Supervised and Unsupervised Learning Comparative Cost Analysis in Real-Time Manufacturing

Otema Yirenkyi

Final Thesis Project

Department of Information Technology

Uppsala University

20th June 2025

Abstract

Anomaly Detection is an integral subject area in machine learning used for various applications, making significant strides in Medical Imaging, Security Surveillance and Autonomous Driving. In this work, focus is on exploring Unsupervised Machine Learning algorithms in manufacturing industry for real-time quality inspection. Using metal coating quality assessment as a case study, the project develops methods that can automatically detect defects without requiring labeled training data, comparing various unsupervised methods and benchmarking their performances against an existing supervised approach. The goal is to establish a generalized approach that reduces implementation time, time to market and resources while maintaining reliable detection capabilities, addressing a key challenge in industrial quality control where labeled defect data is often unavailable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Constant inspection is an assumed practice within the manufacturing and production industry. As frequent and important as this procedure is, it requires careful attention to detail, involving a lot of time. The often manual quality control inspections, are prone to human error and inadvertently expensive. This necessitates the automation of this process to ensure industry standards of production always meet the highest criteria, reduce waste, and meet customer satisfaction. The introduction of automated processes presents another issue - access to labeled data. In quality assessment, the distinction between normal components and defect components need to be defined. There is often an imbalance between normal and anomalous data, with the latter accounting for a smaller proportion of the data. Furthermore, the infrequency and diversity of anomalous or defective data result in limited labeling, which in turn deters the training of supervised machine learning models for automation. To obtain admissible accuracy, supervised learning models require considerable amounts of labeled anomalous data. Anomaly detection is the process of identifying patterns or events that deviate significantly from the norm Chalapathy and Chawla [2019].

The motivation for this project is to explore methods that address the inaccessibility to labeled data through unsupervised anomaly detection machine learning methods. This method, in its implementation, will require only the use of normal, detect-free data samples in its training, to effectively identify anomalous instances in test datasets. There is a need to provide a method that is efficient and resource-conscious to support faster time-to-market and enable widespread adoption. The performance of this method will be compared against supervised learning alternatives, assessing model performances and cost-effectiveness in real-time quality inspection.

1.2 Brief literature review

In prior years, various methods have been designed and refined to fit applications within different sectors including the manufacturing industry. Traditional existing methods have relied on hand-crafted features and texture analysis. However, these fail to meet acceptable performance levels due to their rigidness with non-uniform surfaces Xie [2008]. These traditional machine learning methods including Support Vector Machine (SVM), and Random Forest Algorithms, require feature selection which could sometimes be a limitation Xie et al. [2024b].

Deep learning methods have increasingly become more popular in anomaly

or outlier detection. Convolutional Neural Networks (CNNs), although they perform well in image-based anomaly detection require extensive labeled data Xie et al. [2024b], a limitation in real-life situations.

Incremental one-class neural networks and Dynamic Shallow Neural Networks have also been developed for cognitive detection and diagnosis for unknown faults using limited data. A limitation is the requirement of prior knowledge of the characteristics of the different faults, with the aid of human input Arunthavanathan et al. [2020].

Unsupervised learning approaches such as Convolutional Autoencoders (CAEs) work by learning the distributions within the normal dataset during training. However, this method is likely to be restricted by blurry reconstructions, and a lack of feature interpretability Qian et al. [2022].

Generative Adversarial Networks (GANs) have also been used for generating synthetic defect data, but can be computationally expensive Akcay et al. [2019].

Other methods used beyond the manufacturing sector include hierarchical clustering methods for outlier detection Loureiro et al. [2004]. However, these assume the distribution of anomalies and thus may perform poorly on complex datasets.

Additionally, many existing methods fail to address contextual or logical anomalies, with current anomaly detection benchmarks lacking representation of complex datasets that suit diverse industrial applications Batzner et al. [2024].

1.3 Research questions

This project answers the central research question "What is the computational cost and time efficiency of unsupervised vs. supervised learning approaches for real-time quality inspection in manufacturing?" The project will explore the best unsupervised machine learning models and compare its performance against supervised models in effectively detecting anomalies within manufacturing contexts for unrivaled time-to-market and cost efficiency in real-time scenarios.

1.4 Thesis Outline

The thesis project is structured as follows:

Background: This involves conducting a literature review of existing research on anomaly detection methods in manufacturing, comparing supervised and unsupervised approaches, and discussing relevant image analysis and machine learning techniques.

Methodology: The methods used within this project are described in detail and compared with previous work.

Experimental setup: Each experiment is aligned with the research question "What is the computational cost and time efficiency of unsupervised vs. supervised learning approaches for real-time quality inspection in manufacturing?", and the design of the setup is intended to either support or challenge this. The experiments will include ablation studies to evaluate the contribution of individual components, as well as parameter-sensitivity analyses to observe how different settings can impact overall performance.

Data Collection and Pre-processing: Collecting relevant data for metal coating inspection and preprocessing it through data cleaning for feature extraction and final classification.

Model Implementation: Implementing appropriate unsupervised learning algorithms and comparing with a supervised model trained on the same gathered data.

Evaluation: The primary evaluation metrics will include Accuracy, Precision, Recall, and F1-score for assessing model performance in detecting anomalies. Training, Inference Time and Memory Usage will evaluate the efficiency of each method. Classical statistical methods for anomaly detection, such as Z-Score Analysis, will be reviewed to see how the anomalies deviate from the norm of the data collected.

Comparative Analysis: The supervised baseline will include models like decision trees or neural networks, while unsupervised models will include Autoencoders (AEs) and Knowledge Distillation Networks. Each model will be evaluated based on the performance metrics.

Results: Experimental findings and a comparison of different model results based on chosen metrics is presented.

Discussion: Analysis of the strengths and weaknesses of each approach, a discussion of real-time manufacturing trade offs, a cost-benefit analysis and an assessment of time to market impact is discussed.

Conclusion: Summary of results, conclusions drawn from the study, recommendations for future research, and considerations are made for industrial implementation.

Chapter 2

Background

2.1 Literature Review

Anomaly detection through computer vision is increasingly becoming a highly researched topic of interest particularly in production. This has been fueled by the fourth industrial revolution, to drive efficiency within production lines. Another reason is to reduce the cost of labor and time consumption associated with manual inspection Cui et al. [2023]. Traditionally, quality inspection methods for computer vision surface defect detection extract discriminant features. Other popular techniques that have been applied on material surfaces include texture analysis Xie [2008]. These methods perform incredibly well on uniformly textured surfaces but are limited in their application which requires specificity in design for different material surfaces, tending to fail for non-periodic patterns Tsai and Jen [2021].

One-class methods have also been proposed as fault detection strategies. A one-class support vector machine was proposed by Mahadevan and Shah [2009] for detecting faults within a processing plant. While Arunthavanathan et al. [2020] uses an incremental one-class neural network for fault detection and a shallow neural network for classification.

Isolation forest (iForest), initially introduced by Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou in 2008, identifies outliers via a computationally cost effective approach, performing better than one-class SVM Liu et al. [2012].

Deep learning algorithms are steadily being sought after due to the limitations that come with classical image analysis and machine learning approaches Qian et al. [2022]. Compared with traditional computer vision methods for surface inspection, deep learning algorithms have performed much better in recent times. Data-driven process monitoring, where focus has been on statistical learning, dynamic latent variable analytics, and Autoencoder based fault diagnosis, has been proposed in Yin et al. [2012] Yang et al. [2022].

The under-supply of labeled data remains one of the major obstacles in industrial anomaly detection. Within the manufacturing industry, the occurrence of anomalies can be quite rare due to strict quality and reliability requirements expected. This makes it incredibly challenging to gather a significant amount of data samples with defects, specifically for supervised machine learning classification tasks. Supervised methods require large amounts of labeled data which is often annotated by humans. This creates a limitation due to human errors caused by oversight and tiredness due to the tedious nature of the task. This limitation has increased the need for unsupervised learning approaches for visual anomaly detection within an industrial context.

Unsupervised learning methods typically utilize only normal data for training, which makes it feasible in such industrial situations where acquiring large quantities of normal data is expected Cui et al. [2023].

Autoencoders and their variants remain one of the much studied deep learning models for anomaly detection under unsupervised learning methods. With its feature extraction characteristic, AE-based models reconstruct normal images during training of the defect-free data samples, by learning the pattern structure. Expectedly, the Autoencoder model will deviate from the normal reconstruction when anomalous data is fed into the model Xie et al. [2024a]. Compared to normal data, defect data will result in a higher reconstruction error, also known as the anomaly score. When the score exceeds a set threshold, the input image is tagged an anomaly.

Multiple variants of Autoencoders have been designed to address the challenges of anomaly detection for unsupervised learning within the manufacturing industry.

The basic Autoencoder neural network serves as the foundation for unsupervised feature extraction Bengio et al. [2013]. Linear AutoEncoders, with one hidden layer with linear activation units and no biases, work similarly to Principal Component Analysis algorithm under Dimensionality Reduction, and this similarity in model parameters has been explored by researchers Qian et al. [2022].

Denoising Autoencoders (DAE) have been observed to improve robustness during its learning to reconstruct data irrespective of noisy inputs Vincent et al. [2010]. Convolutional Autoencoders (CAEs) use convolutional neural networks to learn patterns within image data. The limitation here is the overlap of both normal and anomalous features within the latent space; however, regularization techniques can distinguish normal from anomalous data samples. This is done by adding a penalty to the loss function to cause features from the normal data to cluster, thus improving accuracy on material surface detection Tsai and Jen [2021]. Although robust to noise, the tendency to lose vital information may occur while CAEs focus on minimizing sensitivity to input changes Qian et al. [2022].

Variational Autoencoders (VAEs) are generative probabilistic models that learn distributions of latent variables, modeling the mean and variance of these variables Kingma et al. [2013]. For anomaly detection, after learning the data distributions, they evaluate the reconstruction probability or derive fault indices from the distribution or the loss function, making them more complex than simpler Convolutional Autoencoders Cui et al. [2023].

Outside of Autoencoders, other prospective unsupervised learning approaches exist for industrial anomaly detection:

Generative Adversarial Networks (GANs) are neural network models that use adversarial training between a generator and a discriminator. It learns a generative distribution in order to approximate the real normal data distribution. It generates synthetic data for semi-supervised training to mitigate the issue of scarce training data Akcay et al. [2019]. However, training GAN models can be complex due to convergence failure Metz et al. [2017].

Normalizing Flow (NF) based methods are likelihood-based models that learn the distribution of normal data through the transformation of simple data distributions from complex ones Rezende and Mohamed [2015].

Representation-based methods use pre-trained models such as Visual Geometry Group (VGG) or feature extractors in learning image features, applying transfer learning techniques Simonyan and Zisserman [2015].

Since training in unsupervised methods learn patterns from normal data, augmentation-based methods have been designed to artificially create anomalies for training and

detection Zavrtanik et al. [2021].

There are methods such as attention mechanisms that specifically enhance algorithms used for anomaly detection Venkataramanan et al. [2020], inpainting for anomaly detection Pirnay and Chai [2022] and student-teacher training models which train on normal image data Zolfaghari and Sajedi [2022]. One such technique is Knowledge Distillation (KD) technique used to train a smaller, more efficient model (the "student") by leveraging the knowledge learned by a larger, higher-performing model (the "teacher") Gou et al. [2021]. This technique is useful in resource constrained device, allowing the deployment of relatively smaller sized models which are less computationally expensive. With Knowledge Distillation, a distillation loss, which measures the difference between student (intermediate features) and teacher's output, is combined with a standard loss from the student model training. The teacher model transfers vital learned weights and feature representations to the student, enabling the student to achieve higher performance and greater computational efficiency than if trained from scratch. Knowledge Distillation research is actively being pursued within the image processing and classification domain Stanton et al. [2021].

2.2 Background Information

While traditional manual inspection exists for anomaly detection, it is time-consuming and resource contraining. Thus, there have been constant new techniques adopted to automate this otherwise tedious but necessary process, for consistency in product quality, ensuring safety and reliability, while maintaining steady operations and minimizing loss. Anomaly detection aims to identify patterns within the production line that deviate significantly from what is expected.

This has led to the invention of real-time quality inspection, which poses its own challenges. Due to the strict standards within the industrial context, the occurrence of anomalies and its varieties, are relatively uncommon. Thus, collecting diverse data that consists of both normal and known anomalies is tasking and labor-intensive Cui et al. [2023].

Supervised machine learning approaches have been adopted over time, however the collection of data requires extensive labeling for such algorithms, which can also be time-consuming.

This drawback motivates using unsupervised machine learning approaches to identify anomalies from collected data without requiring labeling Chandola et al. [2009], showing great capability in overcoming the associated limitations of supervised methods for visual defect detection Cui et al. [2023].

This thesis project involves exploring machine learning algorithms for anomaly detection in production lines, specifically focusing on real-time quality inspection. The case study for this project is metal coating quality assessment, however the absence of the required data led to the use of publicly accessible concrete data that would serve the same purpose. The main objective is to develop methods capable of automatically detecting defects without the need for labeled training data. This involves implementing unsupervised learning algorithms and comparing their computational cost and time efficiency against a supervised approach trained on the same data.

Unsupervised models like Autoencoders, and Patch Description Networks through Knowledge Distillation are designed for this task, while models like Convolutional Neural Network and Residual Neural Network are implemented under a supervised learning approach. The original dataset used for this experimentation is also subjec-

ted to augmentation. Two separate experiments are conducted: the first experiment group using only the original data, and the second experiment group using a combination of the original and the augmented data. This ablation study is done to investigate the effectiveness of data augmentation techniques in improving model performance under both supervised and unsupervised approaches. The ultimate aim of this study is to establish a generalizable approach that reduces the time and resources needed for implementation and deployment while maintaining reliable detection capabilities, directly addressing the challenge of limited labeled defect data in industrial quality control.

Chapter 3

Methodology

Through experimentation, the steps in this project are taken to make a sound comparison, on the basis of performance, computational complexity, and time, between unsupervised machine learning and supervised machine learning methods. These steps aim to address the central research question "What is the computational cost and time efficiency of unsupervised vs. supervised learning approaches for real-time quality inspection in manufacturing?".

