Operating visual machine learning systems in a manufacturing environment

Philipp Mascha

Angaben zur Veröffentlichung / Publication details:

Mascha, Philipp. 2026. "Operating visual machine learning systems in a manufacturing environment." Augsburg: Universität Augsburg.

Operating Visual Machine Learning Systems in a Manufacturing Environment

Doctoral dissertation
submitted in fulfillment of the requirements for the degree of
Dr. rer. nat.
at the Faculty of Applied Computer Science of the University of Augsburg

submitted by

Philipp Mascha

2025

First examiner:	Prof. Dr. Jörg Hähner
Second examiner:	Prof. Dr.-Ing. Johannes Schilp
Additional oral examiner:	Prof. Dr.-Ing. Lars Mikelsons

Date of oral examination: 2 October 2025

I thank my colleagues at the Osram manufacturing IT and automation department for being who they are: a great bunch to work with. Also Jochen Haeck at the Herbrechtingen plant, who provided all the insight and feedback required to shape a wacky proof of concept into a working application.

I thank my wife for shouldering the weight of this work alongside me. And my children for giving me joy and perspective when everything seemed all too overwhelming.

Symbols

General notation

a	A vector or sample.
$\mathbf{M}$	A matrix.
$\mathbf{M} * \mathbf{K}$	Discrete convolution between matrix $\mathbf{M}$ and kernel $\mathbf{K}$.
$a_i, a[i]$	Value of vector at index i.
$\delta_{[cond]}$	1 if *cond* is true, 0 otherwise.
$\lvert a \rvert$	Absolute value of a vector.
$\lVert a \rVert_p$	p-Norm of a vector.
$\lVert a, b \rVert$	Distance between samples a and b.
(x, y)	A sample x with label y.
$\hat{y}$	Ground-truth.

Models and functions

E	Error function.
$\mathcal{L}$	A loss function.
acc	(Class averaged) accuracy.
$f(x; \theta_f)$	A model or function parameterized by θ_f.
enc	Encoder model of an auto-encoder.
dec	Decoder model of an auto-encoder.
L	Latent space of an auto-encoder.
$M(x; \theta_M)$	Base classification model.
$M_d(x; \theta_d)$	Drift detection model.

Constants, variables and parameters

θ_c	Sensor calibration parameters.
θ_p	Parameters of outside perturbations.
d	Distance metric.
$\overline{d}$	Average measured distance value of system.
d_R	Actual drift present in the system.
d_{max}	Detection threshold for out of distribution samples.
p_{ood}	Threshold probability for out of distribution samples.
l_σ	Sigma limit for SPC.
μ	Mean.
σ	Standard deviation.

Datasets

$\|\mathbb{S}\|$	Number of samples contained in $\mathbb{S}$.
$\mathbb{S}^T$	Set of training samples.
$\mathbb{S}^P$	Set of samples observed during live operation.
$\mathbb{S}^E$	Calibration set used for evaluation and calibration of a trained model.
$\mathbb{S}^{OOD}$	Set of all out-of-distribution samples.
$\mathbb{B}$	Sample buffer used for retraining.

Probabilities and Distributions

$\mathcal{S}$	Distribution of all observable data samples.
$\mathcal{D}$	Distribution of d values over a sample set.
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and standard deviation σ.
$\mathbb{S} \sim \mathcal{S}$	Set $\mathbb{S}$ has underlying distribution $\mathcal{S}$.
$x \sim \mathcal{S}$	Random variable x has distribution $\mathcal{S}$.
$p(a)$	Probability of event a to occur.
$p(a\|b)$	Conditional probability of event a to occur when observing b.

F_D	Cumulative distribution function over D.
$P[a]$	Cumulative probability over discrete realization a.

Acronyms

AE Auto-Encoder.

AI artificial intelligence.

ANN artificial neural network.

AUC area under curve.

CCAE classification coupled Auto-Encoder.

CDF cumulative distribution function.

CLT central limit theorem.

CMA-ES covariance matrix adaptation evolution strategy.

CNN convolutional neural network.

CV computer vision.

DevOps software development and IT operations.

DL deep learning.

DUT device under test.

EA evolutionary algorithm.

EI expected improvement.

ES evolution strategy.

ETL extract, load and transform.

FFT fast fourier transform.

FPR false positive rate.

GA genetic algorithm.

GAN Generative Adversarial Network.

GPU graphical processing unit.

IoU intersection over union.

KDE kernel density estimate.

LCL lower control limit.

LLM large language model.

LRP layer-wise relevance propagation.

MD3 margin density drift detection.

ML machine learning.

MLOps machine learning operations.

MLP multi layer perceptron.

MNIST Modified National Institute of Standards and Technology Database.

MSE mean squared error.

NLP natural language procession.

OC organic computing.

OOD out-of-distribution.

PCA principal component analysis.

PCB printed circuit board.

PoC proof of concept.

PoI probability of improvement.

QA quality assurance.

ReLU rectified linear unit.

ResNet Residual Network.

ROC receiver operating characteristic.

RoI region of interest.

SGD stochastic gradient descent.

SL specification limit.

SPC statistical process control.

SR Super Resolution.

SSIM structural similarity index.

TPR true positive rate.

UCB upper confidence bound.

UCL upper control limit.

VAE variational Auto-Encoder.

VMLC visual machine learning control.

XAI explainable AI.

Contents

Chapter 1

Introduction

This is where the story begins.

Walter Moers

In recent years, machine learning (ML) steadily found its way into our everyday life. Countless applications like image filters, translation services or chat bots have become so commonplace we do not even recognize them anymore. But in the field of high-throughput manufacturing ML applications often miss the step from proof of concept (PoC) to continuous live operation. Especially in the domain of visual quality control potential is plenty and performance promising. But concerns regarding viability, stability and trust hinder ML projects from being fully rolled out.

In this thesis, I will outline the domain of manufacturing and its peculiar requirements. I will show how these impact ML performance, especially when considering long-term stability. Subsequently, I will present a practical framework, visual machine learning control (VMLC), that tackles these problems through an observer/ controller architecture. It encompasses an Auto-Encoder based metric that can measure dissimilarity between current measurements and training data underlying the classification model. By applying statistical process control (SPC) on this metric, control limits can be enforced to guarantee stability and react upon problems.

To handle any breach of limit various self-healing methods are presented. They repair the system through means of active learning, sensor self-calibration and better explanations of faulty classification. The framework is thoroughly evaluated on real-world manufacturing data and the viability of live operation is presented as a case study.

1.1 AI for Manufacturing

Artificial intelligence (AI) is on a steady victory march through countless domains of applications. Thanks to new advances in machine learning (ML), namely deep convolutional neural networks (CNNs) [KSH12] and their derivatives, problems deemed hard or even unsolvable ten years ago are a triviality today: Image classification [DK16], object localization [Zha+21a], natural language procession (NLP) [OMK20] or even speech recognition [Nas+19] can be done with super-human performance. Many applications, from funny smartphone filters in a camera app [MPP18] to graphics optimization in computer games [Che+20] find their way into everyday life. The latest wave of AI applications brought by generative models [Kin+14] even found its way into art through style transfer [Sin+21] and text-to-image generation [Ram+21].

It is obvious that ML could also provide many applications in the manufacturing domain [Wue+16]. There is plenty of potential for predictive maintenance [Sch18], optimization of production processes [Wei+19] or design of products [Edd+14; LW19].

This is also true for **optical quality inspection**, where each manufactured part has to be inspected by a human inspector. Based on apparent defects or imperfections the observer must decide whether the product is safe for further processing or has to be removed from production. This task is both costly and strenuous for the inspector, since it is highly monotonous and often physical demanding [WSL06].

Thus it seems natural to replace the human inspector with an automated system, consisting of a camera taking pictures of the product and an accompanying **classification system** to detect quality issues at its heart. In the past, this has been done through classical computer vision (CV) methods [Mal+03]. Deep learning has spawned new and better quality control applications designed for opto-semiconductor [Lin+19], mechanical [WCL18], pharmaceutical [Unn+19] or electronics [Tab+20; Kim+21a] manufacturing.

Despite this, many surveys done among industrial companies suggest that most applications never leave the status of PoC [BBC20]. Creating continuously operating ML applications is hindered by huge risks and challenges [Pla+23]. While this is often addressed to missing knowledge and the non-service oriented nature of manufacturing [KBC22], there seem to be other problems in place that hinder a wide adoption in the industry.

1.2 Thesis Goals

To formulate the central research question of this thesis:

How can an existing machine learning (ML) classification model used for visual quality control safely operate in an automated manufacturing environment, considering requirements like stability, explainability and maintainability?

To answer this question, my thesis aims to provide the following contributions towards the field of AI for manufacturing:

- Establishing manufacturing quality control as its own domain inside visual learning tasks, based on its peculiar features in data. Furthermore it aims to formulate the implications of these deviations and how they affect the standard ML approach.
- Provide an overview of existing methods to address these issues and evaluate their applicability towards industrial data.
- Propose and evaluate an unified architecture to handle all these problems and guarantee stable operation for ML-based classification inside manufacturing quality control. This shall be done either by combining existing methods or expanding upon them.
- Find a reliable implementation of SPC for visual data. The lack of a generally accepted method was already identified as a research gap in [Tra+22], but is required by customers and quality assurance (QA) experts that operate by Lean practices [WRJ90].
- Find novel concepts to repair an unstable ML system. This should be done as non-invasive as possible to not interrupt standard production flow.
- Make the ML system more transparent towards non-expert staff, preferably through improvements towards explainable AI (XAI).

1.3 Thesis Structure

To address all these goals, this thesis is separated into the following chapters:

Problem Definition and Motivation: This chapter will present the motivation for this thesis: A real-world application of automated quality inspection and the practical issues I had when running it in live production. I will go into detail why these issues arise and what makes industrial data especially sensitive towards them. I will then formulate requirements for a ML system to successfully operate over a long period of time.

Technical Background: Here I will present technology, research and state of the art required to design a system capable of running in continued live operation. I present the workings of artificial neural networks (ANNs), introduce the industry wide adopted method of SPC and explore tools to detect and act upon problems that arise in production.

Proposed Solution: In this chapter I will present VMLC, a system I designed to handle all these challenges in a unified approach, inspired by the observer controller architecture.

