestützten Sprachmodellen

Mahjabin, Sadia — 229913

Name, Vorname (surname, first name) — **Matrikelnummer** (student ID number)

Anomaly Detection with Pre-trained CNN Features: A Comparative Study of Classical Methods

Titel (title) ☐ **Bachelorarbeit** (bachelor thesis) ☒ **Masterarbeit** (master thesis)

In Ergänzung zur abgegebenen eidesstattlichen Versicherung (Affidavit) versichere ich, dass ich bei der Erstellung der vorliegenden Abschlussarbeit betreut von (Erstbetreuer):
Prof. Dr. Emmanuel Müller

und (Zweitbetreuer):
Dr. Simon Klüttermann

im Falle des Einsatzes von generativen IT-/KI-gestützten Sprachmodellen diese Werkzeuge in der Übersicht verwendeter Hilfsmittel mit ihrem Produktnamen und der Art des Hilfsmittels, der Art der Verwendung und ggf. den jeweiligen Prompts vollständig aufgeführt habe (siehe beispielhaft nachfolgende Tabelle). Erlaubte Einsatzzwecke sind unter anderem Optimierung selbstverfasster Texte hinsichtlich Grammatik, Rechtschreibung und Schreibstil, Brainstorming bei der Forschungsfrage oder Unterstützung bei der Literaturrecherche, die die Eigenständigkeit bei der Bearbeitung der Abschlussarbeit nicht einschränken. Ferner versichere ich, dass über die genannten Beispiele hinausgehende Verwendung von IT-/KI-gestützten Sprachmodellen mit dem/der Betreuer:in der Arbeit abgesprochen wurde. Auf die Belehrung hinsichtlich der Folgen von Täuschungsversuchen und der Abgabe einer falschen Versicherung an Eides statt wird auf die eidesstattliche Versicherung und die Prüfungsordnung verwiesen. Diese Belehrung habe ich zur Kenntnis genommen.

In addition to the affidavit submitted, I hereby affirm that I have used generative IT-/AI-supported language models in the preparation of this thesis under the supervision of (first supervisor):
Prof. Dr. Emmanuel Müller

and (second supervisor):
Dr. Simon Klüttermann

and that I have listed these tools in full in the overview of tools used with their product name and the type of tool, the type of use and, if applicable, the respective prompts (see table below as an example). Permitted uses include optimization of self-authored texts with regard to grammar, spelling and writing style, brainstorming on the research question or support in literature research that does not restrict independence in the processing of the thesis. Furthermore, I assure that the use of IT-/AI-supported language models beyond the examples mentioned has been discussed with the supervisor(s) in the thesis.
Please refer to the affidavit and the examination regulations for information on the consequences of attempted cheating and making a false declaration in lieu of an oath. I have taken note of these instructions.

Verwendete IT-/KI-gestützte Sprachmodelle IT/AI-supported language models used

Produktname Productname	**Art des Hilfsmittels** Type of tool	**Art der Verwendung** Type of use	**Prompt**	**Textstelle(n)** Text passage(s)
e.g. ChatGPT	AI Writing Assistant	Brainstorming Research Questions, Optimizing the structure of the chapters and sections, reducing redundancy, assistance in identifying related research literature, support in debugging and resolving implementation errors in code.	"Refine this research question.", "Improve the structure of this chapter.", "Help identify the bug in this code."	-

Dortmund, 22.02.2026 — Sadia Mahjabin

Ort, Datum (place, date) — **Unterschrift** (signature)

Table 9.1: Examples of AI-assisted support and exemplary prompts

Product name	**Type of application**	**Exemplary prompt**
e.g., ChatGPT	Brainstorming on the research question	What are possible research questions on the topic of "X"?
e.g., ChatGPT	Help with programming for troubleshooting	What is the error in the following code?
e.g., ChatGPT	Help with the literature search	What literature can you recommend on the subject of "X"? In which papers or books can I find information on "X"? In which papers is a similar question to "X" investigated?
e.g., ChatGPT	Help with the formulation of texts	Convert my following key points into a continuous text.
e.g., ChatGPT	Optimization of the self-written section XY regarding spelling, grammar and writing style	Optimize the text and correct the grammar and spelling mistake