Through exploring both unsupervised and supervised methods, the research question is addressed, comparing the performance of various deep machine learning models based on model training and inference time. The time monitoring is instrumental in determining how efficient the automation process is, contributing to how much time it takes for a chosen model to apply real-time quality inspection on the production line. The size of the model too is explored. A small size model is proposed to address the cost-effective part of the research question, ensuring that the chosen model does not consume unnecessary computing power but is still able to produce suitable performance based on chosen metrics. Evaluating both unsupervised and supervised methods address the research question such that we compare which of these two approaches produce the better results for the task of anomaly detection within the manufacturing context.

The data acquired is of two classes; normal and anomalous. It is on this data that the machine learning methods are applied and compared.

3.1 Classical machine learning methods

This approach involved a comprehensive review of the applications of both classical machine learning and deep learning methods in anomaly detection across supervised and unsupervised learning contexts. The foundational principles underlying classical machine learning techniques were examined, with particular attention given to the use of statistical methods for identifying data points that deviate from expected statistical distributions or patterns. Further investigation revealed that deep learning models have significantly outperformed classical methods in terms of accuracy, speed, and cost. Consequently, an in-depth analysis of deep learning approaches was conducted.

3.2 Deep learning methods

3.2.1 Model Architecture

The creation of a convolutional neural network consists of three main components Wilmet et al. [2021].

- Convolutional Layers: Convolutional layers work on the principle of kernels or filters. A convolving operation is applied on each image data input into the neural network. The layer convolves the kernel across the spatial dimensions of the image to create a two-dimensional (2D) activation map. The convolving operation involves a scalar product calculation for each image pixel. This allows the network to learn per-pixel characteristics of the image. The rate at which the convolution takes place is dependent on the stride parameter which tell the number of dot products to compute.
- Pooling layer: The pooling layer further reduces the dimensionality of the image representation. The output of the convolutional layers is fed as input to the pooling layers, manipulating each activation map, and scaling the dimensionality.
- Fully-Connected Layer: This layer's neurons are connected directly to the output neurons to give the final predictions from the neural network.

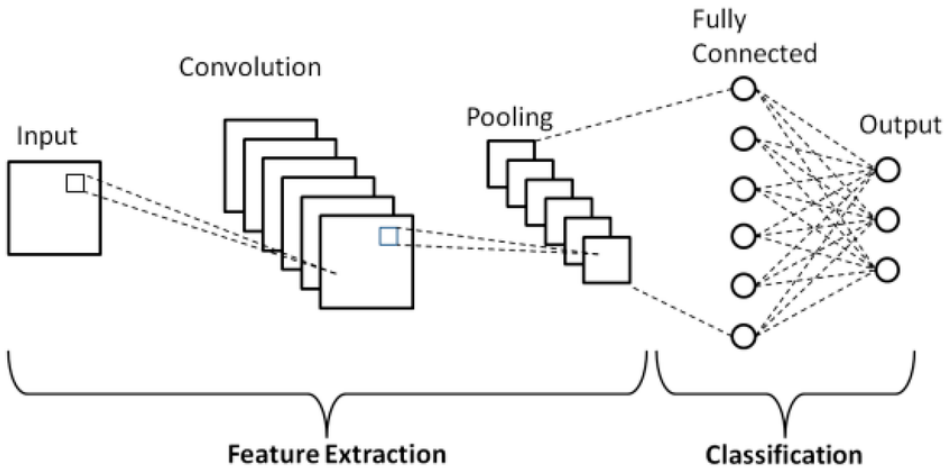


Figure (3.1) A simple Convolutional Neural Network Phung and Rhee [2019]

3.2.2 Unsupervised Learning

The use of the unsupervised technique addresses the limitation of labeled data not always available in real-world settings.

Three distinct approaches for image anomaly detection are implemented. within this project. The first method executes a standard convolutional autoencoder using reconstruction error, to flag image samples with high error as anomalies. It works on the principle that the autoencoder will be able to reconstruct the normal images its been trained on. Assuming that the reconstruction error of normal images will be low while anomalous images which were not used in training will result in a higher reconstruction error.

The Convolutional Autoencoder is a Convolutional Neural Network arranged in an auto-encoding configuration which provides an approximation of input data fed into the network. It leverages convolutional operations for both encoding and decoding. The input data is passed through neurons within the hidden layers up to the bottleneck layer. The input is compressed into a representation map. The decoder then decompresses and maps this representation to the output layers, indicating any representation deviating from the ground-truth, an anomaly.

The second method is a build up on the initial method; an EfficientNet-based Autoencoder which adapts a pre-trained EfficientNet model as its Encoder, with a lightweight Decoder. Being an autoencoder model, it again utilizes reconstruction error for the detection of the anomalies.

The third method is a Patch Description Network (PDN) trained through knowledge distillation from a large pretrained teacher network. In knowledge distillation, the student (smaller model) network learns through training, to reproduce feature maps which were generated by the teacher network on normal data through its learning. The anomalies are detected through the comparison of these learned teacher and student feature representations. The PDN model used is another a lightweight, fully convolutional 4-layer network which servers as the student. While the teacher network used is a pre-trained Residual Neural Network (RNN). The RNN architecture also consists of the input, hidden and output layers. The distinguishable factor here is having shortcut connections. Information from a previous layer is passed directly to the next layer without transformation [He et al., 2016].

3.2.3 Supervised Learning

For this supervised approach, training data comprises of normal and anomalous samples. The supervised deep learning model serves as the performance baseline to the unsupervised learning approach.

For the first approach, a 5-layer convolutional neural network is designed. The model is designed with five sequential blocks of convolution, ReLU activation and Max Pooling. The first layer takes the input image channels and outputs multiple channels after convolving. After each convolution, the ReLU activation is applied to introduce non-linearity. The Max Pooling layer reduces the spatial dimensions of the representations. The output features are then flattened and passed to a fully-connected layer with an activation function applied. That output is fed to a second fully-connected layer with a sigmoid activation function applied to its output for final binary classification probability prediction.

For the second approach, a pre-trained ResNet18 model is implemented to train on the data. Instead of building a convolutional neural netowrk from scratch, this approach applies transfer learning from the ResNet18 model already trained on the large ImageNet dataset Zagoruyko and Komodakis [2016]. The final fully-connected layer of the ResNet18 model is modified to a new linear layer that outputs two values that fit the description of this project; normal and anomalous.

3.3 Augmentation and Pre-processing

To improve model performance, offline augmentation techniques are applied to increase training data and introduce variability for both unsupervised and supervised learning methods. This means that augmentation is applied separately to the

images in an earlier step, and not during model training. This step is aimed at aiding the chosen model to learn diverse features and enable it perform better on unseen anomaly data samples. Augmentation also helps to prevent overfitting when the number of training samples are increased. Overfitting is when the trained model performs well on the training set but does not generalize properly to the test set Liu et al. [2023].

During the offline augmentation, techniques are applied only to the training dataset. The resulting augmented dataset combines the original real data and the generated data. In training, the validation set is not augmented; this ensures that the benchmark for evaluating model performance is not altered to affect the model's generalization ability to real data.

Under the unsupervised approach, the training dataset contains only normal images, while the validation and test images contain a mix of normal and anomalous images. The following augmentation methods were applied on the image:

The first augmentation step involves applying symmetric padding to each image. This is achieved by reflecting the image pixel values at the boundaries, instead of padding with zeros or a constant value, effectively extending the image size without introducing artificial artifacts. The padding is performed using NumPy's pad function with the reflect mode, and the result is converted back to a PIL image. This step helps preserve edge information and prevents information loss during subsequent cropping or affine transformations, which might otherwise introduce artificial artifacts or crop out important features.

Listing (3.1) Padding function

```
def symmetric_pad_pil(img, pad):
    arr = np.array(img)
    if arr.ndim == 2:  # Grayscale
        arr = np.expand_dims(arr, axis=-1)
    arr = np.pad(
        arr,
        ((pad, pad), (pad, pad), (0, 0)),
        mode='reflect')
    arr = arr.squeeze() if arr.shape[2] == 1 else arr
    return Image.fromarray(arr.astype(np.uint8))
```

Next, an adaptive brightness jitter is applied to introduce photometric variation. This custom augmentation adjusts the brightness of each image based on its average pixel intensity. The adjustment factor is computed so that darker images are brightened more, and lighter images are brightened less, within a specified range (from 0.8x to 1.2x the original brightness). This adaptive approach ensures that the augmentation is context-sensitive, preventing overexposure or under-exposure and maintaining natural-looking results.

Listing (3.2) Adaptive Brightness

```
class AdaptiveBrightnessJitter:
    def __init__(self, min_rel=0.8, max_rel=1.2):
        self.min_rel = min_rel
        self.max_rel = max_rel

    def __call__(self, img):
        stat = ImageStat.Stat(img)
        mean_per_channel = stat.mean
        avg_brightness = sum(mean_per_channel) / 3.0
        norm = avg_brightness / 255.0
        weight = 1.0 - norm
        factor = self.min_rel * (1 - weight) + self.max_rel * weight
        return T.functional.adjust_brightness(img, factor)
```

The core augmentation pipeline utilizes Kornia, a differentiable computer vision library for PyTorch. The pipeline includes several transformations applied sequentially:

The image is randomly cropped and resized to a fixed output size (256x256 pixels). The crop scale and aspect ratio are varied within specified ranges, introducing variability in the framing of the subject. The image undergoes random affine transformations, including rotation (up to ±20 degrees), translation (up to 15% of the image size in both x and y directions), and scaling (from 0.8x to 1.2x). The padding mode is set to reflection. Each image has a 50% chance of being flipped horizontally and a 50% chance of being flipped vertically, further increasing dataset diversity. Finally, the image is center-cropped to the target output size, ensuring uniformity in dimensions across the dataset.

Listing (3.3) Kornia Function

```
kornia_augment = torch.nn.Sequential(
    K.RandomResizedCrop(
        size=(output_size, output_size),
        scale=(0.5, 1.0),
        ratio=(0.9, 1.1),
        p=1.0
    ),
    K.RandomAffine(
        degrees=20,
        translate=(0.15, 0.15),
        scale=(0.8, 1.2),
        padding_mode='reflection',  # This is supported in Kornia 0.8.0
        p=1.0
    ),
    K.RandomHorizontalFlip(p=0.5),
    K.RandomVerticalFlip(p=0.5),
    K.CenterCrop((output_size, output_size))
)
```

For each original image, the pipeline generates a specified number of augmented variants (three per image). The dataset contains both the original and multiple augmented versions for training.

Under the supervised approach, augmentation is applied to increase the count and variability of the defective images. The augmentation efforts here take into account the class imbalance between normal and anomalous images in the training dataset. The following augmentation methods were applied to ensure both the preservation of original data and the generation of realistic synthetic variants.

The augmentation process begins with the organization of the dataset into two primary classes: "normals" and "anomalies." The code first determines the number of images in each class and calculates a target number of images per class, intentionally overshooting the majority class by a configurable ratio (in this case, 80%). This ensures that both classes are balanced and that the dataset is sufficiently large for robust model training.

The same augmentation operations in Listing 3.1, Listing 3.2 and Listing 3.3 are also applied to each image under the supervised learning approach.

After all augmentations, the images are resized back to their original dimensions using bilinear interpolation. This ensures consistency in input size for downstream tasks.

Testing the performance of the trained models is done on a separate unaugmented dataset, serving as a benchmark. The unseen dataset is the same set used across the different experimental groups; evaluating the model performance on augmented train data, and unaugmented train data.

3.3.1 Method comparison with previous work

Comparing the implemented methods to other methods:

AutoEncoder

The use of a Convolutional AutoEncoder (CAE) and an EfficientNet-based Autoencoder is among some of the go to methods for reconstruction-based anomaly detection methods. The core idea here as described in Cui et al. [2023] and Xie et al. [2024a] is to train a neural network to reconstruct normal data, using the reconstruction error as an indicator of anomaly. A basic autoencoder comprises an encoder and a decoder, optimized through back-propagation based on the reconstruction error. Qian et al. [2022] discusses similar autoencoder variants and their applications. Yang et al. [2022] reviews AE-based representation learning and its applications in fault diagnosis (FDD). Yan et al. [2016] discusses fault indexes like SPE (reconstruction error) and T2 based on AE and its variants, noting SPE's sensitivity to fault information with DAE (Denoising Autoencoder). Fault classification tasks utilizing sparse AE, DAE, and SAE (Stacked Autoencoder) are also described, often involving feature extraction followed by fine-tuning Chopra and Yadav [2015] Thirukovalluru et al. [2016]. AE-based fault identification methods using contribution plots, such as those based on sparse AE, SAE, and LSTM-AE (Long Short-Term Memory Autoencoder), are used to show the contribution of variables to the fault detection index Hallgrímsson et al. [2020] Jiang et al. [2019] Toikka et al. [2021] Mao et al. [2015].

The Hierarchical Augmented Autoencoder (HAAE) described in Xie et al. [2024b] and is another advanced reconstruction-based model. It includes a symmetric skip-connected autoencoder, hierarchical memory modules, and a compound metric loss function. The memory modules help record multi-scale prototype patterns of normal samples to achieve robust reconstruction for normal data while yielding poor reconstruction for anomalies. Skip-connections transfer global and local information for better reconstruction. This highlights how basic AE structures can be enhanced with additional components to improve performance, particularly in noisy conditions and for anomaly detection. The proposed EfficientNet-based AE, in this paper, with a lightweight decoder can be seen as exploring variations on the standard AE structure, similar to how other works explore DAE, SAE, LSTM-AE, or add components like memory modules or skip connections.

Cui et al. [2023] notes a challenge with classical reconstruction methods like Autoencoders and GANs: where a coarse reconstructions can produce excessive image differences, preventing accurate anomaly detection. This suggests that basic AE models might have limitations. Thus, enhanced versions such as the EfficientNet variant proposed in this project aims to overcome this limitation.

Knowledge Distillation (KD)

The implementation of the proposed Patch Description Network (PDN) trained through knowledge distillation aligns closely with the method introduced as EfficientAD in Batzner et al. [2024] and Bergmann et al. [2020]. This approach uses KD to train a smaller student model (PDN) to mimic the feature output of a larger, pre-trained teacher network. Anomalies are detected by comparing the representations from the teacher and student networks.