Experiments: Subsequently, I will evaluate the system and its components on real industrial data. Results are presented in both a qualitative and quantitative manner as well as a real-world application deployed to production.

Discussion: Conclusively, I discuss the findings and their implications of said experiments in detail.

Summary: The last chapter will summarize the whole thesis and outline its research contributions. Additionally, I will iterate over possible future research and further methods that may be considered in this domain.

Chapter 2

Problem Definition and Motivation

> Theory and practice sometimes clash. And when that happens, theory loses. Every single time.
>
> *Linus Torvalds*

Sometimes, the best concepts are derived from practical challenges. When I first sat down to work on this thesis, I wanted to solve the problem of measuring uncertainty in deep learning models. But it was hard to come up with novel ideas from a field where every intuitive solution had already been found by someone else.

But only after I had to solve the practical problems that came with the work I had to do at my company, operating machine learning models in a live production, it dawned on me where the real issues lie: The methods are there, but theoretical applications are based on misconceptions, stemming from a lack of real world context predominant in academia. Many approaches that succeed in benchmarks struggle with real world data, which is imperfect and incomplete. Solving these shortcomings became an engineering problem which spans a multitude of research fields from handling class imbalance, active learning, anomaly detection and novelty discovery to reliability, explainability and many more.

In this chapter, I will present the project that inspired me to develop visual machine learning control (VMLC), the heart of this research thesis. Based on its implications, I will introduce the fundamental issues that arise when operating inside the industrial manufacturing domain. Furthermore, I

will formulate attributes a machine learning (ML) system needs to fulfill to succeed in this peculiar setting.

2.1 Laser Welding Application

Figure 2.1: The product being manufactured: A classical H16 halogen lamp, used in headlights for various vehicles.

The inspiration for this thesis came from a practical application: A visual control of a laser welded component. It is located in an assembly line in the Herbrechtingen manufacturing plant of Osram GmbH. The product in question is an H16 headlight lamp (see figure 2.1) sold in large quantities to automotive suppliers in a business-to-business model.

As is usual in automotive production, the manufacturing process holds itself accountable to Lean standards [For96]. This means that quality and process stability comes before everything else. It is expected by the customer that not a single faulty product is shipped, making reliable controls a necessity.

2.1.1 Original Solution

The device under test (DUT) is an electrode welded onto a contact fin (see figure 2.2 on the next page). The quality of the weld is automatically inspected by a camera system through a pinhole in the socket. To do so, the lamp is automatically picked up and rotated into place by an automated

Figure 2.2: Electrode welded to the contact fin. The socket was sawed open for demonstration purposes.

rotary table. Light is provided by a diffuse light source from below (see figure 2.3 on the following page).

Figure 2.4 on the next page shows examples of the images taken. The high amount of problematic artifacts is apparent: A lot of lighting noise, reflections and inconsistencies exist. This is mainly credited to the control being a retro-fit, put into place on an existing manufacturing line were this kind of control was not planed during initial design. The pinhole setup and problematic lighting conditions result in the high visual complexity and variance of the captured images.

Originally, the images were classified by traditional computer vision (CV) techniques like edge detection and pattern matching. Results where never satisfying with many over-rejects (properly welded parts where sorted as defect). Even worse, many actual defects were not properly detected (see figure 2.5 on the following page). The original developer even commented:

The system is basically a random number generator.

The complexity and variation of the visual material was simply to hard to

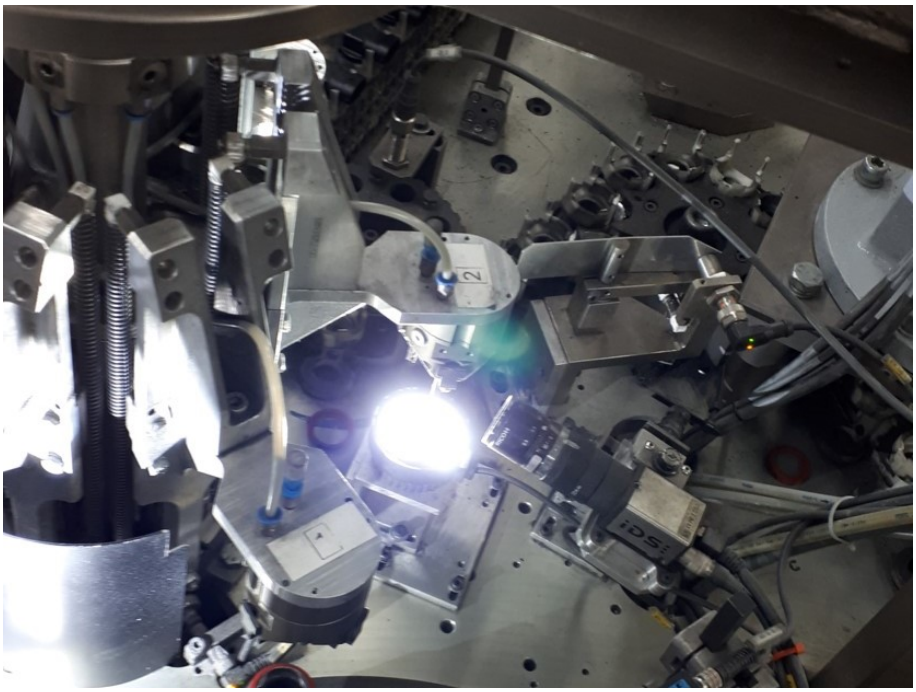


Figure 2.3: Camera setup for automated inspection.

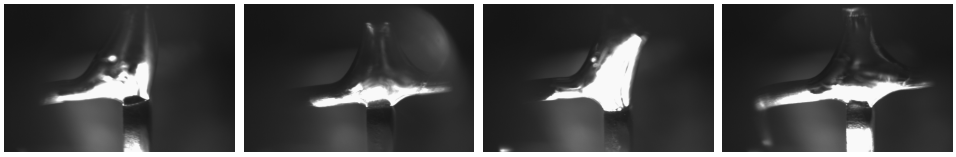
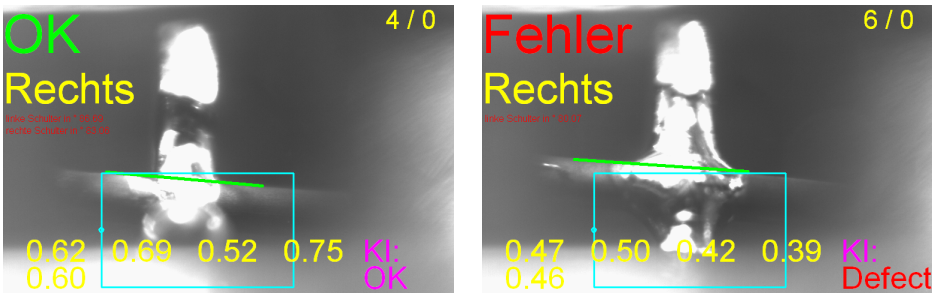


Figure 2.4: Examples of properly welded parts.



(a) False negative.

(b) False positive.

Figure 2.5: Images misclassified by the original approach (*Fehler* meaning *Defect*).

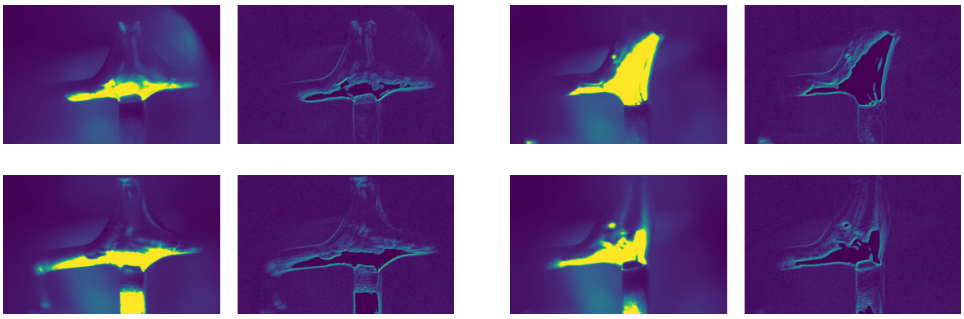


Figure 2.6: Edges of welding images generated with a sobel filter.

handle by the classical approach. Many reflections generated false edges in the derivative domain, making pattern matching on these features highly unstable (see figure 2.6). Additionally, the many ways the electrode was bent after processing introduced further variance into the system. Bad preconditions when relying on finite and hand-crafted detection patterns. Trust in the system was rightfully missing, necessitating a costly manual re-evaluation of each part through a visual control.

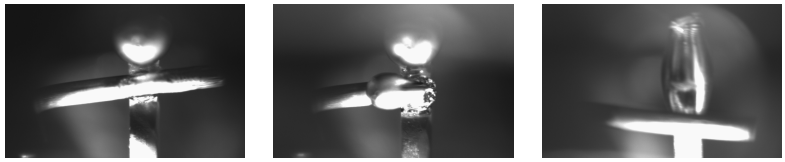
2.1.2 Introducing Machine Learning

To get rid of these problems, I replaced the existing classification with a state of the art artificial neural network (ANN). A 51 layer ResNET architecture was selected based on [He+16]. Apparent defects often where not confined to a specific region of the image, segmentation or marking regions of interest (RoIs) was not applicable. Thus, I chose a per-image classification approach.

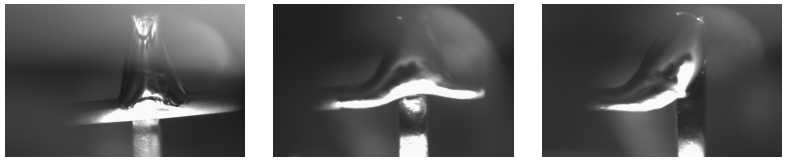
As is true for this and many other manufacturing projects: Domain experts typically possess significantly more knowledge than what is captured in requirements and process documentation [QWJ17]. Not all potential defect reasons where known or had adequate samples at the beginning of the project. This lead to the output being confined to only two classes: *OK* and *Defect*, effectively modeling it as a binary classification problem. Finer granularity of defects (see figure 2.7 on the following page) was only considered at a later point, but discarded due to the advanced state of the project.

Since a working imaging system was already in place, generating image data was no problem: Images could be taken directly from production. For the first proof of concept (PoC) 5000 pictures, taken over the course of three days, where labeled by a process engineer. A model trained on this data reached close to 100 percent validation accuracy and was tested in production,

although running in parallel to the existing solution.