Batzner et al. [2024] presents EfficientAD-S and EfficientAD-M, which utilize this PDN via KD method. They demonstrate superior performance in anomaly detection and localization, as well as significantly higher computational efficiency (lower latency, higher throughput) compared to other methods like AST (Asymmetric Student Teacher), S-T (Student-Teacher), PatchCore, VAE, and CAE. This strongly sup-

ports this thesis's focus on evaluating computational cost and time efficiency, as KD is presented as a technique explicitly designed for this purpose Pettersson [2023].

This shows that KD is a general technique applicable to different base model architectures like the CNN used for the Patch Description Network in this project, to improve effectiveness, encompassing both performance and computational efficiency. The goal is to transfer knowledge from a powerful teacher to a more efficient student.

Beyond reconstruction-based and KD methods, other literature discuss other established categories of unsupervised anomaly detection that are different from methods proposed within this thesis work. Clustering-based methods identify outliers as instances that do not belong to any cluster or form small clusters.

Loureiro et al. [2004] describes using hierarchical clustering for outlier detection in foreign trade transaction data, finding that metrics like Canberra distance are effective. These methods are often applied to structured or time-series data, which deviates from the focus on images for thesis paper.

Representation-based methods use deep neural networks to extract features, and anomaly scores are calculated based on the distance between test features and reference normal features Cui et al. [2023]. Examples include Deep SVDD, PatchCore, SPADE, and PaDIM. Deep SVDD trains a network to map normal data features within a minimal hypersphere. PatchCore uses a memory bank of normal patch features Batzner et al. [2024]. Xie et al. [2024b] and Tsai and Jen [2021] also list Deep SVDD as a comparison method, similar to how this paper includes supervised methods for comparison.

Normalizing Flow (NF)-based methods learn the distribution of normal data, and anomalies are detected as samples with low probability under this learned distribution.

Data Augmentation-based methods such as CutPaste is a method that creates synthetic anomalies to train models. This method compares the normal image samples with the generated samples by measuring the distance between them. This differs from this paper's approach where transformed images are classified as anomalous or not.

Supervised Methods

Cui et al. [2023] provides a direct comparison between supervised and unsupervised algorithms for anomaly detection. While supervised approaches can achieve high accuracy, they require a large amount of labeled data, which is often difficult and costly to acquire in industrial settings. This limitation makes unsupervised methods, which do not require labeled anomaly data, particularly appealing for practical applications. Comparing unsupervised methods against a supervised baseline in this thesis work, directly addresses this key trade-off.

Evaluation Context and Metrics

This paper evaluates performance using metrics like accuracy, precision, recall, F1-score, and confusion matrices. Other metrics such as training time, inference time, and model parameters & size to assess computational cost and efficiency are also measured. Other papers implements approaches using a variety of metrics, including AU-ROC, AU-PRC, AU-PRO, AU-sPRO (for localization), False Alarm Rate (FAR) Xie et al. [2024b]. Efficiency is assessed through training time, test/inference time, latency, throughput, number of parameters, and FLOPs Batzner et al. [2024]. The emphasis on computational cost and time efficiency in this work is mirrored in these approaches, which provide detailed efficiency comparisons, highlighting the practical importance of these factors for real-time industrial applications.

In summary, the proposed thesis methods, particularly the AE variants and the

PDN via Knowledge Distillation, represent state-of-the-art approaches within the deep learning paradigm for unsupervised anomaly detection, as corroborated by previous work. The comparison between unsupervised and supervised methods directly addresses a fundamental trade-off (data requirements vs. potential performance) prevalent in industrial anomaly detection. While other work present a broader landscape of methods (clustering, representation-based, normalizing flow) and applications (various materials, time series, transaction data), the chosen methods within this paper are central to current research in deep learning for visual industrial anomaly detection.

Chapter 4

Experimental Design Setup

The research question remains "What is the computational cost and time efficiency of unsupervised vs. supervised learning approaches for real-time quality inspection in manufacturing?" To address this question, multiple experiments with varying parameters and hyperparameters are tweaked for model performance improvement. This section outlines the experimental setup used to train, validate and evaluate multiple models for anomaly detection.

The following experiments are conducted on concrete data for crack detection due to metal data unavailability.

4.1 Libraries

The models are implemented in the Python language using the PyTorch library. The libraries used are as follows:

- torch: This core Pytorch library is designed to build and train neural networks. It provides tensor operation and GPU acceleration. It contains pre-built layers such as Fully Connected (Dense) Layers and Convolutional Layers. It also contains optimization algorithms such as Adam which is used within this project. These are used to update model weights during training. Torch library also includes utilities such as DataLoader and Dataset which handles the loading of datasets to be used in the training, aiding with batching and data shuffling.
- torchvision: This library provides image transformation techniques such as resizing for pre-processing and data augmentation.
- os: This is a system library that interactions with the operation system for file and directory creation and manipulation.
- random: It is a python built-in library used in generating random numbers, that is instrumental in operations such as data shuffling.
- time/datetime: This library measures training and inference times, and timestamps for logging.
- numpy: This library is used for array operations and mathematical computations.
- pandas: This is a data manipulation library for data analysis.
- PIL: The pillow library handles operations such as image conversion in Python.
- sklearn: This library contains functions for evaluating the models, implementing metrics such as the confusion matrix, recall, f1 score, and accuracy.

- matplotlib: This library is used to visualize data.
- seaborn: This is another statistical tool built on top of Matplotlib for data visualization.
- csv: This built-in module is used for reading and writing CSV files.

4.2 Dataset Structure

Here on, the dataset is denoted by

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}, \quad N \text{ being the total number of images.} \tag{4.1}$$

Each sample satisfies

$$x_i \in \mathbb{R}^{H \times W \times C}, \quad y_i \in \{0, 1\}, \tag{4.2}$$

where x_i represents the i-th image data (flattened pixel values), also known as the input features. $y_i = 0$ denotes a normal image while $y_i = 1$ denotes an anomalous image. H, W are the image height and width, respectively.

In this experimental setup:

- An online preprocessing pipeline is applied via a custom PyTorch `Dataset`, which reads the images and applies standardized transforms.
- For unsupervised training, all images are resized to 128×128 pixels (with the exception of the PDN which is distilled from the WideResNet101 model and requires 224x224).

  Hence,

  $$H = W = 128, \quad C = 1, \quad x_i \in \mathbb{R}^{128 \times 128}, \tag{4.3}$$

  because of the grayscale ($C = 1$) inputs.
- For supervised training, images are resized to 224×224. Therefore,

  $$H = W = 224, \quad C = 1, \quad x_i \in \mathbb{R}^{224 \times 224}. \tag{4.4}$$

The data is split into three sets.

1. **Training Set:** Denoted by

   $$\mathcal{D}_{\text{train}} = \{(x_i, y_i) \in \mathcal{D} \mid i \text{ used for training}\}. \tag{4.5}$$

   - For *unsupervised* learning, only normal samples are included:

     $$\mathcal{D}_{\text{train}}^{(\text{unsup})} = \{(x_i, y_i) \in \mathcal{D}_{\text{train}} \mid y_i = 0\}. \tag{4.6}$$

   - For *supervised* learning, all labeled samples (both normal and anomalous) are used:

     $$\mathcal{D}_{\text{train}}^{(\text{sup})} = \{(x_i, y_i) \in \mathcal{D}_{\text{train}}\}. \tag{4.7}$$

2. **Validation Set:** Denoted by

   $$\mathcal{D}_{\text{val}} = \{(x_i, y_i) \in \mathcal{D} \mid i \text{ used for validation}\}. \tag{4.8}$$

   This set contains a mixture of $y_i = 0$ and $y_i = 1$ samples, used to monitor all model performances during training and detect overfitting.

3. **Test Set:** Denoted by

   $$\mathcal{D}_{\text{test}} = \{(x_i, y_i) \in \mathcal{D} \mid i \text{ used for testing}\}. \tag{4.9}$$

   The test set is unaugmented, contains both normal and anomalous samples ($y_i = 0$ or 1), and is held out for final evaluation.

4.3 Features

As indicated in Equation (4.2), the outcome (classification label) is denoted by

$$y \in \{0, 1\}, \tag{4.10}$$

The implemented models will extract or learn features from the raw input image. From the high-level, feature extraction can be viewed as a mapping

$$\phi : \mathbb{R}^{H \times W} \longrightarrow \mathbb{R}^{D}, \quad x \mapsto \phi(x), \tag{4.11}$$

where $\mathbb{R}^D$ is a lower-dimensional feature space of dimension D.

4.4 Unsupervised Learning

4.4.1 Convolutional AutoEncoder Model

The first model implemented under the unsupervised learning approach is the Convolutional AutoEncoder. This model consists of an encoder and decoder. The encoder consists of three convolutional layers with ReLU activation after the first two layers to introduce non-linearity. The final encoder layer does not have any activation because it outputs raw features for the bottleneck. This layer outputs a compact representation.

- **First layer:** This layer converts the grayscale input image x from the single-channel channel to 16 channels, reducing the image spatial resolution .

  Let ReLU be the activation function.

  The output of the first layer is $\text{ReLU}(\text{Conv}_1(x))$.

- **Second layer:** This layer increases depth to 32 channels and reduces the spatial resolution to 32x32.

  Let Conv_2 denote the operation of the second convolutional layer.

  The output of the second layer is $\text{ReLU}(\text{Conv}_2(\text{ReLU}(\text{Conv}_1(x))))$.

- **Third layer:** This layer produces a 64-channel feature map. This final encoder layer does not have any activation function, outputting raw features for the bottleneck where $\phi(x) \in \mathbb{R}^D$.

  Let Conv_3 denote the operation of the third convolutional layer.

  The output of the third layer, representing the compact representation or bottleneck features, is $\phi(x) = \text{Conv}_3(\text{ReLU}(\text{Conv}_2(\text{ReLU}(\text{Conv}_1(x)))))$.

Although a smaller bottleneck size may prevent overfitting, this chosen size helps to improve reconstruction quality.

The decoder mirrors the encoder. Thus, it is also composed of three transposed deconvolutional layers that upsample the encoded representation back to the primary image dimensions. ReLU activation is again applied after the first two layers but Sigmoid activation is applied after the final layer in the decoder.

$\hat{x}$ is the reconstructed output image, which matches the input image x in dimensions (128x128, 1 channel).

The decoder's operations on the bottleneck representation $\phi(x)$ is represented sequentially using transposed convolutions (Deconv) and activation functions:

- **First layer:** This layer converts from 64 to 32 channels and up-samples the input image's dimensions.

  Let Deconv_1 denote the operation of the first transposed convolutional layer.

  The output is $\text{ReLU}(\text{Deconv}_1(\phi(x)))$.

- **Second layer:** Further expansion is made from 32 channels to 16 channels.

  Let Deconv_2 denote the operation of the second transposed convolutional layer.

  The output is $\text{ReLU}(\text{Deconv}_2(\text{ReLU}(\text{Deconv}_1(\phi(x)))))$.

- **Third layer:** This layer produces a 16-channel feature map, resizing the image back to its original input size.

  A Sigmoid activation function is applied after this final layer. The Sigmoid activation bounds the output pixel values between zero and one, suitable for grayscale image reconstruction and binary classification.

  Let Deconv_3 denote the operation of the third transposed convolutional layer.

  The reconstructed output image is

  $$\hat{x} = \text{Sigmoid}(\text{Deconv}_3(\text{ReLU}(\text{Deconv}_2(\text{ReLU}(\text{Deconv}_1(\phi(x))))))). \quad (4.12)$$

  The full model operation is $\hat{x} = D(E(x))$, where E is the encoder and D is the decoder.

4.4.2 EfficientNet-based Autoencoder Model

The second model implemented is an EfficientNet-based Autoencoder. This design leverages transfer learning principles, making this approach suitable for industrial applications where resource constraints are to be factored while achieving optimal performance. The model consists of an encoder and a decoder, where this encoder, denoted by E_{Eff}, rests on the EfficientNet-B0 model with its pretrained weights set to True. This model was chosen due the balance it strikes between model size and computational efficiency. The modification occurs at the first convolutional layer where the model accepts the grayscale input image. Thus, the layer that was originally designed to accept RGB images, is replaced with single-channel variant layer.

The encoder processes 128x128 sized input image x. Through the EfficientNet-B0 extraction pipeline, the input images are downsampled to capture a compact representation of hierarchical features $\phi_{Eff}(x)$ in the bottleneck layer, from low-level textures and edges to high-level patterns.

The decoder, denoted by D_{Eff}, is of a lightweight architectural design. It contains six transposed convolutional layers of the encoder $\phi_{Eff}(x)$ which upsamples the encoded features from the 4x4 spatial dimensions back to the original 128x128 image size.

Let $\psi_0 = \phi_{Eff}(x)$

ReLU activation is applied after the first five layers to introduce non-linearity while the Sigmoid activation is applied after the final sixth layer to bound the output pixels between zero and one.

The decoder layers operate sequentially:

- **First layer:** This layer converts from 1280 to 128 channels and upsamples the spatial dimensions from 4×4 (of the bottleneck representation) to 8×8, beginning the reconstruction process with significant dimensionality reduction.

  Let $\phi_{Eff}(x) \in \mathbb{R}^{4\times4\times1280}$ be the output of the encoder.

  $\psi_1 = \text{ReLU}(\text{Deconv}_1(\psi_0))$, where $\psi_1 \in \mathbb{R}^{8\times8\times128}$.
- **Second layer:** Further channel reduction occurs from 128 to 64 channels while upsampling the feature maps from 8×8 to 16×16 dimensions.

  $\psi_2 = \text{ReLU}(\text{Deconv}_2(\psi_1))$, where $\psi_2 \in \mathbb{R}^{16\times16\times64}$.
- **Third layer:** Continued expansion reduces channels from 64 to 32 while doubling spatial dimensions from 16×16 to 32×32.

  $\psi_3 = \text{ReLU}(\text{Deconv}_3(\psi_2))$, where $\psi_3 \in \mathbb{R}^{32\times32\times32}$.
- **Fourth layer:** This layer produces a 16-channel feature map while expanding spatial dimensions from 32×32 to 64×64.

  $\psi_4 = \text{ReLU}(\text{Deconv}_4(\psi_3))$, where $\psi_4 \in \mathbb{R}^{64\times64\times16}$.
- **Fifth layer:** Channel reduction continues from 16 to 8 channels with simultaneous upsampling from 64×64 to 128×128 dimensions.