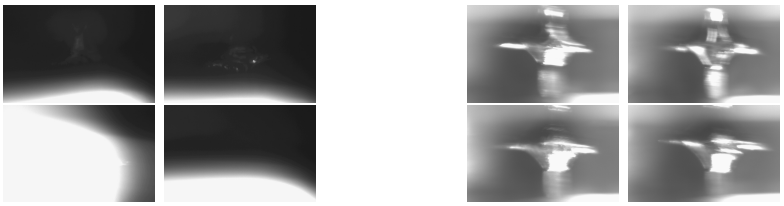


(a) Singular weld bead (b) Two weld beads (c) No connection



(d) Brittle electrode (e) Bent electrode (f) Duck beak

Figure 2.7: Various defects that were encountered.



(a) Lighting over-exposition (b) Blurry images

Figure 2.8: Images that are un-rateable.

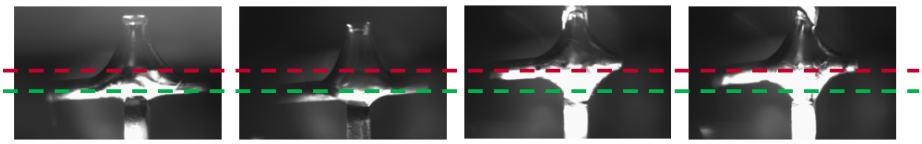


Figure 2.9: Changes in positioning over time.

But it soon became apparent that the initial data was not sufficient: Performance degraded quickly and critical defects where overlooked. This behavior was caused by several factors:

- The time period captured for labeling did not cover all possibilities for defects. For example, in the initial dataset one of the most prominent defects where so-called *weld beads* (see figure 2.7a). After a month of operation, images with two beads emerged (figure 2.7b), which were not classified correctly. This was also true for other defects shown in efigure 2.7.

- Mechanical influences caused the positioning inside the image to change after a certain point in time (see figure 2.9 on the preceding page). This degraded overall performance, while it was not apparent to operators how subjectively similar pictures received different classification scores. Although data augmentation was able to fix these issues, operator confidence in the new ML system took a sustained decrease.
- From time to time pictures would not be rateable due to problems during image capture (see figure 2.8 on the previous page). The model had a tendency to rate this as *Ok* while they should have been sorted out.
- The sub-optimal lighting conditions and unstable light source led to a high variance in samples between long time intervals. Since such changes were not observable on the initial images captured over a single week, it was mandatory to label additional data taken over the time span of multiple weeks.
- Changes in the perception of what is considered an *Ok* part changed over the course of the project. Thus images labeled as *Defect* had to be reclassified to *Ok* and vice versa.

2.2 Machine Learning for Manufacturing

These problems encountered on the weld inspection seem to be fundamental for industrial applications [Din+20]. To understand why the system failed in operations despite good evaluation performance, I will briefly examine the basic assumptions of machine learning and how well they apply to industrial data.

2.2.1 Basic Machine Learning Concepts

ML is an umbrella term for various methods to computationally solve problems. In contrast to the classical engineering approach the underlying algorithm or its parameters are not designed by hand but rather *learned* from data [Mit97].

Terms and Components

ML problems are usually defined by the following components:

- A training set $\mathbb{S}$ composed of various **input** samples x. These inputs encompass encoded information about the observable state of the problem that needs solving. They may represent an image, text, time-series values or any other form of encodable data.
- A **model** $f(x; \theta_f) : x \to y$ that maps observation x to an output y.
- The models **parameters** θ_f. These may be fixed for learned parameters or store data about the models current state (for sequence learning or multi-step problems).
- A measurement of **performance** $\mathcal{L}$ (f, $\mathbb{S}$, θ_f) (often called loss) that evaluates how well the system is solving the given task. It is calculated over all samples contained in $\mathbb{S}$.

To find suitable parameters for the model, its respective loss function is minimized over all instances of $\mathbb{S}$:

$$\theta_f = \arg\min_{\theta_f} \mathcal{L}(f, \mathbb{S}, \theta_f) \tag{2.1}$$

This formulation describes ML as a mathematical optimization problem [GBC16], allowing various tools of parameter optimization to be applied. Since the totality of $\mathbb{S}$ is often not available or too large to process, solving this equation is usually done on a subset of $\mathbb{S}^T \subset \mathbb{S}$, the so-called **training set**).

ML methods usually fall into one of three categories:

Supervised Learning

In **supervised learning** $\mathcal{L}$ usually measures the distance or dissimilarity between the model's output and a desired result (the so called ground truth) over each sample. An example for such a problem would be image classification, where x is the image and $\hat{y}$ denotes the type of object depicted.

The process of assigning ground truth (also called labels) for each sample is called **annotation** and is done by one or more (usually human) **domain experts**.

Unsupervised Learning

For problems where a ground truth can't be specified, $\mathcal{L}$ may be used to impose various meta-criteria on the result. These may include clustering of

similar samples, elimination of uninformative data (noise) or finding other forms of correlation. Instead of relying on additional labels, the optimization is based on the input samples alone.

Additionally, a plethora of in-between forms exist: In self-supervised learning, labels are generated in an unsupervised manner to assist in training a supervised system [Liu+21]. Semi-supervised learning considers both labeled and unlabeled data during training [VH20].

Reinforcement Learning

For the aforementioned methods, the training is decoupled from its environment. Data is gathered beforehand and compiled into a static training set. But when the system interacts with a complex and independent environment, **reinforcement learning** needs to be applied.

Here the model's output describes actions made on observations x inside an environment with unknown internal state s. Each action results in a changed state s' and a reward r, measuring the current performance of the model. $\mathcal{L}$ is usually heavily correlated with the inverse of r to incentivise actions maximizing overall reward.

An example for a task solved by this method would be an artificial intelligence (AI) inside a game: Reward is assigned depending on game score and x models the visible information available to the player.

2.2.2 The Extrapolation Problem

Supervised and unsupervised models are usually static: They search for suitable parameters during training, which are then fixed during **inference** (invoking the model $f(x;\theta_f)$ on data $x \notin \mathbb{S}^T$ not available during training). It is expected that a model performs well on any data drawn from the same distribution $\mathcal{S}$ as $\mathbb{S}^T$. This is based on the assumption of **generalization** done by the model:

$$\mathbb{S}^T \sim \mathcal{S} \land \mathbb{S} \sim \mathcal{S} \Rightarrow \mathcal{L}(\mathbb{S}^T) \approx \mathcal{L}(\mathbb{S}') \tag{2.2}$$

It implies the model's prediction is accurate for both samples seen during training and any other sample in $\mathbb{S}$ it may process afterwards, given that these sets have the same underlying data distribution. To verify this assumption an additional disjunct **evaluation set** $\mathbb{S}^E$ is used during **evaluation**.

$$\mathbb{S}^E \subset \{x : x \in \mathbb{S} \land x \notin \mathbb{S}^T\} \tag{2.3}$$

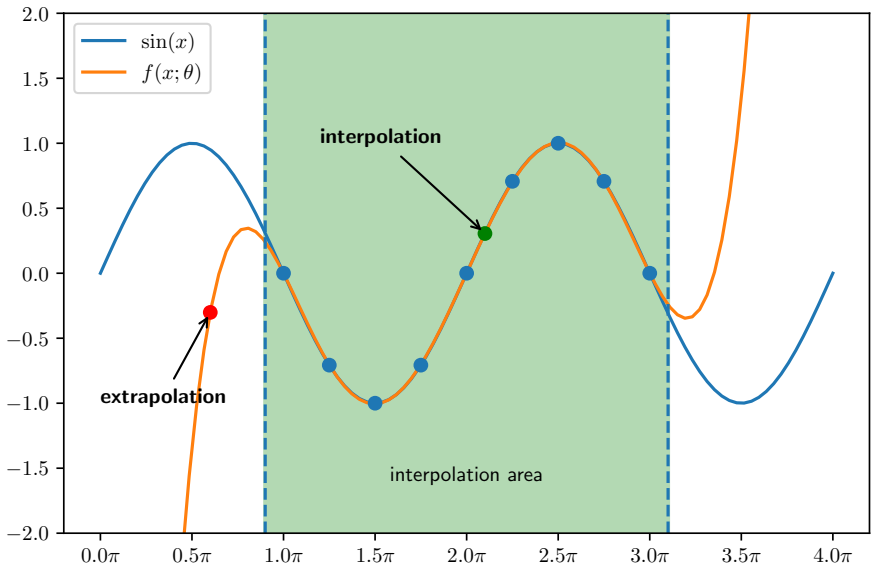


Figure 2.10: Extra- and interpolation area of an polynomial estimator $f(x;\theta)$ on a sine function. The green point is correctly inferred while the red point is far from the correct value.

If generalization holds and $\mathbb{S}^E$ follows the same data distribution as $\mathbb{S}^T$, the term describing the generalization error

$$|\mathcal{L}(\mathbb{S}^E) - \mathcal{L}(\mathbb{S}^T)| \tag{2.4}$$

should be minimal [XM12]. If not, the model is considered **over-fitting**, implying its parameters are too specifically fine-tuned towards the training data. In state-of-the-art ML models this issue is researched quite well [Li+19], providing many applicable solutions [Yin19].

But what if the samples encountered during live operations (denoted $\mathbb{S}^P$) are from a different data distribution (called out-of-distribution (OOD) samples [Yan+24])? Or their features are influenced by an unknown confounder, drifting towards another distribution (called data drift [Mor+12])? The assumption $\mathbb{S}^P \sim \mathcal{S}$ does no longer hold and no guarantee is given that the model is able to generalize to these new samples.

A sample that is not covered by the training distribution can fall into two areas: The area between known samples, which is usually adequately handled by the model by **interpolating**. Or in an area that is completely detached from any training data, forcing the model to **extrapolate** (basically

guessing).

Herein lies one fundamental misunderstanding between human and machine learning: As humans we base our extrapolation on a vast amount of additional experience, sourcing from information that is not directly derived from the problem [DBM97]. We are able to infer knowledge from similar problems we encountered in the past. The ML algorithm solely relies on the information contained in $\mathbb{S}^T$ as well as prior knowledge encoded in its design. The current drive towards foundation [Bom+21] models aims to address this issue by learning from a very broad range of information and then optimizing towards the specific task [Che+23].

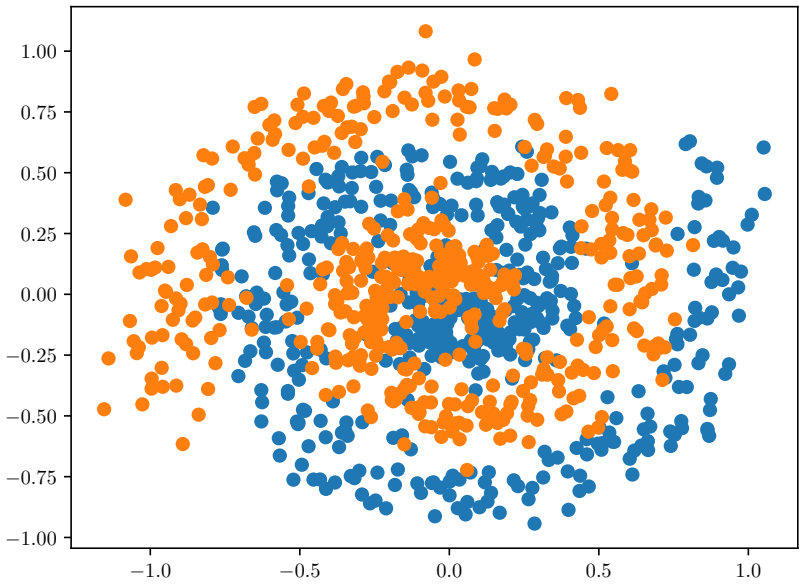
To illustrate the issue figure 2.10 on the preceding page shows the encoding of a sinus wave. For a human, the continuation is completely obvious: We have encountered this kind of function before and have a concept of infinite continuation. For the machine, the interpretation based on its training samples is completely valid. Not addressing this issue leads to mistrust in the system, as obviously wrong predictions are made by the model. Even worse, the labels assigned by the model are usually of high confidence, despite the output being a complete guess [CW17]. This is often attributed towards poor calibration [Guo+17], but neglects the problem of over-confident extrapolation visualized in figure 2.11 on the next page.

Recent research into trustworthy AI [Li+23] usually focuses on the interpolation area (*e.g.* [Cho+20a]), especially at the **class boundary**: it denominates a line in the feature space where a sample is assigned another class when crossing it. This is due to the discussion mainly evolving around adversarial attacks [Mad+18], where a malicious adversarial tries to fool the model by manipulating samples and pushing them over the class boundary.

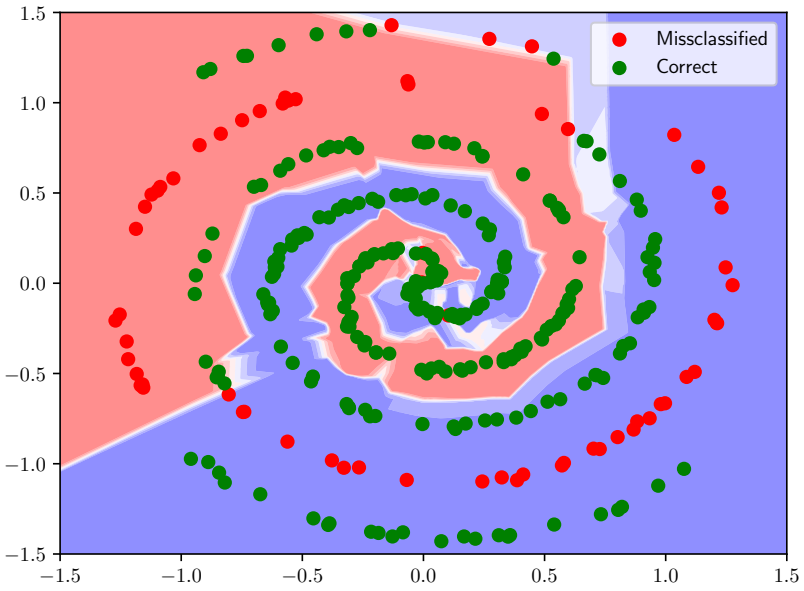
Figure 2.11 on the following page demonstrates why focusing on class boundary is not sufficient. The apparent miss-classification does not occur in the area between the two sample classes. It rather appears at the logical continuation of the spiral which is not covered by any data points at all. Recent research often completely disregards this problem [Gaw+23].

Issues with wrongful extrapolation has also become more apparent with generative large language models (LLMs) [Tho07] like BERT [KT19] or GPT-3 [Bro+20]. These may produce invented facts (termed 'hallucinations') which are not based on existing data [Ji+23]. Hallucinations are basically extrapolations without support of training data [Ton+24] where the model does not provide any implication that it is actively guessing.

In the visual domain, a widely accepted method to measure per-sample



(a) Original training samples.



(b) Evaluation with in and out of distribution samples.

Figure 2.11: Output of an ANN on training data sampled from two spirals. The model signals uncertainty only at the class boarder between sample groups (white areas) and misclassifies with high output values. Since the model has no concept of a spiral, the correct continuation can not be found outside the distribution of training samples.

reliability including extrapolation is missing [Tra+22]. This prevents visual ML models from usage in safety critical domains where miss-classifications are intolerable. Assigning unnoticed defects with a high confidence score undermines trust into the application. Thus a quality control is not operable if it can not provide an accurate estimation about its own reliability.

2.2.3 Challenges of Manufacturing Data

To make matters worse, there are certain aspects of manufacturing data that amplify these problems [Din+20]. During my research, it was interesting how many problems are overlooked or handled individually in academia, while they are obvious to almost anyone working in the manufacturing industry. This is due to the nature of datasets used in academic research, which are usually of very high quality. This high degree of data quality is essential for industrial ML applications to be successful [Ari+20]. In the following sections, I will explain where these challenges to quality stem from with respect to manufacturing data.

Sample Bias

Many applications of ML tackle rather broad problems: Speech-to-Text, natural language procession (NLP) or visual object classification can usually resort to huge datasets with millions and billions of samples. They are usually composed of data from various sources, thus covering most of the space over possible samples. To the contrary, manufacturing applications are very narrow in scope. Few datasets exists [Jou+21] and are usually highly specialized towards a specific use-case. Thus data is usually gathered for the specific task through a single source, imposing a selection bias [HHR04] on the dataset.

Additionally, most popular ML applications do not have to handle data changing over time. *E.g.* a face a hundred years ago looks quite similar to a face today, and probably will so for many years to come. Thus, a face detection algorithm is pretty certain to pertain its performance through time. Manufacturing processes and data may change over time:

- The equipment or setup deployed in production is not the same as the one used for data generation during training. This might be because the initial data was produced in a lab and lab conditions can not be reproduced. Or equipment was changed due to availability or secondary issues.

- The underlying production process has changed. Different pre-materials, preceding production steps or manual processes might influence the quality of the data captured.
- Measuring equipment degraded or broke due to strain produced by uninterrupted operation. There might also be environmental influences, like particles smudging a camera lens.

But data gathered to implement a project is usually gathered over the course of a few weeks. Thus a temporal selection bias is introduced. The data is very likely to not capture all deviations occurring during continued operations - which is usually expected to last multiple years.

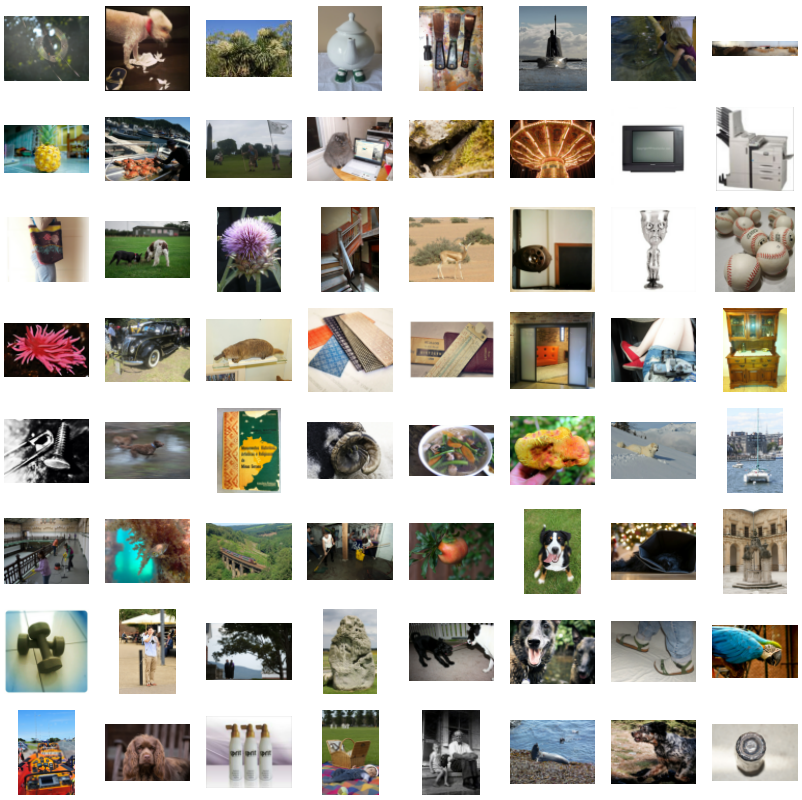
Some of these effects may be countered through data augmentation [SK19]. But is often not clear what augmentations are required to cover the whole variance present over the whole application life-cycle. Subsequently, the question *How much data is required?* is hard to solve if the amount of potential risk factors (*e.g.* possible product faults) is not fully known [Bla16].