  $\psi_5 = \text{ReLU}(\text{Deconv}_5(\psi_4))$, where $\psi_5 \in \mathbb{R}^{128\times128\times8}$.
- **Sixth layer:** The final reconstruction layer uses a standard convolution (not transposed) to reduce from 8 channels to the single grayscale output channel while maintaining the 128×128 spatial dimensions.

  $\hat{x} = \text{Sigmoid}(\text{Conv}_6(\psi_5))$, where $\hat{x} \in \mathbb{R}^{128\times128\times1}$ is the reconstructed image.

The full decoder operation is $\hat{x} = D_{Eff}(\phi_{Eff}(x))$.
The complete model operation is $\hat{x} = D_{Eff}(E_{Eff}(x))$.

4.4.3 Patch Description Network (PDN) model

The third model is a Patch Description Network that also leverages transfer learning by knowledge distillation from a deep pretrained teacher network to train a lightweight student network. It balances the challenge involved in capturing complex features while maintaining computational efficiency. A WideResNet-101 model tranfers its learned representations to a compact four-layer convolutional neural network exactly for patch level analysis.

Student Network

The student network (PDN) starts with a single-channel input that accepts the grayscale images, pre-processing them to a 224x224 pixel size resolution.

- **First layer:** The first convolutional layer transforms the single input channel into 16 feature maps using a 9×9 kernel with stride 1 and padding 4, followed by ReLU activation. This relatively large kernel size enables the network to capture broader spatial patterns from the initial input.

  Padding:
  $$224 \times 224 \rightarrow (224 - 9 + 2 \times 4)/1 + 1 = 224 \tag{4.13}$$
- **Second layer:**The second layer expands from 16 to 32 channels

- **Third layer:** This layer expands from 32 to 64 channels
- **Fourth layer:** The final layer produces 64 output feature maps. Each intermediate layer maintains the same 9×9 kernel configuration with stride 1 and padding 4, ensuring spatial dimensions are preserved throughout the forward pass.

$$\phi student(x) \in \mathbb{R}^{224\times224\times64} \tag{4.14}$$

The final convolutional layer omits any activation function, producing raw feature representations that can be directly compared with teacher network outputs during the distillation process. The full student network operation can be written as $\phi_{student}(x) = S_{PDN}(x)$.

A particularly noteworthy aspect of the PDN architecture is its receptive field calculation. With four layers each using 9×9 kernels, the effective receptive field reaches exactly 33×33 pixels, calculated as 9 + 3×(9-1) = 33. This receptive field size represents a carefully balanced design choice: large enough to capture meaningful local patterns and spatial relationships, yet small enough to maintain computational efficiency and enable detailed patch-level analysis.

Teacher Network

The teacher component utilizes a pretrained WideResNet-101 model, T_{WRN}, chosen for its proven capability in learning rich, hierarchical feature representations Zagoruyko and Komodakis [2016]. To make it suitable for the distillation task, the original classification head is removed, with the network truncated before the global average pooling layer to preserve spatial feature maps rather than producing a single classification vector.

$$\psi_1 = \text{Conv1}x1(TWRN(x_{rgb})), \text{where } \psi_1 \in \mathbb{R}^{H_T\times W_T\times64} \tag{4.15}$$

with H_T, W_T being the reduced spatial dimensions.

The raw teacher network outputs feature maps with 2048 channels at reduced spatial resolution due to the depth and downsampling operations inherent in the WideResNet architecture. To align with the student network's output format, a 1×1 convolutional projection layer reduces the channel dimension from 2048 to 64, matching the PDN's output channels. Subsequently, bilinear interpolation upsamples the teacher's feature maps to match the original input resolution, ensuring spatial correspondence with the student network's outputs.

While the PDN processes single-channel grayscale images, the pretrained WideResNet expects three-channel RGB input, thus the modification. This system addresses this mismatch by replicating the grayscale channel$x \in \mathbb{R}^{224\times224\times1}$ three times before feeding images to the teacher network $x_{rgb} \in \mathbb{R}^{224\times224\times3}$, enabling utilization of pretrained weights while maintaining consistency in the distillation process.

4.4.4 Optimization

The models are optimized using Adaptive Moment Estimation (Adam) optimizer, whose choice is due to the fact that it is simpler to tune as a hyperparameter, and it often leads to faster convergence during the training process Zhang [2018], a feature that is crucial for real-time quality inspection in this project's context. The default learning rate of 0.001 with the PyTorch library is used for training. An optimal

learning rate will lead to stable convergence along with the optimal speed. All the models have an epoch limit set to 30 epochs.

However, an early stopping regularization mechanism with patience of 5 is set to prevent overfitting. This technique halts training when the model's performance does not improve over a number of epochs on the validation set. The chosen value of 5 is set to balance between reactivity and stability. A too low patience is likely to stop training prematurely while one too high may overfit because training will continue even after validation loss has stopped improving. A patience of 5 will give the model time to recover from short-term fluctuations and continue improving where liable.

4.4.5 Statistical Threshold Determination

After training, the reconstruction errors for each image is calculated based on the mean squared errors. The calculated errors form a distribution which is representative of the model's reconstruction performance on the normal images. The reconstruction function is denoted by

$$g : \mathbb{R}^{H \times W} \longrightarrow \mathbb{R}^{H \times W}. \tag{4.16}$$

With the anomaly score defined by:

$$\{e_i\}_{i=1}^{N}, \quad e_i = \left\| x_i - g(x_i) \right\|_2^2, \tag{4.17}$$

where x_i is the i-th normal training image and g is the trained autoencoder's reconstruction function.

x is classified as anomalous if $e_i > \tau$, where τ is a calibrated threshold.

From this set of errors, the average reconstruction error is computed as

$$\mu = \frac{1}{N} \sum_{i=1}^{N} e_i, \tag{4.18}$$

and the standard deviation:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \big(e_i - \mu\big)^2}. \tag{4.19}$$

A threshold is calibrated based on the statistical distribution of the reconstruction errors from the normal training data samples. This threshold calculation is based on the standard three-sigma rule. This rule is derived from the statistical Z-score test for outlier detection under the assumption that the train data is of a normal distribution. A threshold is set at Z > 3 or -3, corresponding to the tails of a normal distribution. The Z-score aids to see how far away from the mean (in standard deviations) the reconstruction error is from the average, making outlier detection easy. Any data point exceeding the threshold means it is more than three times the standard deviation above the mean.

Using the 3-sigma rule is equivalent to applying a Z-score with a threshold of 3.

$$\tau = \mu + 3\,\sigma. \tag{4.20}$$

This ensures that approximately 99.7% of normal images will fall below the threshold and serve as a basis for classifying the anomalies.

When applied to an unseen test image, reconstruction error of the models can be denoted by

$$e_{\text{test}} = \left\| x_{\text{test}} - g(x_{\text{test}}) \right\|_2^2. \tag{4.21}$$

The binary decision is made as follows:

$$\text{label}(x_{\text{test}}) = \begin{cases} 0, & \text{if } e_{\text{test}} \leq \tau, \quad 0 = \text{normal}, \\ 1, & \text{if } e_{\text{test}} > \tau, \quad 1 = \text{anomalous}. \end{cases} \tag{4.22}$$

4.5 Supervised Learning

Under the supervised learning approach, two models were designed to serve as baselines for the unsupervised learning approach.

4.5.1 Pretrained ResNet18 Model

The first model is a modified ResNet-18 model that serves a feature extractor. Its pretrained weights are maintained, thus leveraging features learned such as edges, textures and shapes, from the ImageNet dataset it was trained on. It is a deep convolutional network with residual connections. These skip connections allow gradients to flow directly through the network, enabling the training of much deeper networks while avoiding the vanishing gradient problem. The architecture consists of four main residual blocks, each containing multiple convolutional layers with batch normalization and ReLU activation functions. The residual blocks progressively reduce spatial dimensions while increasing channel depth, creating a hierarchical feature representation. The network begins with a 7×7 convolutional layer followed by max pooling, then proceeds through residual blocks with 64, 128, 256, and 512 channels respectively.

4.5.2 Class Imbalance Management

Weighted cross-entropy loss is implemented, where class weights are computed as the inverse frequency of each class in the training dataset. For each class, the weight is computed as the reciprocal of its frequency, then normalized so that all weights sum to unity. This weighting scheme forces the model to pay equal attention to both classes during training, regardless of their natural frequency distribution in the dataset. This approach ensures that the rarer anomalous samples receive higher weights during loss computation, preventing the model from becoming biased toward predicting the majority normal class.

4.5.3 Custom 5-layer Convolutional Neural Network Model

The network follows a classical CNN design pattern with five convolutional blocks, each consisting of a convolutional layer followed by ReLU activation and max pooling.

- **First layer:** The network begins with a single 3-channel input (RGB images) and expands to 16 feature maps.
- **Second layer to Fifth Layer:** These layers maintain this 16-channel depth. This constant channel depth design prioritizes computational efficiency over feature complexity.

Each convolutional layer uses 3×3 kernels with padding=1, ensuring that the spatial dimensions are preserved before pooling. The consistent use of ReLU activation functions introduces non-linearity, enabling the network to learn complex relationships in the data.

The five MaxPool2d layers with 2×2 kernels and stride 2 systematically reduce the spatial dimensions by half at each stage. Starting from 224×224 inputs, the spatial size after all pooling operations becomes $224 \div (2^5) = 7 \times 7$. This aggressive dimensionality reduction serves dual purposes: It reduces computational load and creates translation invariance, making the model less sensitive to small spatial shifts in anomalous features.

The transition from convolutional to two fully connected layers occurs through a flattening operation, converting the 16×7×7 feature maps into a 784-dimensional vector.

- **First layer:** The first fully connected layer compresses this to 128 dimensions, followed by ReLU activation.
- **Second layer Layer:** The final layer reduces to a single output unit with sigmoid activation, producing a probability score between 0 and 1 for binary classification.

4.5.4 Optimization

The ResNet model is optimized using Adaptive Moment Estimation (Adam) optimize with a learning rate of 0.0001 to balance training speed and stability. This lower learning rate will preserve the pretrained features while giving space to the model to adaptively learn features from the given dataset. The choice of RMSprop optimizer for the with a learning rate of 1e-4 is a more conservative approach to parameter updates aimed at stable convergence given the relatively simple architecture.

4.5.5 Loss Functions

The Mean Square Error loss function calculates the difference between the original x and reconstructed image $\hat{x}$ on pixel-wise level.

The objective is to minimize the MSE:

$$\mathcal{L}_{\text{MSE}}(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2, \tag{4.23}$$

where N is the total number of pixels in the image.

A small MSE loss means that the average squared differences between the input image and reconstructed image pixels is minimal, and thus, the models are close to replicating the training image as accurately as possible.

For all the three unsupervised models, the training minimizes the Mean Squared Error (MSE) loss function. For the patch description network, MSE between the student and feature maps are minimized. This pixel-wise comparison in feature space encourages the student to learn similar spatial activation patterns as the teacher, effectively transferring the teacher's understanding of normal image patterns.

For the ResNet model, the MSE is used to calculate the difference between the original x and reconstructed image $\hat{x}$ on pixel-wise level. The CNN Model employs Binary Cross-Entropy (BCE) loss, appropriate for the binary classification task.

Analyzing Performance

For both supervised and unsupervised learning methods, experiments are conducted on two groups of data. The first being the original data set while for the second set, augmentation techniques are applied and combined with the original data. The design of the model architectures remain the same for training on both augmented and unaugmented training datasets.

For a fair performance comparison of both supervised and unsupervised methods, the same unseen dataset was used. This unseen unaugmented test set comprises of both normal and anomalous images. Annotation was done and were labels created for all the normal and anomalous images for all images. For the unsupervised learning, labels only serve as performance benchmarks during model testing, while labels were used for training and testing for the supervised learning approach.

For both supervised and unsupervised approaches, the performance metrics used are accuracy, precision, recall and F1-score. A confusion matrix is generated to provide details into true positives, false positives, true negatives, and false negatives. False Negative Rate (FNR) and False Positive Rate (FPR), expressed as percentages, are calculated to provide additional detailed understanding of misclassification pattern.

For the supervised approach, a standard threshold of 0.5 is set for the binary classification, to convert the sigmoid output probabilities to hard predictions.

4.6 Prediction

For unsupervised learning, prediction rule is denoted by

$$f : \mathbb{R}^{H\times W} \longrightarrow \{0, 1\} \quad \text{(after thresholding)}, \tag{4.24}$$

where $f(x)$ assigns 0 (normal) or 1 (anomalous) to input features $x \in \mathbb{R}^{H\times W}$.

The Autoencoder's optimal parameters $\hat{\theta}$ are obtained by

$$\hat{\theta} = \arg\min_{\theta} \sum_{(x_i,y_i)\,\in\,\mathcal{D}_{\text{train}}^{(\text{unsup})}} L_{\text{unsupervised}}\big(g_\theta(x_i),\, x_i\big), \tag{4.25}$$

with $L_{\text{unsupervised}}$ often taken as the Mean Squared Error (MSE).

For unsupervised learning (knowledge distillation,PDN), the student and teacher networks are

$$S_\theta : \mathbb{R}^{H\times W} \longrightarrow \mathbb{R}^{D'}, \quad T : \mathbb{R}^{H\times W} \longrightarrow \mathbb{R}^{D'}, \tag{4.26}$$

where D' is the feature dimension (e.g., 64 channels). The anomaly score is defined as

$$S_{\text{PDN}}(x) = \big\|\, S_\theta(x) \;-\; T(x)\big\|_F^2, \tag{4.27}$$

and x is classified as anomalous if $S_{\text{PDN}}(x) > \tau$. The student's parameters $\hat{\theta}$ are learned by minimizing a distillation loss over the normal training set:

$$\hat{\theta} = \arg\min_{\theta} \sum_{(x_i,y_i)\,\in\,\mathcal{D}_{\text{train}}^{(\text{unsup})}} L_{\text{distillation}}\big(S_\theta(x_i),\, T(x_i)\big), \tag{4.28}$$

where a typical choice for $L_{\text{distillation}}$ is the MSE between student and teacher feature maps.

For supervised learning, the model is parameterized by θ, written as f_θ. The optimal parameters $\hat{\theta}$ are found by minimizing the loss $L_{\text{supervised}}$ over the labeled training set:

$$\hat{\theta} = \arg\min_{\theta} \sum_{(x_i, y_i) \in \mathcal{D}_{\text{train}}^{(\text{sup})}} L_{\text{supervised}}\big(f_\theta(x_i),\, y_i\big), \tag{4.29}$$

where examples of $L_{\text{supervised}}$ include Binary Cross-Entropy (BCE) for the CNN model and Cross-Entropy Loss for the ResNet model.

In all cases, the optimization (finding $\hat{\theta}$) is performed using Adam algorithm except in the CustomCNN model which uses RMSprop.

Chapter 5

Implementation

This chapter details the implementation of deep machine learning models selected to address the central research question regarding the computational cost and time efficiency of unsupervised vs supervised learning approaches for real-time quality inspection in manufacturing. The models were implemented using the Python language and the PyTorch library. Common elements for data loading, training, validation loop structure, and performance evaluation metrics were standardized across methods to ensure a fair comparison.

5.1 Data Collection and Pre-processing

The original data that was to be worked on for this project was on metal data. Unfortunately, this was unavailable during the project and so improvisation was made.

A larger dataset of concrete data ᴄağlar Fırat Özgenel [2018], focusing on detecting potential cracks was used. This is publicly accessible data explicitly designed for research in automated crack detection. In total, 40,000 images of 255 MB total size is collected. The images are each of original dimension 227 x 227 size pixels with JPEG lossless format type in RGB colorspace with gray level colors. The images are originally of 458 high-resolution (4032x3024 pixel) photographs of several buildings on the Middle East Technical University campus. The final dataset was generated by extracting patches from the collected high resolution images, a method in Zhang et al. [2016] paper. It is licensed by CC BY 4.0, a Creative Commons Attribution 4.0 International license. The dataset does not contain any sensitive information such as user data or personal identifiers, and so, no extra steps are taken to protect privacy.

The use of this dataset aids in answering the research question by providing a suitable benchmark in research in the robustness of anomaly detection under varying surfaces. Knowledge from the results of this experiment can be transferred and applied in other sectors within manufacturing and production.

From visual inspection of some of the images, notable features can be identified. Generally, the images are not smooth surfaced but compose of varying textures and finishes as seen in Figure 5.3. The images have varying illumination conditions, either in the overall image or in specific regions within the image. Image b of Figure image 5.3 has lower brightness levels as compared to a and c of the normal images 5.3. Some images have occlusions due to foreign objects as seen in the top left corner of image a of Figure 5.2 and an unfamiliar red color seen in image b of Figure 5.2. Cracks within images can be identified by sharp intensity changes creating overall color distortions as seen in Figure 5.2. There may be single or multiple cracks within

an image as in image a of 5.2 again. In such images, other regions of the image have a fairly uniform color as seen in image c of Figure 5.2. These intensity changes are of varying shapes and sizes, appearing with rugged or jagged edges, often creating an illusion of depth. Cracks typically run across the surface of the concrete. Even with images with comparable overall darkness as in image c of Figure 5.2, the presence of cracks is demonstrated with sharper intensity changes. Thus, the presence of cracks, and occlusions will typically be tagged as anomalous while images devoid of this, will be labeled normal. Figure 5.3 shows samples of a normal/negative image from the train dataset, Figure 5.2 displays a sample of an anomalous/positive image from the validation and test datasets. An unusual pattern in the surface will be significant because of how it deviates from the normal pattern. This is what our algorithms will detect as an anomaly.

The dataset composes of 20,000 normal (negative) and 20,000 anomalous (positive) images, indicating two class categorization. Further division was done before using it for model implementation. The dataset is split into training, validation and test subsets. For unsupervised training, the training set contains 70% of the normal dataset, which is 14,000 in total. No anomalous images are included in this set. The validation set contains 10,000 images which is 50% of the anomalous dataset, as well as 15% of the normal dataset, adding an extra 3,000 images. In total, the validation set contains 13,000 images. The Test set mimics the validation set, containing 13,000 images: 10,000 images from the other 50% of the anomalous dataset, and 3,000 images from 15% of normal images. A table for easier visualization is shown in Table 5.1.

The filenames of each of the images were encoded with key identifiers. A python program is used for this process. A random seed of 42 is set for reproducibility in shuffling the files. Images are then copied into the respective output folders of train, validation and test. During copying, the image is renamed with a prefix, for example 'pos_train' and a counter to create a new encoded filename. The naming convention reflects the image label and dataset split. A separate CSV labels file is generated where the filenames are labeled 0 for normal/negative images and labeled 1 for anomalous/positive images.

Since the baseline for this task is supervised learning, labels generated aid in the training for this approach. The dataset split for supervised training is displayed in Table 5.2. In the unsupervised learning approach, the generated labels serve as a benchmark to evaluate how well the model is, in accurately predicting positive and negative images. However, for use in the unsupervised context, none of the labels are used during training - only as a benchmark during testing on the unseen dataset.

Dataset Split	Content	Normal	Anomalous	Total	Purpose
Training Set	Normal Data Only	14 000	0	14 000	Unsupervised model training.
Validation Set	Mix of Normal and Anomalous	3 000	10 000	13 000	Monitor overfitting during training.
Test Set	Mix of Normal and Anomalous	3 000 (remaining 15% of total normal)	10 000 (remaining 50% of total anomalous)	13 000	Final model evaluation.

Table (5.1) Dataset splits and their composition for Unsupervised training

Online pre-processing is applied where all images within the training pipeline are transformed through resizing and converted to grayscale to reduce computational complexity.

As this dataset is drawn from the manufacturing industry, its use is within the purpose of this project. The data set is representative of the real-world scenario where objects have normal surface as well as naturally occurring or artificially pro-

Dataset Split	Content	Normal	Anomalous	Total	Purpose
Training Set	Mix of Normal and Anomalous	13 600	8 000	21 600	Unsupervised model training.
Validation Set	Mix of Normal and Anomalous	3 400	2 000	5 400	Monitor overfitting during training.
Test Set	Mix of Normal and Anomalous	3 000	10 000	13 000	Final model evaluation.

Table (5.2) Dataset splits and their composition for Supervised Training

(a) Normal Sample 1 (b) Normal Sample 2 (c) Normal Sample 3

Figure (5.1) Comparison of Normal Samples

duced surface defects. By detailing the collection methods, data structure, and preprocessing techniques, a clear understanding of the inherent challenges and strengths of working with this data is provided.

5.2 Model Implementation

In the process of implementing the models, experiments were conducted using two variations of the training data to investigate the effectiveness of data augmentation techniques in improving model performance under both supervised and unsupervised approaches.

Subsequently, all experiments under both unsupervised and supervised approaches are re-trained using augmented data. This augmented data is generated by applying augmentation techniques to the original training dataset in a separate, offline pre-processing step. The augmentation is not performed dynamically during model training but as a prior step, resulting in a larger, combined dataset of real and generated images.

Augmentation approach

For the unsupervised approach, augmentation was applied only to the normal images within the training dataset. The goal is to increase the variability of the normal data distribution that the unsupervised models learn from. For the supervised approach, augmentation is applied to increase the count and variability of the anomalousim ages. These augmentation efforts take into account the class imbalance present between normal and anomalous images in the training dataset.

The design of the model architectures remains consistent whether training on the original or the augmented training datasets. Crucially, the performance of all trained models, regardless of whether they were trained on original or augmented data, is evaluated on the same unseen, unaugmented test dataset. This standardized

(a) Anomalous Sample 1 (b) Anomalous Sample 2 (c) Anomalous Sample 3

Figure (5.2) Comparison of Anomalous Samples

evaluation benchmark ensures a fair comparison of model performance and generalization ability to real, unprocessed data. The labels for this test set, though not used in training for the unsupervised models, are essential for evaluating their performance against the ground truth. This dual training approach (on original and augmented data) within the implementation aims to directly investigate how enriching the training data impacts the performance and potentially the computational characteristics of both supervised and unsupervised anomaly detection methods, answering the central research question.

5.2.1 Unsupervised Learning Models

These models are trained using only normal, defect-free data samples. The goal is to identify anomalous instances by detecting deviations from the learned distribution or features of normal data.

Convolutional AutoEncoder (CAE) Model:

Consists of an encoder and a decoder. The encoder has three convolutional layers with ReLU activation after the first two, outputting raw features at the bottleneck. The decoder mirrors the encoder with three transposed deconvolutional layers, using ReLU activation after the first two and Sigmoid activation after the final layer to bound pixel values between 0 and 1. This model contains 46529 parameters.

EfficientNet-based Autoencoder Model:

Leverages transfer learning by adapting a pre-trained EfficientNet-B0 model as the encoder. This model is chosen for its balance between size and computational efficiency. The first convolutional layer of the encoder is modified to accept single-channel grayscale input images. The decoder is lightweight, composed of six transposed convolutional layers using ReLU activation (first five layers) and Sigmoid activation (final layer) to upsample encoded features back to the original image size. This model contains 5579773 parameters.

Patch Description Network (PDN) Model:

Leverages transfer learning through knowledge distillation from a deep pre-trained WideResNet-101 teacher network. A four-layer convolutional neural network serves as the student (PDN).

Student Network (PDN): A lightweight, fully convolutional 4-layer network.

It accepts single-channel grayscale images pre-processed to 224x224. It uses 3x3 kernels with padding=1 to preserve spatial dimensions in most layers, with the first layer using a 9x9 kernel. ReLU activation is applied after the intermediate layers, but the final convolutional layer omits activation to produce raw feature representations. This model contains 540608 parameters

Teacher Network: A pre-trained WideResNet-101 model is used and truncated before the global average pooling layer to preserve spatial feature maps in feature extraction. To align with the student, a 1x1 convolutional layer reduces the teacher's 2048 channels to 64, matching the student's output channels.

The anomalies are detected through the comparison of the student and teacher learned feature representations. The student, through training, reproduces the feature maps generated by the teacher on normal data.

5.3 Baselines

The supervised learning models, the Pretrained ResNet18 Model and the Custom 5-layer Convolutional Neural Network Model, are explicitly designed to serve as baselines for the unsupervised learning approaches. Their performance, computational cost, and time efficiency are compared against the unsupervised models.

Pretrained ResNet18 Model:

A modified ResNet-18 model used as a feature extractor, is implemented with its pre-trained weights retained from prior training on the ImageNet dataset. It includes residual connections to facilitate training of deeper networks. The final fully-connected layer is modified to output two values for binary classification (normal/anomalous). This model contains 11,177,538 parameters.

Custom 5-layer Convolutional Neural Network Model:

A classical CNN is designed with five convolutional blocks. Each block consists of a convolutional layer, ReLU activation, and Max Pooling. The network starts with a 3-channel input and maintains a 16-channel depth in subsequent layers, prioritizing computational efficiency. Five max pooling layers reduce the spatial dimensions, resulting in 7x7 feature maps before flattening. A flattening operation transitions to two fully connected layers, with the final layer using sigmoid activation for binary classification probability prediction. The model contains 110,337 parameters.

5.4 Results and Evaluation

Comprehensive logging was undertaken and the following information was recorded in one logging sheet: timestamp, model name, loss function, optimization function, learning rate, batch size, epochs run, train time, inference time, train, validation and test sizes, image size, threshold, primary evaluation metrics (accuracy, precision, recall, f1 score), true positives, false positives, false negatives, true negatives, model parameters, and model size. Evaluation on how well a model is able to correctly identify where an image is an anomaly or not, is determined by accuracy, precision, recall and f1 score.

For unsupervised learning, the validation set is used to monitor performance and apply early stopping. Training is limited to 30 epochs, with early stopping set with a patience hyperparameter of 5, implemented to prevent overfitting.

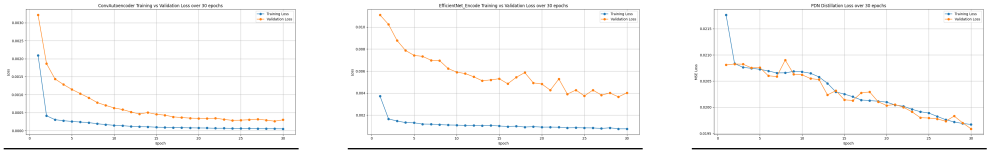
A confusion matrix of the prediction classification outcomes is also generated. The True Positives highlight the cases where the model correctly predicts the anomalous images (positive class). The False Negatives highlight the case where the model incorrectly predicts that an image is normal (negative), when, in fact, it is anomalous (positive). This means that the model missed an actually anomalous image. True Negative prediction means that the model correctly predicted a normal (negative) image. False Positive Prediction means that the model incorrectly predicted that an image was anomalous (positive) when it is actually normal, predicting a false alarm.

For standardization, all the methods under both supervised and unsupervised approaches, have similar implementation elements for data loading, training & validation loop structure.

5.5 Training Performance

5.5.1 Unsupervised Training Performance on Original Data

This section displays the training performance via loss curves of the unsupervised learning methods on original unaugmented data.



(a) Conv AutoEncoder (b) EfficientNet (c) PDN

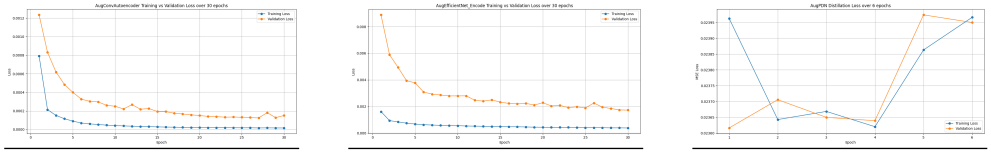
Figure (5.3) Training Performance on Original Data under Unsupervised Approach

The 5.3a curve presents the training and validation loss for the Convolutional Autoencoder model over 30 epochs on original unaugmented data. The Training Loss starts moderately low (around 0.0021) and decreases rapidly in the beginning, then slows down, approaching zero by epoch 30. The Validation Loss starts higher (around 0.0033) and decreases rapidly like the training loss. It remains consistently higher than the training loss throughout the 30 epochs. Both loss curves show a clear decrease, indicating successful learning. The gap between validation and training loss narrows in the early epochs and remains relatively stable, with validation loss staying above training loss. This indicates an optimal fit as the gap between them remains minimal. The model is learning the underlying patterns of the data without overfitting.