Low Inter-Sample Variance

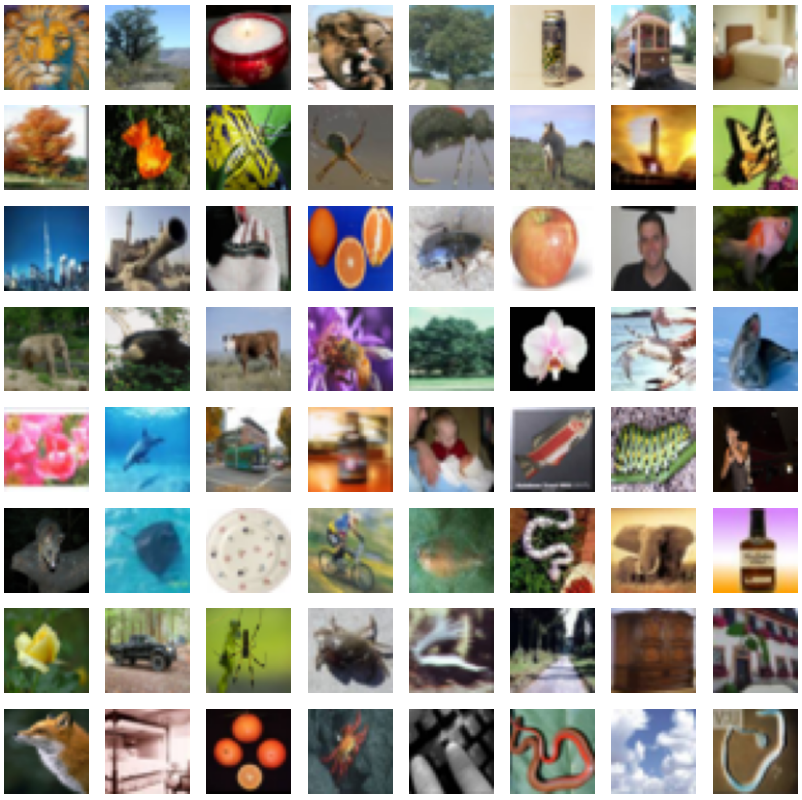
The problem of incorrect extrapolation is enhanced by the particular nature of manufacturing data: Data associated with most machine learning problems displays high inter-sample variance (see figure 2.12 on the next page). The set over all possible data $\mathbb{S}$ covers a broad space in features. On the contrary, industrial data is usually confined to a very narrow sample space, ideally with close to no difference between each sample. In fact, the goal of a manufacturing process is to be stable by containing as little variance as possible. This implies that small details are important to the assigned class while the overall image composition is negligible.

Many open set recognition methods try to train an OOD detector on artificial or collected task-irrelevant samples [GHC20a]. But for low-variance data, it is usually not possible to simply construct a set of OOD samples, since there is no data close enough to contain relevant differences [GL21]. It basically has to be produced by the same physical process, which is a contradiction in itself.

The only way to circumvent this fact is to generate artificial adversarial samples by either modifying captured samples through image manipulation, physically introducing abnormalities to the inspected objects or capturing images with faulty sensor configurations. But there is still no guarantee to cover the whole space of possible productive samples, since these methods



(a) Imagenet-v2 [Rec+19]



(b) CIFAR-100 [Kri09b]

Figure 2.12: Snapshots from popular image classification datasets.

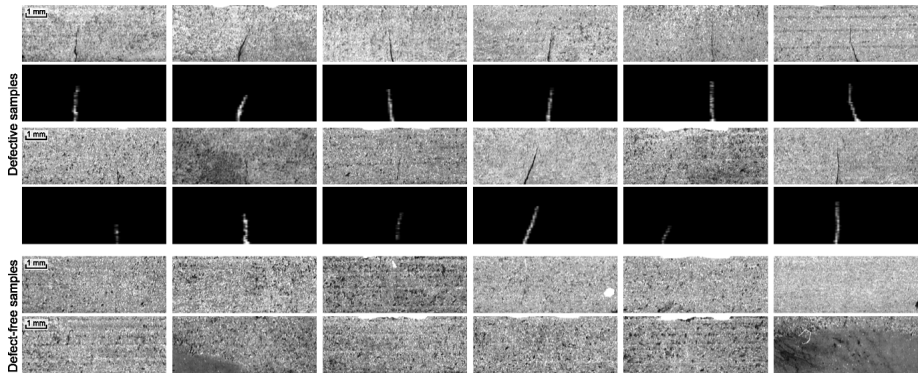


Figure 2.13: Segmentation to detect surface scratches [Tab+19].

introduce bias by themselves based on their inherent limitations: Image manipulation can not mimic every possible visual effects, faulty configurations like a smeared camera lens are often not reproducible and not all physical abnormalities can be created by design.

Class Imbalance

An additional challenge for creating an adequate machine learning model is the high imbalance in class labels present in manufacturing data. Defects that should be detected are usually less frequent than faultless parts to a high degree (often less than one in a thousand). The same is true for segmentation problems, where the significant parts are usually confined to small areas in the image (see figure 2.13).

This imposes multiple challenges: Data has to be cleverly selected from the pool of available samples. Model training must be aligned by oversampling [Yap+14] or weighting the loss function [Cui+19].

Class imbalance also increases the influence of the extrapolation problem: Since samples from defect classes are scattered more sparsely over the feature space, the model is likely to output the class with higher variance when guessing. Which is the *no defect* class. This is especially problematic for manufacturing quality control, where a rejected good part (called **over-reject** or **false positive**) is less problematic than an undetected fault (**false negative**). This kind of miss-classification cost is seldom considered in research [Lan+10].

2.3 System Requirements

All these factors necessitate certain capabilities in a ML system that the bare bone implementations do not provide:

Verifiable: The system needs to provide information about its current state. There must be a clear indication on whether the model is still capable to accurately classify the samples provided.

Repairable: An operator must be able to fix the issues on the machine, given they are not on the models side. The system must give clear indications what the desired configuration of the measurement equipment is and how the measurement equipment must be setup to provide correct classification results.

Optimizable: There must be a process in place that occurs when the outside factors have changed in a way that the system is no longer repairable. When the nature of data changes significantly, there must be an efficient and stable way to improve the model and restore its classification capabilities.

Explainable: If miss-classifications occur, it must be transparent to the operator why this is the case. Modern ML models are of a black-box nature (the model itself is only understandable by its in- and outputs, since its inner structure is too complex). While this fact can not be mitigated, there should be means to visualize the assumptions the model based its decisions on.

In the upcoming chapters I will present available solutions to these requirements. Furthermore, I will introduce the VMLC system, which addresses all these challenges in one unified solution.

Chapter 3

Technical Background

> Those who are in love with practice without knowledge are like the sailor who gets into a ship without rudder or compass and who never can be certain whether he is going.
>
> *Leonardo da Vinci*

This chapter will give a brief introduction into the methods and technology available to tackle the aforementioned problems. At first, I describe modern machine learning (ML) using artificial neural networks (ANNs), which were used throughout this thesis. Afterwards, I give an overview over monitoring techniques available to measure system confidence, drift and stability and outline the topic of machine learning operations (MLOps). Subsequently, topics directly relevant to industrial imaging are explored, like sensor calibration using deep learning (DL) and visual statistical process control (SPC).

3.1 Artificial Neural Networks

ANNs are currently the prevalent method to implement machine learning. While being a staple of the ML spectrum since pretty much its beginning, they received increased traction in recent years. This was due to the vast increment in computing power [Sch15] as well as novel methodologies like Batch Normalization [IS15], rectified linear units (ReLUs) [NH10] and convolutional layers [KSH12]. Their scalability towards arbitrary sizes of data dimensionality, complexity and quantity makes them usable for almost any

ML task. Accordingly, any model used in this thesis is based on ANNs, if not specified otherwise.

3.1.1 Design

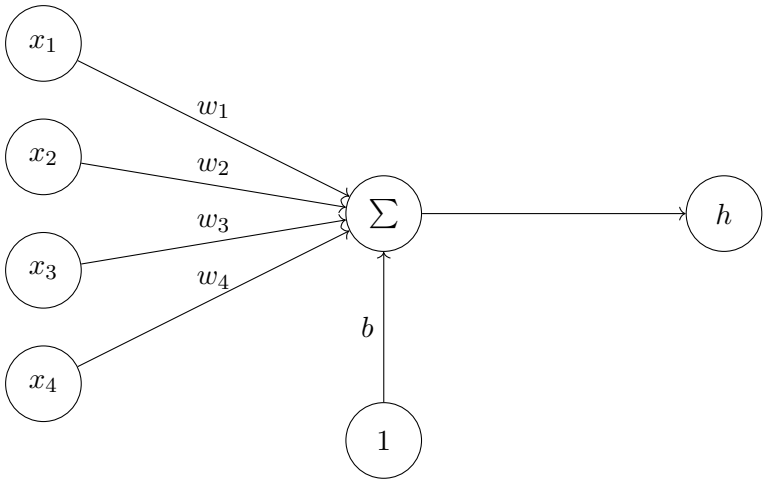


Figure 3.1: Single perceptron cell.

The initial design of ANNs was the attempt to reproduce the behavior of interconnected neural cells. Their first formulation was introduced way back in 1960 as the multi layer perceptrons (MLPs)[Ros58]. A single cell, as depicted in figure 3.1, is defined as:

$$f_p(x; w, b; \theta_{f_p}) \coloneqq h(w \cdot x + b) \tag{3.1}$$

where the **weight** w and **bias** b are parameters of the cell. h describes the (usually non-linear) activation function. It models the threshold required by a nerve cell to actually *fire*. In the original perceptron, the Heaviside step function was used, which becomes 1 if $x > 0$ and -1 if not.

By specifying x as a vector of length N the definition can be extended to describe a layer of interconnected perceptrons, often called a **fully connected** or **dense layer** layer:

$$f_{fl}(x; \mathbf{W}) \coloneqq h(\mathbf{W} \cdot [x, 1]) \tag{3.2}$$

The weight matrix $\mathbf{W} \in \mathbb{R}^M, N$ contains the weights and biases of each cell, where M determines the size of the output. A constant value of 1 is appended to include the bias as an offset independent from the input.

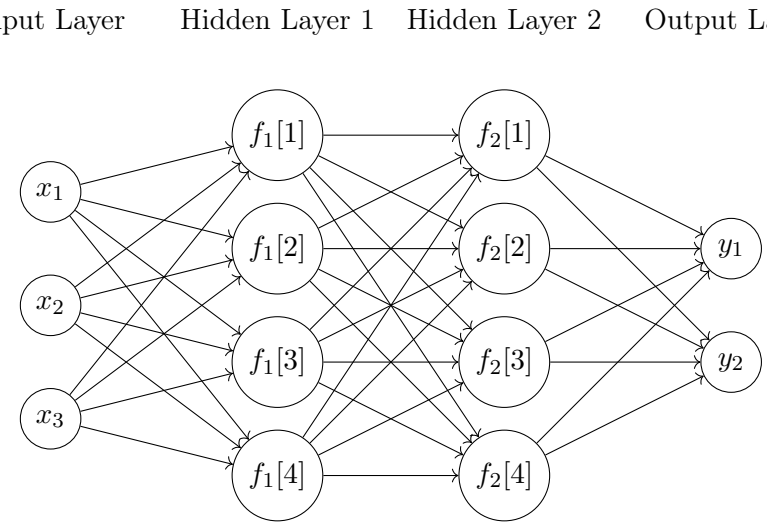


Figure 3.2: Fully connected feed-forward network with 2 hidden layers.