The 5.3b figure shows the training and validation loss for the EfficientNet-based AutoEncoder model over 30 epochs on original unaugmented data. The Training Loss starts moderately low (around 0.0038) and steadily decreases, reaching a low value by the end of training. The Validation Loss starts very high (around 0.011) and decreases sharply in the initial epochs. After the initial drop, it continues to decrease but exhibits more pronounced fluctuations between epochs 15 and 30 compared to the training loss. Both curves show an overall decreasing trend, suggesting the model is learning. However, the validation loss is consistently higher than the training loss. Although the initial large gap decreases as training progresses, it does not minimize significantly, suggesting the model is overfitting.

Figure 5.3c shows the distillation loss (MSE Loss) for the PDN model over 30 epochs on original unaugmented data. The Training Loss starts high (around 0.0217)

and gradually decreases over the 30 epochs, with some minor fluctuations. The Validation Loss starts slightly lower than the training loss (around 0.0208) and also gradually decreases with fluctuations. The training and validation loss curves are relatively close throughout the 30 epochs and cross multiple times. This close to zero gap between the losses suggests that the model is an optimal fit, generalizing well to the validation data.



(a) Conv AutoEncoder (b) EfficientNet (c) PDN

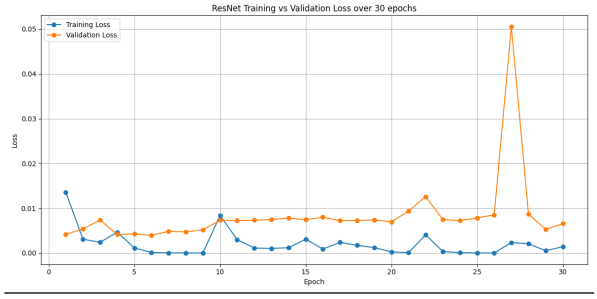
Figure (5.4) Training Performance on Augmented Data under Unsupervised Approach

5.5.2 Unsupervised Training Performance on Augmented Data

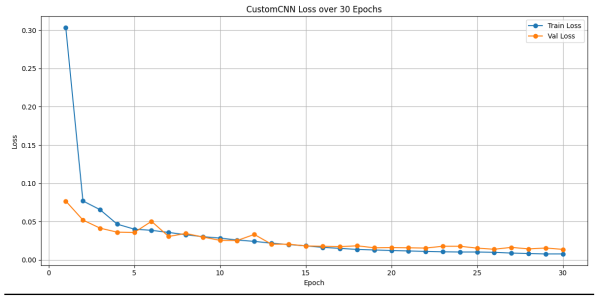
The 5.4a curve shows the training and validation loss for the Convolutional AutoEncoder model trained over 30 epochs on augmented data. The Training Loss starts relatively high (around 0.0008) and decreases rapidly in the initial epochs, then continues to decrease more slowly, approaching a low value (close to 0) by epoch 30. The Validation Loss starts significantly higher (around 0.00125) than the training loss. It also decreases rapidly initially and then slows down, remaining consistently higher than the training loss throughout the 30 epochs. Both curves show a clear learning trend as the loss decreases with increase in epochs. The gap between validation and training loss reduces significantly in the early epochs. Close to the end of training, the validation loss seems to slightly increase, but not significantly.

The 5.4b figure displays the training and validation loss for the EfficientNet-based AutoEncoder model over 30 epochs on augmented Data. The Training Loss starts low (around 0.0016) and steadily decreases over the 30 epochs, ending closer to zero. The Validation Loss on the contrary begins very high (around 0.009) and drops sharply in the first few epochs. It continues to decrease at a slower rate, staying consistently higher than the training loss. The large initial gap between validation and training loss narrows considerably in the early stages of training and then remains relatively stable, with validation loss being higher than training loss. However, the gap does not close as the final epoch 30 is reached.

This 5.4c curve shows the distillation loss (specifically, MSE Loss) for the Patch Description model over a much shorter period of 6 epochs on the augmented data. The Training Loss starts high (around 0.02396), decreases initially, then shows an increase at epoch 3, continues to sharply increase after epoch 4. The Validation Loss starts lower (around 0.02362), increases at epoch 2, then decreases at epoch 4, followed by a sharp increase at epoch 5 and a slight drop at epoch 6. The curves fluctuate significantly and cross multiple times within these 6 epochs. The observed fluctuations from unstable training and the limited number of epochs, is triggered by early stopping which activates due to model performance not improving.



(a) ResNet18



(b) CNN

Figure (5.5) Training Performance on Original Data under Supervised Approach

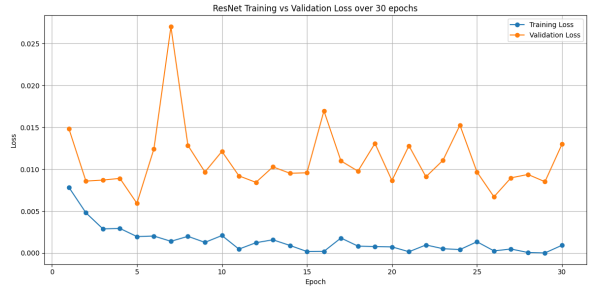
5.5.3 Supervised Training Performance on Original Data

Figure 5.5a presents the training and validation loss for the ResNet18 model over 30 epochs on original unaugmented data. The Training Loss starts moderately low (around 0.01) and decreases steadily with very minimal spikes, approaching zero by epoch 30. The Validation Loss, although slightly above the training loss, also starts low. It remains consistently low until a spike around epoch 27, and then drops significantly at epoch 30 with a closing gap. This gap indicates an optimal fit as the gap between them remains minimal except the occurrence of the spike. The model is learning the underlying patterns of the data without overfitting.

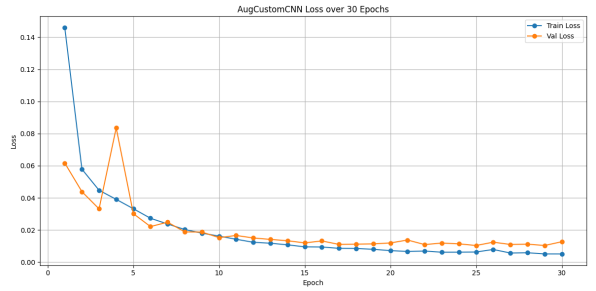
The 5.5b curve presents the training and validation loss for the Custom CNN model over 30 epochs on original unaugmented data. The Training Loss starts moderately low (around 0.07) and decreases steadily in the beginning, then slows down, approaching zero by epoch 30. The Validation Loss starts higher (around 0.30) and decreases rapidly. Both loss curves show a clear decrease, indicating successful learning. The training and validation loss curves are relatively close through to the end of the 30 epochs and overlap multiple times. This indicates an optimal fit as the gap between them remains almost zero. The model is learning the underlying patterns of the data without overfitting.

5.5.4 Supervised Training Performance on Augmented Data

The 5.6a loss graph presents the training and validation loss for the ResNet18 model over 30 epochs on augmented data. The Training Loss starts lower (around 0.008), decreases steadily. It remains consistently lower than the validation loss. The Validation Loss starts relatively low (around 0.015) and increases rapidly around epoch 7, then sharply dips, and fluctuations until epoch 30. The gap between validation and training loss widens with each fluctuation in the validation loss. This indicates an



(a) ResNet18



(b) PDN

Figure (5.6) Training Performance on Augmented Data under Supervised Approach

overfitting of the model. The model is unable to learn the underlying relationship between the image data points.

The 5.6b curve presents the training and validation loss for the Custom CNN model over 30 epochs on augmented data. The Training Loss starts relatively higher (around 0.14) and decreases rapidly in the beginning, then slows down, approaching zero by epoch 30. The Validation Loss starts lower (around 0.06), decreases and rapidly fluctuates by epoch 4. It drops and remains consistently lower than the training loss from epoch 11, through to epoch 30. Both loss curves show a decrease that indicates successful learning. The gap between validation and training loss narrows in the latter epochs and remains relatively stable, with validation loss staying above training loss. This indicates an optimal fit as the gap between them remains minimal. The model is learning the underlying relationship between data points without overfitting.

5.6 Comparative Analysis

The following tables display an overview of the results from all experiments conducted. In Table 5.3, the results are on the original unaugmented data experiment group.

Original Unaugmented Data

Model	Epochs	Train time(s)	Inference time(s)	Accuracy	Precision	Recall	F1 score	Size (MB)	Approach
Convolutional AutoEncoder	30	706.63	13.76	.7565	.9901	.6904	.8135	.18	Unsupervised
EfficientNet	30	4949.38	40.38	.9035	.9937	.8801	.9334	21.58	Unsupervised
PDN	30	39785.32	152.03	.9386	.9936	.9262	.9587	2.07	Unsupervised
ResNet18	30	2804.12	29.89	.9992	.9997	.9993	.9995	42.72	Supervised
CNN	30	1800.93	23.31	.9952	.9995	.9943	.9969	.43	Supervised

Table (5.3) Experimental results on Original Unaugmented data

In Table 5.4, the results are on the calculated False Negative Rate (FNR) and False Positive Rate (FPR) on the original unaugmented data experiment group.

FNR & FPR on Original Unaugmented Data

Model	TN	FP	FN	TP	FNR(%)	FPR(%)
Convolutional AutoEncoder	2931	69	3096	6904	30.96	2.3
EfficientNet	2944	56	1199	8801	11.99	1.87
PDN	2940	60	738	9262	7.38	2.0
ResNet18	2997	3	7	9993	0.07	0.1
CNN	2995	5	57	9943	0.57	0.17

Table (5.4) FNR & FPR on Original Unaugmented Data

Comparative results on Original Unaugmented Data

Fastest Training Time:

1. Convolutional AutoEncoder (706.63s)
2. CNN (Supervised)
3. ResNet18 (Supervised)
4. EfficientNet
5. Patch Description Network (PDN) (39785.32s)

Fastest Inference Time:

1. Convolutional AutoEncoder (13.76s)
2. CNN (Supervised)
3. ResNet18 (Supervised)
4. EfficientNet
5. PDN (152.03s)

Highest F1 Score:

1. ResNet18 (Supervised) (0.9995)
2. CNN (Supervised)
3. PDN
4. EfficientNet
5. Convolutional AutoEncoder (0.8135)

Lowest False Positives:

1. ResNet18 (Supervised) (3)
2. CNN (Supervised)
3. EfficientNet
4. PDN
5. Convolutional AutoEncoder (69)

Lowest False Negatives:

1. ResNet18 (Supervised) (7)
2. CNN (Supervised)
3. PDN
4. EfficientNet
5. Convolutional AutoEncoder (3096)

Augmented Data

Model	Epochs	Train time	Inference time	Accuracy	Precision	Recall	F1 score	Size (MB)	Approach
Convolutional AutoEncoder	30	4092.01	30.0	.8724	.9924	.8405	.9102	.18	Unsupervised
EfficientNet	30	13238.55	32.75	.8389	.9848	.803	.8847	21.59	Unsupervised
PDN	6	35170.74	407.11	.7722	.9969	.706	.8266	2.07	Unsupervised
ResNet18	30	20304.28	30.07	.9995	.9996	.9997	.9997	42.72	Supervised
CNN	30	3618.15	22.59	.9981	.9993	.9982	.9987	.43	Supervised

Table (5.5) Experimental results on Augmented data

In Table 5.6, the results are on the calculated False Negative Rate (FNR) and False Positive Rate (FPR) on the original Augmented data experiment group.

FNR & FPR on Augmented Data

Model	TN	FP	FN	TP	FNR(%)	FPR(%)
Convolutional AutoEncoder	2936	64	1595	8405	15.95	2.13
EfficientNet	2876	124	1970	8030	19.7	4.13
PDN	2978	22	2940	7060	29.4	0.73
ResNet18	2996	4	3	9997	0.03	0.13
CNN	2993	7	18	9982	0.18	0.23

Table (5.6) FNR & FPR on Augmented Data

Comparative results on Augmented Data

Fastest Training Time:

1. CNN (Supervised) (3618.15s)
2. Convolutional AutoEncoder
3. EfficientNet
4. ResNet18 (Supervised)
5. PDN (35170.74s)

Fastest Inference Time:

1. CNN (Supervised) (22.59s)
2. Convolutional AutoEncoder
3. ResNet18 (Supervised)
4. EfficientNet
5. PDN (407.11s)

Highest F1 Score:

1. ResNet18 (Supervised) (0.9997)
2. CNN (Supervised)
3. Convolutional AutoEncoder
4. EfficientNet
5. PDN (0.8266)

Lowest False Positives:

1. ResNet18 (Supervised) (4)
2. CNN (Supervised)
3. PDN
4. Convolutional AutoEncoder
5. EfficientNet (124)

Lowest False Negatives:

1. ResNet18 (Supervised) (3)
2. CNN (Supervised)
3. Convolutional AutoEncoder
4. EfficientNet
5. PDN (2940)

Smallest Model Size:

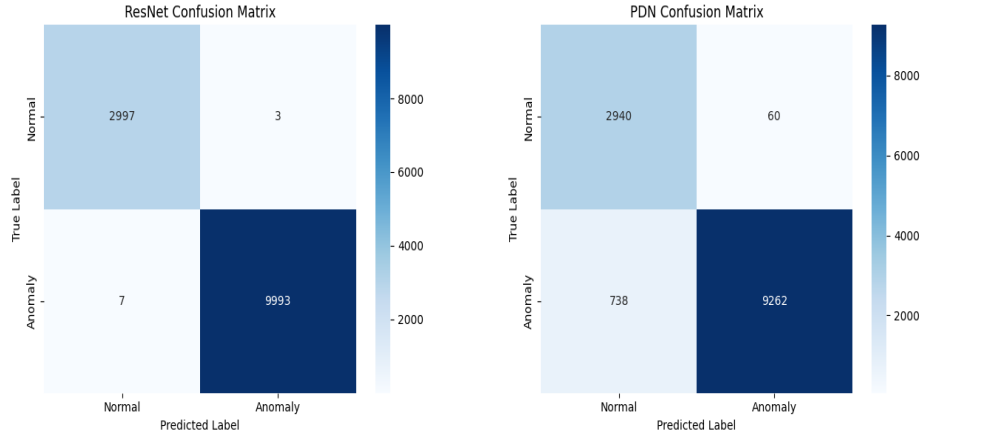
1. Convolutional AutoEncoder (0.18mb)
2. CNN (Supervised)
3. PDN
4. EfficientNet
5. ResNet18 (Supervised) (42.72mb)

Chapter 6

Discussion

This chapter analyzes the experimental findings presented in the Results and Evaluation 5.4 section of Chapter 5 in the context of the research question and broader implications for industrial quality inspection.

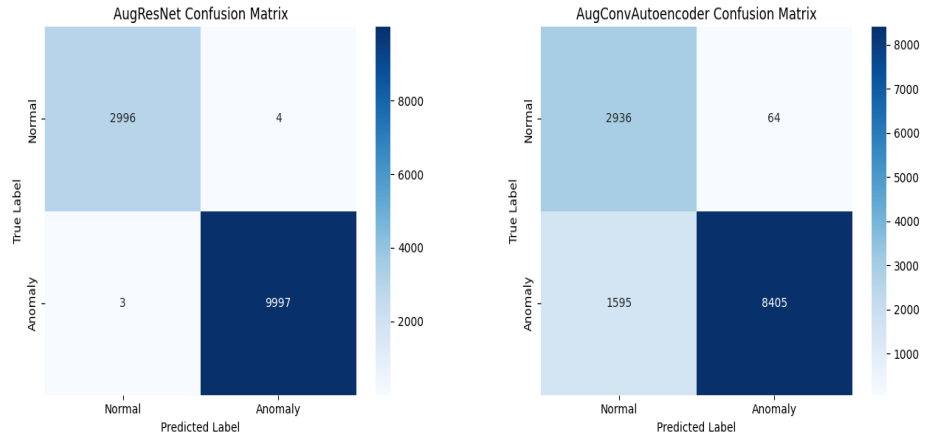
The experiments utilize a concrete crack dataset as a proxy for metal coating quality assessment, evaluating three unsupervised deep learning models (Convolutional AutoEncoder (CAE), EfficientNet-based Autoencoder, and Patch Description Network (PDN)) against two supervised baselines (Pretrained ResNet18 and a Custom 5-layer Convolutional Neural Network). Training is conducted on both original unaugmented data and augmented data, with evaluation performed on an unseen, unaugmented test set for all models. Performance is measured using Accuracy, Precision, Recall, F1-score, True Positives (TP), False Positives (FP), False Negatives (FN), True Negatives (TN), False Negative Rate (FNR) and False Positive Rate (FPR), expressed as percentages. Computational cost and time efficiency were assessed via Model Size, Training Time, and Inference Time.



(a) ResNet18 (b) Patch Description Network

Figure (6.1) Confusion matrix on models trained on original Unaugmented data

The Confusion matrices displayed in Fig 6.1 are of the best performing models based on highest F1 score, for both unsupervised and supervised learning approaches, trained on Unaugmented data. The selected models are the ResNet18 model under the supervised approach, and the Patch Description Network under the unsupervised approach.



(a) ResNet18 (b) Convolutional AutoEncoder

Figure (6.2) Confusion matrix on model trained on Augmented data

The Confusion matrices displayed in Fig 6.1 are of the best performing models based on highest F1 score, for both unsupervised and supervised learning approaches, trained on Augmented data. The selected models are the ResNet18 model under the supervised approach, and the Convolutional AutoEncoder under the unsupervised approach.

6.1 Results link to research question

The central research question explored in this project is: "What is the computational cost and time efficiency of unsupervised vs. supervised learning approaches for real-time quality inspection in manufacturing?". The experimental results on the concrete crack dataset, presented in Tables 5.3 and 5.5, provide significant insights into this question by comparing the performance and efficiency metrics of the models under study.

6.1.1 Analyzing the results from the Original Unaugmented Data

With Table 5.3 and 5.4 as references:

- **Performance (F1 Score, FP, FN, FNR, FPR):**
  - ▷ The supervised models, ResNet18 (F1: 0.9995, FP: 3, FN: 7, FPR: 0.10%, FNR: 0.07%) and CNN (F1: 0.9969, FP: 5, FN: 57,FPR: 0.17%, FNR: 0.57%), demonstrate significantly higher F1 scores and substantially lower false positive and false negative rates compared to the unsupervised models. Their extremely low FNR (ResNet18 at 0.07% representing 7 missed anomalies, CNN at 0.57% representing 57 missed anomalies) is particularly crucial in quality inspection where failing to detect actual defects can have severe consequences
  - ▷ Among unsupervised models, the PDN (F1: 0.9587, FP: 60, FN: 738) and EfficientNet-based AutoEncoder (F1: 0.9334, FP: 56, FN: 1199) performed better than the CAE (F1: 0.8135, FP: 69, FN: 3096). A high number of false negatives, like the CAE's 3096, with a high False Negative Rate (FNR) of 30.96%, represents a significant failure to detect actual anomalies, which

is a critical concern in quality inspection. Conversely, the PDN achieved a much lower FNR of 7.38%, making it more effective at identifying true anomalies than the EfficientNet-based AE, which had an FNR of 11.99% on unaugmented data

- **Computational Cost (Model Size):**
  - ▷ The unsupervised CAE had the smallest model size (0.18 MB).
  - ▷ It is followed by the supervised CNN (0.43 MB), the unsupervised PDN (2.07 MB), the unsupervised EfficientNet (21.58 MB), and the supervised ResNet18 (42.72 MB). Smaller model size is crucial for deployment in resource-constrained industrial environments.
- **Time Efficiency (Train and Inference Time):**
  - ▷ The unsupervised CAE exhibits the fastest training time (706.63s) and fastest inference time (13.76s) among all models on original data.
  - ▷ The supervised CNN (Train: 1800.93s, Inference: 23.31s) and ResNet18 (Train: 2804.12s, Inference: 29.89s) had moderate times.
  - ▷ The unsupervised PDN shows the slowest training time (39785.32s) and inference time (152.03s). This suggests a significant trade-off between computational complexity and speed. The slow training can be attributed to the knowledge distillation overhead from the large, deep WideResNet-101 teacher network. This adds a computational load per batch during training since loss calculation also uses the teacher network's features.

6.1.2 Analyzing the results from the augmented data

With Table 5.5 as reference:

- **Impact of Augmentation on performance:**
  - ▷ ResNet18 achieved an F1 score of 0.9997 (TN:2996, FP:4, FN:3, TP:9997,FPR: 0.13%, FNR: 0.03%), yielding almost perfect separation of classes. Its remarkably low FNR of 0.03% (representing only 3 missed anomalies) further demonstrates its superior performance.
  - ▷ CNN reached an F1 score of 0.9987 (TN:2993, FP:7, FN:18, TP:9982, FPR: 0.23%, FNR: 0.18%), with only a few more misclassifications than ResNet18. These results represent very slight reductions in false positive rates and false negative rates compared to training on the original dataset. ResNet18 still attained the highest F1 (0.9997) when trained with the augmented data.
  - ▷ Convolutional AutoEncoder (CAE) achieved an F1 of 0.9102 (TN:2936, FP:64, FN:1595, TP:8405, FPR: 2.13%, FNR: 15.95%), showing an increase than when trained on original data. Augmentation boosted the CAE's F1 score (from 0.8135 on the original data), significantly reducing its FNR from 30.96% to 15.95%, representing an almost 50% improvement in detecting actual anomalies.
  - ▷ EfficientNet-based AutoEncoder achieved an F1 of 0.8847 (TN:2876, FP:124, FN:1970, TP:8030, FPR: 4.13%, FNR: 19.70%). After augmentation, the F1 dropped slightly (from 0.9334 to 0.8847), with FN rising from 1199 to 1970, indicating a modest degradation in the recall - model's ability to detect actual anomalies.

- ▷ PDN's performance worsened markedly under augmentation, with an F1 of 0.8266 (TN:2978, FP:22, FN:2940, TP:7060,FPR: 0.73%, FNR: 29.40%). This represents a substantial increase in its FNR from 7.38% to 29.40% and a drop in F1 (from 0.9587 on original data to 0.8266). While its FPR improved to 0.73%, the significant rise in FNR is a major disadvantage. Such a decline suggests that the augmentation strategies may have disrupted the PDN's student/teacher feature-map matching, causing the instability during its brief training run as seen in Fig 5.4c.

Impact of Augmentation on Time Efficiency:

- ▷ Augmentation generally increased training time for all models due to the larger dataset size CAE's training time rose from 706 s to 4092.01 s due to the larger augmented dataset. EfficientNet's training time increased from 11 000 s to 13 238.55 s. PDN, which originally took 15 000 s, now required 35 170.74 s for just 6 epochs.
- ▷ CAE's inference time went from 13.76 s to 30.00 s. EfficientNet's inference rose slightly to 32.75 s. PDN's inference time jumped from 152.03 s to 407.11 s. ResNet18 and CNN remained near 30 s and 22.59 s, respectively, with only minor changes.
- ▷ Overall, augmentation substantially improved the CAE's detection performance at the cost of longer training/inference, slightly degraded EfficientNet, and destabilized PDN. Supervised models (ResNet18, CNN) remained extremely robust and saw only marginal gains in reducing misclassifications.

Summary relating to research question:

The results demonstrate a significant trade-off between performance and data requirements. Supervised methods, particularly the ResNet18 model, achieve the highest accuracy and lowest error rates (FPR and FNR) when sufficient labeled data is available. However, acquiring substantial labeled anomaly data is a known, costly challenge in industrial settings.

Unsupervised methods address this data labeling limitation by training only on normal data. Among the unsupervised methods tested, the CAE offers the best efficiency in terms of model size (0.18 MB) and fastest inference time (approximately 13.76s on original data, 30.0s on augmented data). While its performance (F1, FPR, FNR) is lower than supervised models, its speed and small size make it a highly efficient option for real-time quality inspection where fast inference is paramount and a certain level of detection accuracy is acceptable without extensive labeling. Augmentation significantly improved the CAE's performance, particularly its FNR.

The PDN shows promising performance with original data (low FNR of 7.38%) and a moderate size (2.07 MB) but is significantly slower in inference than AE-based models and was negatively impacted by the specific augmentation strategy applied, leading to a substantial increase in its FNR to 29.40% and unstable training. The EfficientNet-based AE is larger and slower than the CAE but generally performs better on original data.

For real-time quality inspection, inference time is a critical factor. The CAE demonstrates the fastest inference time among all tested models, unsupervised or

supervised. While its performance is not as high as the top supervised models, its efficiency makes it a strong candidate for deployment where speed and low resource usage are priorities.

If any experiments "failed" or did not behave as expected, the most prominent example is the decline in PDN performance with augmented data. As reasoned above, this is likely due to the specific augmentation strategy applied to the normal training data impacting the knowledge distillation process. The short training duration observed for the PDN on augmented data in Figure 5.3c corroborates this performance decline.

6.2 Limitations and Future Work

6.2.1 Limitations of this work:

- **Dataset Specificity:** The primary limitation is the use of a concrete crack dataset as a proxy for the intended metal coating quality assessment case study due to data unavailability. While concrete crack detection serves as a relevant benchmark for varying surfaces, results might not directly translate to the specific texture, patterns, and anomaly characteristics unique to metal coatings.
- **Scope of Methods:** The study focused on a selection of deep learning-based unsupervised and supervised methods. Other relevant unsupervised approaches like clustering-based methods, Normalizing Flows, or traditional methods mentioned in the literature review are not implemented for direct comparison in the experiments.
- **Augmentation Strategy:** The specific offline augmentation techniques applied might not be universally optimal for all models, as suggested by the performance drop observed for the PDN and EfficientNet-based AutoEncoder with augmented data.
- **Hyperparameter Tuning Depth:** While some hyperparameters like learning rates and optimization functions are chosen, extensive tuning across a wide range of possibilities is limited, which could potentially improve the performance of some models.
- **Hardware Dependence:** Computational time measurements (Training Time, Inference Time) are dependent on the specific hardware used for implementation. This hardware used here is a virtual machine is a NC4as machine: Ubuntu 24.04 LTS, 16GB VRAM NVIDIA T4 GPU, 4 vCPUs, 28 GB RAM, 1TB data drive not detailed in the sources, making direct comparison of absolute times across different computing environments challenging.

6.2.2 Potential avenues for future work:

Additional experiments that could be performed:

- Investigate and test alternative augmentation strategies model-specific or adaptive data augmentation strategies for particularly for knowledge distillation-based methods as used in the PDN model, and the EfficientNet-based AutoEncoder model, focusing on methods that enhance normal data variability without introducing features that interfere with the student-teacher feature comparison mechanism, and transfer learning.

- Perform more extensive hyperparameter tuning for all models, particularly learning rates and optimization settings, to ensure optimal performance for each architecture and dataset variant, as the default settings might not be universally optimal.
- Evaluate the models on the original metal coating dataset if it becomes available, to confirm the generalizability of the findings across different material surfaces.
- Conduct ablation studies on the custom CNN architecture or other models to evaluate the contribution of individual components (e.g., layers, activation functions) to performance and efficiency.
- Implement and compare a wider range of unsupervised methods such as GAN-based anomaly detection or representation-based methods like PatchCore, or methods leveraging attention mechanisms, inpainting, or student-teacher training beyond Knowledge Distillation, to provide a more comprehensive comparative analysis.
- Investigate the impact of the training duration on the PDN model, especially with augmented data, to see if longer or more stable training would improve its performance.
- Conducting a more detailed cost-benefit analysis that incorporates not just computational cost but also the cost associated with data acquisition and labeling for supervised methods versus the development and tuning cost for unsupervised methods in a realistic industrial context.
- Investigating the application of explainable AI (XAI) techniques to unsupervised anomaly detection models to provide insights into why a specific instance is flagged as anomalous, increasing trust and usability in an industrial context.
- Evaluating the robustness of the best-performing unsupervised models to different types and severities of anomalies not explicitly present in the training data, as anomalies in real production can be highly diverse.
- Combining defect data in the training of the unsupervised models to investigate the performance of the models when trained on not only normal data, but both normal and defect data.

Under these assumptions, primarily the characteristics of the concrete dataset, the selected subset of methods, and the specific implementations and hyperparameters used, the conclusions regarding the performance-efficiency trade-offs hold within the scope of this study.