To improve the complexity that a model comprised of perceptrons can represent, multiple of these layers are stacked on top of each other. Each uses the output of the preceding layer as its input:

$$f(x;\theta_f) \coloneqq \{f_0(x) \coloneqq x\} \cup \{l \in 1 \dots n : f_l(x) \coloneqq f_l(f_{l-1}(x);\theta_{f_l})\} \tag{3.3}$$

Since every layer only depends on the preceding layers, this is called a **feed-forward network**, illustrated in figure 3.2. The first layer f_0 only consists of the input x, called the **input layer**. The last layer f_n is the systems **output layer**. All layers in between are usually coined **hidden layers**.

3.1.2 Training

In equation (2.1) on page 12 I introduced machine learning as an optimization problem solved over the training data $\mathbb{S}^T$. This is also true for ANNs, which are usually (but not exclusively) employed in a supervised manner.

To reiterate, ML aims to find ideal parameters based on a loss function $\mathcal{L}$:

$$\theta_f = \arg\min_{\theta_f} \mathcal{L}(f, \mathbb{S}^T, \theta_f) \tag{3.4}$$

ANNs usually calculates this function by accumulating over the error E

between its output and a provided ground truth label $\hat{y}$:

$$\mathcal{L}(f, \mathbb{S}, \theta_f) \coloneqq \sum_{x,\hat{y} \in \mathbb{S}^T} \mathrm{E}(\hat{y}, f(x; \theta_f)) \tag{3.5}$$

$\hat{y}$ is the ground truth, resembling a desired output or label for each sample. E is an error function that aims to measure the dissimilarity between the model's output $f(x)$ and the respective label $\hat{y}$ of x.

This means that ANNs fall into the category of supervised learning by definition. Through some modifications neural networks are also capable of unsupervised (see Auto-Encoders (AEs) explained in section 3.4.3 on page 45) or reinforcement learning [Aru+17].

The definition for $\mathcal{L}$ may be expanded with additional criteria, for example to improve generalization. This is usually done so by a regularization term that enforces a certain distribution over the weights [GJP95] or outputs of the network [KW14].

Error Functions

To specify the error function E a multitude of options exist. Of those, mean squared error (MSE) is one of the most common, used both in statistics [HLS13] and ML. It is defined as:

$$\mathrm{E}_{MSE}(x, \hat{y}) \coloneqq \frac{1}{n} \sum_{i=1}^{n} (f(x)_i - \hat{y}_i)^2 \tag{3.6}$$

It is usually sufficient for most problems, but does not work well in classification scenarios since each value in y is evaluated independently.

When $\hat{y}$ is a binary vector with $\forall i : x_i \in \{0, 1\}$, **cross entropy** may be applied to calculate $\mathcal{L}$. $f(x)$ should model a probability estimation, with $\sum_i x_i = 1 \land \forall i : 0 \leq x_i \leq 1$. For C classes cross entropy loss is defined as:

$$\mathrm{E}_{CE}(x, \hat{y}) \coloneqq -\sum_{c=1}^{C} \delta_{\hat{y}_c=1} \log(f(x)_c) \tag{3.7}$$

where $\delta_{\hat{y}_c=1}$ is 1 if c is a label contained in the ground truth and 0 otherwise.

Based on the task at hand further error functions exist: One may add the frequency of classes as weight to each sample to handle class imbalance [Yap+14] or represent other criteria like intersection over union (IoU) for image segmentation tasks [Zha+22].

From these I want to highlight the structural similarity index (SSIM) [Wan+04] as an improved method to measure error between two images x and y. Unlike MSE, where error is calculated over the whole images, multiple sub-images with various sizes (regions of interest (RoIs)) are taken. Total error is defined as the average SSIM value over these regions:

$$\mathrm{E}_{SSIM}(x, y) \coloneqq \frac{1}{N} \sum_{r \in ROIs} SSIM(ROI(x; r), ROI(y; r)) \tag{3.8}$$

Afterwards, the SSIM value is calculated by:

$$SSIM(x, y) \coloneqq \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.9}$$

Here μ_i is the mean of pixel values and σ_i the variance over the region. $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ are variables to stabilize the division, with L being the max pixel value. The original paper suggests $k_1 = 0.01$ and $k_2 = 0.03$, but higher values up to 0.4 for k_2 achieve better results according to [Doc]. To use SSIM as a loss function, the actual structural dissimilarity is calculated:

$$\mathrm{E}_{SSIM}(y, \hat{y}) \coloneqq \frac{1 - SSIM(y, \hat{y})}{2} \tag{3.10}$$

Gradient Descent

To solve the optimization problem an adequate optimizer is required. A huge benefit of ANNs is that they are completely differentiable. Thus training is usually conducted by using gradient descent combined with back-propagation [Ama93]. The method has become the dominant way of training deep neural networks due to its broad applicability and performance [Sch15].

In gradient descent the global minimum of $\mathcal{L}$ is searched for along the gradient over the parameter space θ. $\mathcal{L}$ is evaluated on the current set of θ over the full sample set $\mathbb{S}^T$. Afterwards, the gradient $\frac{\partial \mathcal{L}}{\partial \theta_i}$ is calculated for each parameter $\theta_i \in \theta_f$. When all parts of $f(x; \theta_f)$ are differentiable this term can be evaluated for any parameter by applying chain rule.

Afterwards, back-propagation is done by shifting all parameters along their respective gradient:

$$\theta_i \leftarrow \theta_i - \eta \frac{\partial \mathcal{L}}{\partial \theta_i} \tag{3.11}$$

where the learn rate η is a scalar value $0 < \eta \leq 1$. It is required to make the algorithm more sensitive towards small changes in E. Updates on θ are

repeated until $\mathcal{L}$ converges. The value of η determines the rate of convergence. Usually, a higher η is used in the beginning to find the general minimum, which is reduced after some iterations to capture smaller deviations in E [GBC16].

Original gradient descent optimizes over all instances of $\mathbb{S}$ at the same time, which is usually not possible due to memory constraints. Instead, a small extension called stochastic gradient descent (SGD) is introduced: $\mathbb{S}^T$ is sub-sampled into n smaller **mini-batches** $\mathcal{B}$ with $\bigcup_i^n \mathcal{B}_i = \mathbb{S}^T$. Weights are updated on each mini-batch instead of the whole dataset. Training time is often measured in **epochs**: An epoch has passed when each sample in $\mathbb{S}^T$ has been processed at least once.

Nowadays, a lot of extensions to SGD exist: They either add momentum to the gradient update (including past gradient updates with a small factor) [HSS12], estimate lower-order moments for updates [Kin14] or dynamically change the learning rate during training [Zei12].

Alternatives

Despite SGD being dominant for network training, other approaches exist: Basically any optimization method on a fixed number of correlated real values may be used (*e.g.* non-linear conjugate gradient [DY99], Levenberg-Marquardt [Mar63] or L-BFGS [LN89]). These include methods that do not require a gradient: An evolution strategy (ES) was used in [Sal+17], creating multiple instances of a slightly altered network and improving parameters based on the best performing ones. Through implementation of an evolutionary algorithm (EA) the optimum might even be searched globally[Din+13]. Extensions even include and alter the topology of the network in its parameters [Mii+24]. These methods usually require an immense amount of computational power since every set of parameters requires a full evaluation on the training data.

3.1.3 Advancements

The great advantage of gradient descent is its applicability to any differentiable function. Initial networks where ratter shallow, containing only few hidden layers. In recent years, deep learning [Sch15] has emerged as a universal solution to many ML problems, sometimes employing far over hundred layers. To enable this, in addition to better computing through the use of graphical processing units (GPUs), many improvements had to be added to the original concept through new or modified layers. In this section, I will list the most

important ones relevant to this work.

Activation Function

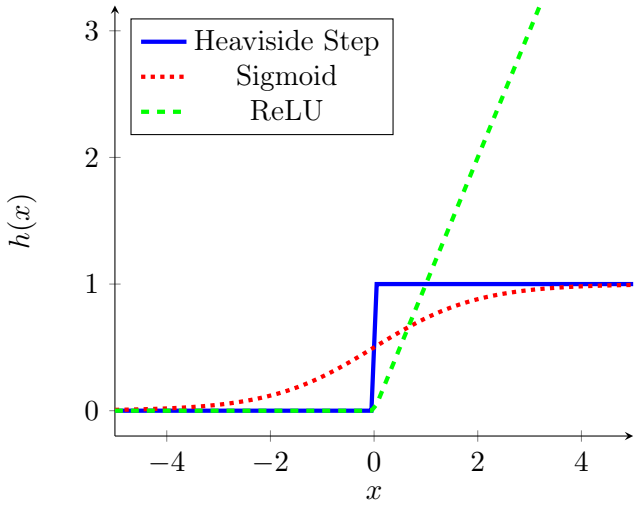


Figure 3.3: A comparison of activation functions.

The layer defined in equation (3.2) on page 23 contains an activation function h to introduce an element of non-linearity to the model. It usually does not contain parameters and represents a threshold. Traditionally, the **sigmoid layer** was used as a differentiable version of the binary Heaviside step-function:

$$h_{SIG}(x) \coloneqq \frac{1}{1 + e^{-x}} \tag{3.12}$$

A multi-dimensional extension is the **softmax** function, which has the added benefit that its output sums to 1, which is a requirement for cross entropy described in equation (3.7) on page 25:

$$h_{Softmax}(x)[i] \coloneqq \frac{e^{x_i}}{\sum_{j=1} e^{x_j}} \tag{3.13}$$

Many characteristics of these functions, especially the mapping on a value range of 0 to 1, seemed desirable at first. They have since been proven problematic through the so called **vanishing gradient problem**: When performing gradient updates on very high or low values of x, the gradient itself becomes so small that it might lead to no update at all.