Chapter 7

Conclusion

Anomaly Detection is an integral and critical subject area in machine learning with significant applications in various industries, including manufacturing, particularly for real-time quality inspection. The increasing need for automation in manufacturing is fueled by the limitations of traditional manual quality control, which is prone to human error, is time-consuming, and expensive. Automation aims to ensure consistent product quality, enhance safety, and minimize loss.

A key challenge in implementing automated quality inspection using machine learning, especially with supervised methods, is the scarcity and diversity of labeled defect data. Anomalies in manufacturing can be infrequent and highly diverse, making the collection of large, labeled datasets required by supervised models difficult, labor-intensive, and costly. This limitation often serves as a deterrent in training supervised machine learning models to achieve admissible accuracy for automation.

This project directly addresses this challenge by focusing on exploring and developing unsupervised machine learning algorithms. The motivation is to provide a generalized approach that requires only normal, defect-free data samples for training, thereby potentially reducing implementation time, time to market, and resource requirements while maintaining reliable detection capabilities.

Using metal coating quality assessment as an intended case study, the project aimed to compare the computational cost and time efficiency of unsupervised versus supervised learning approaches for this task. However, due to data unavailability, the experimental validation was performed on a publicly accessible concrete crack detection dataset, serving as a relevant benchmark for anomaly detection on varying surfaces.

Regarding the computational cost and time efficiency trade-offs between unsupervised and supervised methods for real-time industrial quality inspection, this project implemented and evaluated three unsupervised models: the Convolutional AutoEncoder (CAE), an EfficientNet-based Autoencoder, and a Patch Description Network (PDN) via Knowledge Distillation. These were compared against two supervised baseline models: a Pre-trained ResNet18 and a Custom 5-layer Convolutional Neural Network. Experiments are conducted using both the original unaugmented dataset and an augmented version to investigate the impact of data enrichment on performance and efficiency. Performance is evaluated using standard metrics including Accuracy, Precision, Recall, F1-score, and Confusion Matrices (analyzing TP, FP, FN, TN), while efficiency was assessed based on Training Time, Inference Time, and Model Size.

The experimental results demonstrate that supervised methods achieve superior

anomaly detection performance (higher F1 scores and lower False Positive Rates/-False Negative Rates) when sufficient labeled data is available for training. The Pre-trained ResNet18 model showed the highest performance, achieving an F1 score of 0.9997 with an exceptionally low FNR of 0.03% and FPR of 0.13% on augmented data. However, it has a significantly larger model size and higher training times compared to other unsupervised alternative.

The project shows that unsupervised methods offer a viable solution for scenarios with limited labeled data, requiring only normal images for training. Among the unsupervised models tested, the Convolutional AutoEncoder (CAE) stands out for its exceptional efficiency, having the smallest model size (0.18 MB) and the fastest inference time (approximately 13.76s on original data, 30.00s on augmented data). While its detection performance is lower than the top supervised models (e.g., its FNR was 30.96% on original data, improving to 15.95% with augmentation, its speed and low computational cost make it highly suitable for real-time deployment in resource-constrained industrial settings where fast inference is paramount. Augmentation significantly improved the performance of the CAE (F1 score increased from 0.8135 to 0.9102), substantially reducing its FNR. The Patch Description Network (PDN) shows competitive performance on original data (FNR of 7.38%, FPR of 2.0%) but was negatively impacted by the specific augmentation strategy applied (FNR worsened to 29.40%, while FPR improved to 0.73%) and had higher inference times than AE-based models.

In conclusion, this study confirms the fundamental trade-off in industrial anomaly detection: supervised methods offer higher accuracy but are constrained by data labeling requirements, while unsupervised methods overcome the data constraint and can offer significant advantages in terms of computational efficiency and deployment time, especially simple architectures like the Convolutional AutoEncoder. The findings highlight the potential of the CAE for efficient real-time quality inspection where minimizing deployment resources and time-to-market is a primary goal, although the optimal choice of method ultimately depends on the specific application's tolerance for false positives (FPR) and false negatives (FNR) and the feasibility of acquiring sufficient labeled anomaly data. With the right augmentation techniques applied to increase count and variability, the CAE stands a chance at competing with supervised machine learning methods.

The research successfully demonstrates that unsupervised methods can operate without labeled anomaly data, addressing key aspects of the central research question "What is the computational cost and time efficiency of unsupervised vs. supervised learning approaches for real-time quality inspection in manufacturing?".

Bibliography

Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 622–637. Springer, 2019.

Ahad Alloqmani, Yoosef B Abushark, Asif Irshad Khan, and Fawaz Alsolami. Deep learning based anomaly detection in images: insights, challenges and recommendations. *International Journal of Advanced Computer Science and Applications*, 12 (4), 2021.

Rajeevan Arunthavanathan, Faisal Khan, Salim Ahmed, Syed Imtiaz, and Risza Rusli. Fault detection and diagnosis in process system using artificial intelligence-based cognitive technique. *Computers & Chemical Engineering*, 134:106697, 2020.

Kilian Batzner, Lars Heckler, and Rebecca König. Efficientad: Accurate visual anomaly detection at millisecond-level latencies. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 128–138, 2024.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4183–4192, 2020.

R. Chalapathy and S. Chawla. Deep learning for anomaly detection: A survey. Research report, arXiv preprint, arXiv:1901.03407, January 2019. Available at `https://arxiv.org/abs/1901.03407`.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. Published article, ACM Computing Surveys (CSUR), Volume 41, Issue 3, Pages 1-58, 2009. https://doi.org/10.1145/1541880.1541882.

Chieh-Yu Chen, Shi-Chung Chang, and Da-Yin Liao. Equipment anomaly detection for semiconductor manufacturing by exploiting unsupervised learning from sensory data. *Sensors*, 20(19):5650, 2020.

Praveen Chopra and Sandeep Kumar Yadav. Fault detection and classification by unsupervised feature extraction and dimensionality reduction. *Complex & intelligent systems*, 1:25–33, 2015.

Yajie Cui, Zhaoxiang Liu, and Shiguo Lian. A survey on unsupervised anomaly detection algorithms for industrial images. *IEEE Access*, 11:55297–55315, 2023.

Diana Davletshina, Valentyn Melnychuk, Viet Tran, Hitansh Singla, Max Berrendorf, Evgeniy Faerman, Michael Fromm, and Matthias Schubert. Unsupervised anomaly detection for x-ray images. *arXiv preprint arXiv:2001.10883*, 2020.

Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *International conference on pattern recognition*, pages 475–489. Springer, 2021.

Edsger W. Dijkstra. Go To Statement Considered Harmful. *Communications of the ACM*, 11(3):147–148, 1968. doi: 10.1145/362929.362947.

Thibaud Ehret, Axel Davy, Mauricio Delbracio, and Jean-Michel Morel. How to reduce anomaly detection in images to anomaly detection in noise. *Image Processing On Line*, 9:391–412, 2019.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

Ásgeir Daniel Hallgrímsson, Hans Henrik Niemann, and Morten Lind. Improved process diagnosis using fault contribution plots from sparse autoencoders. *IFAC-PapersOnLine*, 53(2):730–737, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Naoya Ishida, Yuki Nagatsu, and Hideki Hashimoto. Unsupervised anomaly detection based on data augmentation and mixing. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 529–533. IEEE, 2020.

Qingchao Jiang, Shifu Yan, Xuefeng Yan, Hui Yi, and Furong Gao. Data-driven two-dimensional deep correlated representation learning for nonlinear batch process monitoring. *IEEE Transactions on Industrial Informatics*, 16(4):2839–2848, 2019.

Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM), and IEEE Computer Society. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, New York, NY, USA, 2013. ISBN 978-1-4503-2309-3. 999133.

Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

Jiang Lin and Yaping Yan. A comprehensive augmentation framework for anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8742–8749, 2024.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–39, 2012.

Ruikang Liu, Weiming Liu, Zhongxing Zheng, Liang Wang, Liang Mao, Qisheng Qiu, and Guangzheng Ling. Anomaly-gan: A data augmentation method for train surface anomaly detection. *Expert Systems with Applications*, 228:120284, 2023.

Antonio Loureiro, Luis Torgo, and Carlos Soares. Outlier detection using clustering methods: a data cleaning application. In *Proceedings of KDNet Symposium on Knowledge-based systems for the Public Sector*. Springer Bonn, 2004.

Sankar Mahadevan and Sirish L Shah. Fault detection and diagnosis in process data using one-class support vector machines. *Journal of process control*, 19(10): 1627–1639, 2009.

Ting Mao, Yun Zhang, Huamin Zhou, Dequn Li, Zhigao Huang, and Huang Gao. Data driven injection molding process monitoring using sparse auto encoder technique. In *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 524–528. IEEE, 2015.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=BydrOIcle`.

MITRE. Common Weakness Enumeration: CWE-193: Off-by-one Error, December 2018. `https://cwe.mitre.org/data/definitions/193.html`.

Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, 54(2): 1–38, 2021.

Nils Gustav Erik Pettersson. Knowledge distillation for anomaly detection, 2023.

Van Phung and Eun Rhee. A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. *Applied Sciences*, 9:4500, 10 2019. doi: 10.3390/app9214500.

Jonathan Pirnay and Keng Chai. Inpainting transformer for anomaly detection. In *International Conference on Image Analysis and Processing*, pages 394–406. Springer, 2022.

Jinchuan Qian, Zhihuan Song, Yuan Yao, Zheren Zhu, and Xinmin Zhang. A review on autoencoder based representation learning for fault detection and diagnosis in industrial processes. *Chemometrics and Intelligent Laboratory Systems*, 231:104711, 2022.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 07–09 Jul 2015. PMLR.

Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14318–14328, 2022.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. URL `https://arxiv.org/abs/1409.1556`.

Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A Alemi, and Andrew G Wilson. Does knowledge distillation really work? *Advances in neural information processing systems*, 34:6906–6919, 2021.

Raghuveer Thirukovalluru, Sonal Dixit, Rahul K Sevakula, Nishchal K Verma, and Al Salour. Generating feature sets for fault diagnosis using denoising stacked auto-encoder. In *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–7. IEEE, 2016.

Tauno Toikka, Jouko Laitinen, and Kari T Koskinen. Failure detection and isolation by lstm autoencoder. In *World Congress on Engineering Asset Management*, pages 390–399. Springer, 2021.

D. M. Tsai and P. H. Jen. Autoencoder-based anomaly detection for surface defect inspection. Published article, Advanced Engineering Informatics, Volume 48, Article 101272, 2021. https://doi.org/10.1016/j.aei.2021.101272.

U. A. Usmani, A. Happonen, and J. Watada. A review of unsupervised machine learning frameworks for anomaly detection in industrial applications. Published book chapter, Lecture Notes in Networks and Systems, Springer, Cham, Volume 507, Pages from Intelligent Computing. SAI 2022, 2022. https://doi.org/10.1007/978-3-031-10464-$0_1$1.

Shashanka Venkataramanan, Kuan-Chuan Peng, Rajat Vikram Singh, and Abhijit Mahalanobis. Attention guided anomaly localization in images. In *European Conference on Computer Vision*, pages 485–503. Springer, 2020.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

Vincent Wilmet, Sauraj Verma, Tabea Redl, Håkon Sandaker, and Zhenning Li. A comparison of supervised and unsupervised deep learning methods for anomaly detection in images. *arXiv preprint arXiv:2107.09204*, 2021.

Xianghua Xie. A review of recent advances in surface defect detection using texture analysis techniques. *ELCVIA: electronic letters on computer vision and image analysis*, pages 1–22, 2008.

Z. Xie, Z. Zhang, J. Chen, Y. Feng, X. Pan, Z. Zhou, and S. He. Data-driven unsupervised anomaly detection of manufacturing processes with multi-scale prototype augmentation and multi-sensor data. Published article, Journal of Manufacturing Systems, Volume 77, Pages 26-39, January 2024a. https://doi.org/10.1016/j.jmsy.2024.08.027.

Zongliang Xie, Zhipeng Zhang, Jinglong Chen, Yong Feng, Xingyu Pan, Zitong Zhou, and Shuilong He. Data-driven unsupervised anomaly detection of manufacturing processes with multi-scale prototype augmentation and multi-sensor data. *Journal of Manufacturing Systems*, 77:26–39, 2024b.

Weiwu Yan, Pengju Guo, Zukui Li, et al. Nonlinear and robust statistical process monitoring based on variant autoencoders. *Chemometrics and Intelligent Laboratory Systems*, 158:31–40, 2016.

Zheng Yang, Binbin Xu, Wei Luo, and Fei Chen. Autoencoder-based representation learning and its application in intelligent fault diagnosis: A review. *Measurement*, 189:110460, 2022.

Shen Yin, Steven X Ding, Adel Haghani, Haiyang Hao, and Ping Zhang. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process. *Journal of process control*, 22(9):1567–1581, 2012.

Jaemin Yoo, Tiancheng Zhao, and Leman Akoglu. Data augmentation is a hyperparameter: Cherry-picked self-supervision for unsupervised anomaly detection is creating the illusion of success. *arXiv preprint arXiv:2208.07734*, 2022.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL http://arxiv.org/abs/1605.07146.

Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. Draem-a discriminatively trained reconstruction embedding for surface anomaly detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8330–8339, 2021.

Lei Zhang, Fan Yang, Yimin Daniel Zhang, and Ying Julie Zhu. Road crack detection using deep convolutional neural network. In *2016 IEEE international conference on image processing (ICIP)*, pages 3708–3712. IEEE, 2016.

Lingrui Zhang, Shuheng Zhang, Guoyang Xie, Jiaqi Liu, Hua Yan, Jinbao Wang, Feng Zheng, and Yaochu Jin. What makes a good data augmentation for few-shot unsupervised image anomaly detection? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4345–4354, 2023.

Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2, 2018. doi: 10.1109/IWQoS.2018.8624183.

Mohammad Zolfaghari and Hedieh Sajedi. Unsupervised anomaly detection with an enhanced teacher for student-teacher feature pyramid matching. In *2022 27th International Computer Conference, Computer Society of Iran (CSICC)*, pages 1–4. IEEE, 2022.

Cağlar Fırat Özgenel. Concrete crack images for classification, 2018. URL https://doi.org/10.17632/5y9wdsg2zt.1.