Thus, a sigmoidal function is usually only used in the last **output layer**

of the network. For intermediate layers, ReLUs are used. They are defined as:

$$h_{ReLU}(x) \coloneqq \max(x, 0) \tag{3.14}$$

Despite theoretical arguments against its use (it is not differentiable at value 0, which can be ignored in practice) it proves dominant in most ANN architectures [NH10]. Many extensions to ReLU exist: Leaky ReLU [MHN+13] returns $a \cdot x$ when $x < 0$, where a is a small constant. Parametric ReLU (PReLU)[He+15] extends on this by making a a trainable parameter.

Batch Normalization

Until the emergence of the term Deep Learning, ANNs where considered unsuitable to solve more complex problems. This was due to the problem that after a certain amount of layers, training began to become unstable. This was mainly credited towards the **exploding gradient problem**. High gradients multiplied throughout the layers and influenced the weights of the up-most parts of the network too much to converge towards a meaningful solution.

Many solutions to mitigate this problem can be applied: For example pre-training the network layer-wise with stacked auto-encoders [Ben+06] or clipping the gradient to a certain maximum value [BSF94].

The most successful method proved to be **batch normalization** [IS15]. By normalizing the distribution of weights inside a layer to $\mathcal{N}(0, 1)$, extreme values where mitigated. A batch normalization layer over input x is defined as:

$$\begin{aligned} norm(x) &\coloneqq \frac{x - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \\ f_{BN}(x; \gamma, \beta) &\coloneqq \gamma \cdot norm(x) + \beta \end{aligned} \tag{3.15}$$

where $\mu_{\mathcal{B}}$ is the mean and $\sigma_{\mathcal{B}}$ the variance of the current mini-batch during training, which is replaced by an average over the last n batches during inference. $\{\gamma, \beta\} \hat{=} \theta_{BN}$ are learnable parameters.

Convolutional Layers

Another problem for deep networks is the vast amount of parameters. Since each fully connected layer $f(x)$ with input $x \in \mathbb{R}^N$ and output $f(x) \in \mathbb{R}^N$

requires $(M+1) \cdot N$ parameters, large data like images must be scaled down significantly, leading to omission of important features.

A solution to this problem offer **convolutional layers** [LeC+89]: They implement the discrete convolution operation

$$\mathbf{M}[n] * \mathbf{K} \coloneqq \sum_{k}^{K} \mathbf{K}[i] \cdot \mathbf{M}[n-l] \tag{3.16}$$

on the input $x \in \mathbb{R}^N$:

$$f_{Conv}(x; \mathbf{K}) \coloneqq x * \mathbf{K} \tag{3.17}$$

$\mathbf{K}$ is a convolution **kernel** $\in \mathbb{R}^M$. It is part of the trainable parameters of the network. The output dimensions of f_{Conv} are the same as its input, given that undefined values $(n-i < 0 \vee n-i > N)$ get provided through **padding**. Making the output size independent of its weights greatly reduces memory consumption, since parameters of $\mathbf{K}$ are shared across the whole input.

An additional benefit of convolution is that it can be represented as a singular multiplication when transformed into a fourier series [DJ12]:

$$x * \mathbf{K} = DFT(x) \cdot DFT(\mathbf{K}) \tag{3.18}$$

DFT is the **discrete fourier transform**. This decreases computational complexity from $O(M \cdot N)$ to $O(N) + O(DFT)$. Since $O(DFT)$ can be reduced to $O(N \cdot \log N)$ by applying fast fourier transform (FFT)[CT65] it makes convolutional layers both fast and computationally inexpensive.

Another motivation for using convolution is its broad application in signal processing: They can be used as band filters or detectors on time-variant signals [DJ12]. When extended to the 2-dimensional space, they may be used as edge detectors, image smoothing or even template matching through auto-correlation [Cas96; Hei92]. These operations are a useful part of every computer vision (CV) toolbox and mainly used for feature extraction [Umb05]. By allowing the network to select its own kernel parameters, it can reproduce any possible linear filter.

To allow for a multitude of filters at the same time the kernel is extended to a higher rank tensor $\in \mathbb{R}^{N,K,L}$ (N representing the dimensions of input x)[LeC+89]:

$$f_{Conv}(x; \mathbf{K})[l] \coloneqq \sum_{k}^{K} x[k] * \mathbf{K}[k, l] \tag{3.19}$$

This produces L outputs to be supplied to preceding convolutional layers.

Up- and Down-sampling

To work on different scales of input data and reduce memory consumption during training, it is advisable to scale down the input inside the network. One of the earliest example is the LeNet [LeC+89], where each feature map is halved in its dimensions after each convolution (see figure 3.4). The scaling is done by **max pooling**:

$$f_{maxPool}(x)[m,n] \coloneqq \max(\{x[sm \dots sm+s-1, sn \dots sn+s-1]\}) \quad (3.20)$$

Over every window of dimension $s \times s$ only the maximum value is retained. The value of s is usually set to 2 [GBC16]. Instead of using the max function, the average may be used with the added benefit that gradients are back propagated for all values, not just the maximum [GK20].

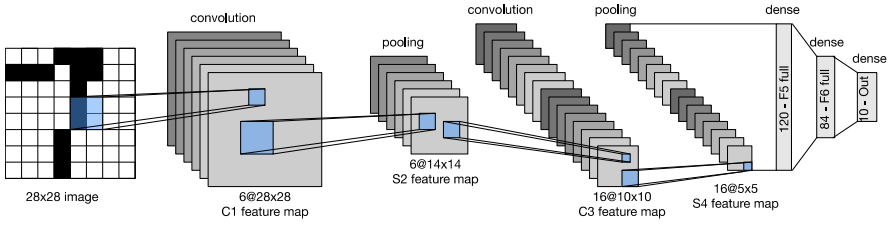


Figure 3.4: LeNet-5 architecture as depicted in [Zha+23].

For some use-cases the dimensionality of data has to be increased, for example when increasing the resolution of images for Super Resolution (SR) [Don+14]. This can be done by simply copying all values into tiles of size sxs:

$$f_{supSampling}(x)[m,n] \coloneqq x[\left\lfloor \frac{m}{s} \right\rfloor, \left\lfloor \frac{n}{s} \right\rfloor] \quad (3.21)$$

Residual Networks

When creating deep ANNs with large number of layers, model complexity might be too high to properly reflect the task at hand. Training might be slow or not work at all since the output's influence on the input gradient diminishes while propagating through all the layers. To mitigate this risk, skip connections or residual layers are introduced. They simply add the

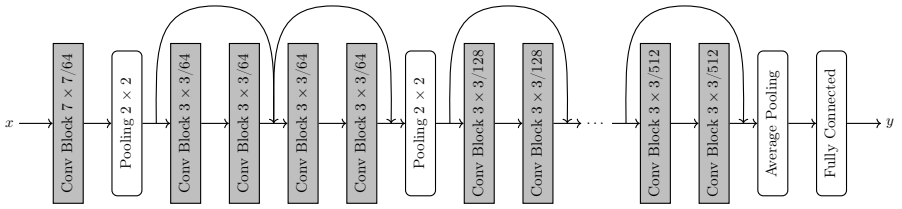


Figure 3.5: Stacked residual blocks inside a full Residual Network (ResNet) architecture.

output of a layer further up in the model to the current layer:

$$f_{Res_n}(x) \coloneqq f_l(x) + f_{l-n}(x) \tag{3.22}$$

Thus the original gradient is also propagated to this layer, while the skipped layer functions as residual block that may improve the overall performance.

For modern image classification networks, the residual architecture has been proven to be superior [He+16]. It utilizes residual **skip connections** to use a very deep architecture while allowing the gradient to propagate to higher layers, mitigating the problem of previous architectures where loss was not properly received by the up-most layers.

Figure 3.5 illustrates a typical ResNet architecture. Its first layer consists of a broader convolutional layer (with kernel sizes usually around 7×7), producing multiple feature maps. These are then scaled down through a max pooling layer.

The inner layers of the ResNet are made up of residual blocks, as illustrated in figure 3.6 on the next page. Each block contains several (usually two) inner blocks of batch normalization, ReLU and convolutional layers, encapsuled by a residual layer.

After a number of successive blocks, the data is scaled down again and fed into the next block. This is repeated for several scales, based on the number of layers contained in the network (typically ranging from 34 to over 150 layers). The last layer is a full pooling over the first two dimensions, leaving only the feature dimension containing one scalar value for each feature map. These are transformed into the output vector y by a fully connected layer, containing an activation function based on the desired properties of y.

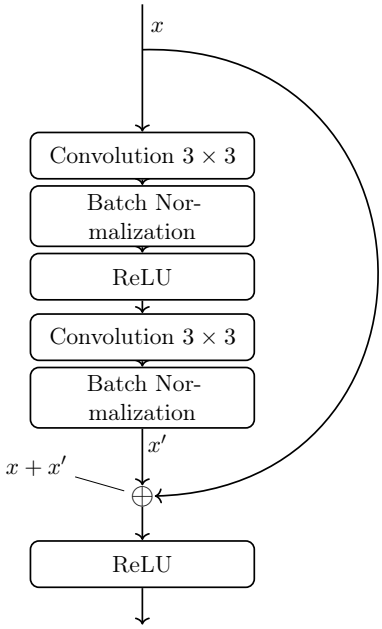


Figure 3.6: Main residual building block inside the ResNet architecture.

3.2 Bayesian Optimization

Sometimes, one may encounter problems where a gradient is not available. For example when a model tries to optimize a physical system where it has no access to its internal parameters. In recent years, Bayesian optimization has been successfully applied to such parameterization problems in the manufacturing domain [BKS23]. It is typically used where sampling is costly and no functional form is assumed [Rub+19].

Algorithm 1: Basic Bayesian optimization.

```
/* Observe n_0 initial points from the parameter space.
   */
ℙ ← {i ∈ n_0 : θ_i → f(θ_i)}
while |ℙ| ≤ N do
    𝒫 ← P(f(θ)|ℙ)
    /* Find the next point to sample.                    */
    θ_n ← arg max_θ acqu(𝒫, θ)
    /* Observe f at point θ_n.                           */
    ℙ ← ℙ ∪ {θ_n → f(θ_n)}
end
return arg max(ℙ)
```

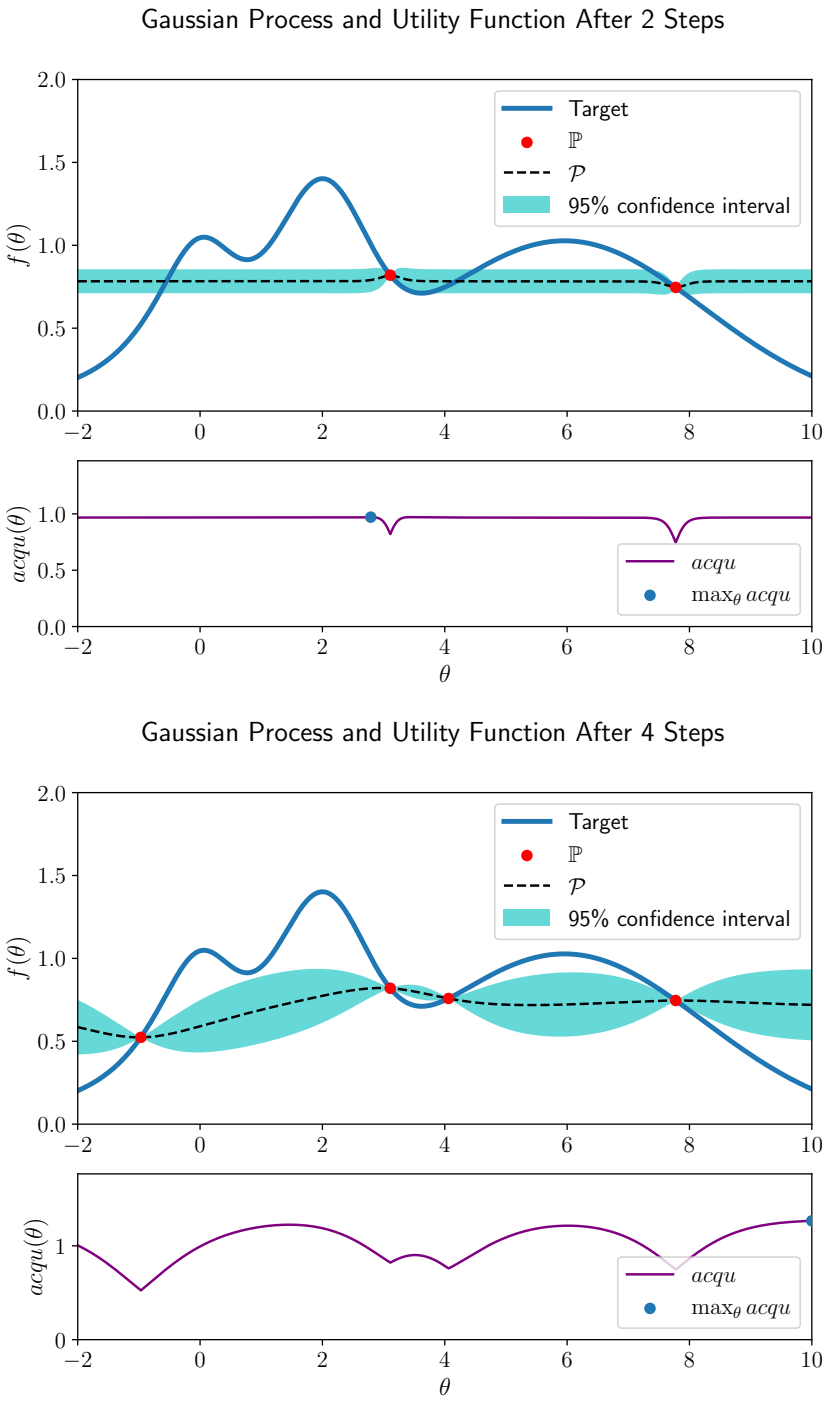


Figure 3.7: Example of Bayesian optimization on a 1-dimensional function $f(\theta)$ with the upper confidence bound (UCB) acquisition function.

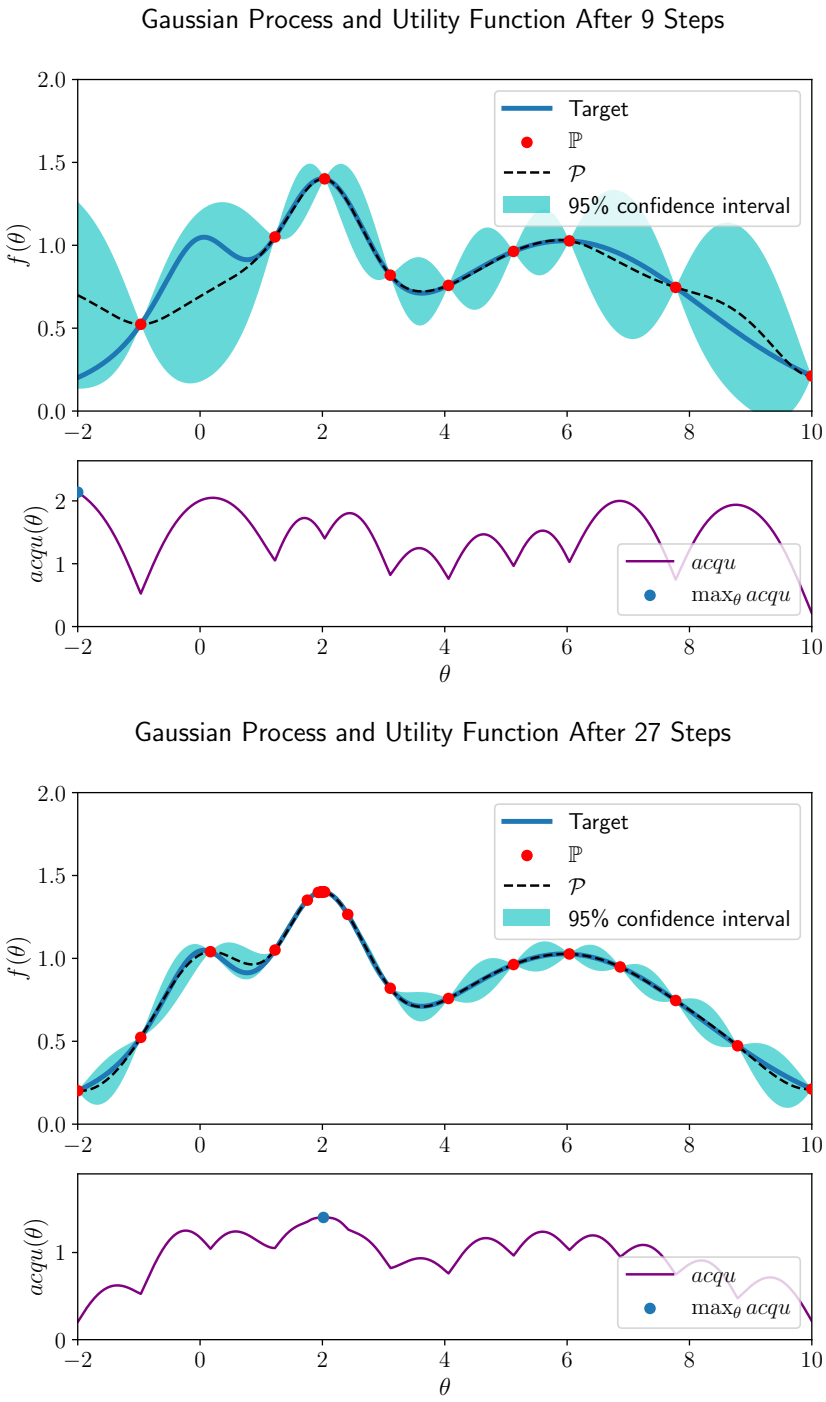


Figure 3.7: Example of Bayesian optimization on a 1-dimensional function $f(\theta)$ with the UCB acquisition function (cont.).

Algorithm 1 describes basic Bayesian optimization [Fra18]. The system to be optimized is denoted as f with its accessible parameters contained in θ_n . Sampling for a specific configuration is done through the operation $f(\theta_n)$. It optimizes f over the space of possible parameters Θ. $\mathcal{P}$ is modeled as a Gaussian process that denotes the posterior distribution over the observed points $\mathbb{P}$, usually consisting of a continuous normal distribution $\mathcal{N}(\mu, \sigma^2)$ over its mean $\mu(\theta)$ and variance $\sigma^2(\theta)$.

It is calculated using a mean function $\mu_0(\theta)$ and the kernel function $\Sigma_0(\theta_i, \theta_j)$, where close-by points should have higher positive correlation than those far apart, encoding the believe of adjacent parameters providing similar results:

$$\begin{aligned} \mathcal{P}(\theta) &\coloneqq P(f(\theta)|\mathbb{P}) \sim \mathcal{N}(\mu(\theta), \sigma^2(\theta)) \\ \mu(\theta) &\coloneqq \Sigma_0(\theta, \mathbb{P})\Sigma_0(\mathbb{P}, \mathbb{P})^{-1}(f(\mathbb{P}) - \mu_0(\theta)) + \mu_0(\theta) \\ \sigma^2(\theta) &\coloneqq \Sigma_0(\theta, \theta) - \Sigma_0(\theta, \mathbb{P})\Sigma_0(\mathbb{P}, \mathbb{P})^{-1}\Sigma_0(\mathbb{P}, \theta) \end{aligned} \tag{3.23}$$

where $\mathbb{P}$ contains the already observed parameters (and $f(\mathbb{P})$ its evaluated values) in vector representation.

Σ_0 is usually the exponential or Gaussian kernel:

$$\Sigma_0(x, x') \coloneqq \alpha_0 \exp(-\|x - x'\|^2) \tag{3.24}$$

with the kernel parameter vector α_0 determining the assumed amount of change that can occur inside f. Other, more sophisticated kernels like the Màtern kernel [RW06] are possible. The mean is usually set to a constant value $\mu_0(\theta) = \mu$.

The acquisition function *acqu* determines how the next configuration is chosen for evaluation. It has to handle the trade-off between exploration (sampling in areas that are unexplored and contain few samples) and exploitation (sampling in areas that are most probable to contain the maximum). Figure 3.7 on page 34 shows a visualization of the Bayesian optimization and its acquisition function.

There are multiple options to select *acqu* from:

UCB: The weighted sum $\mu(\theta) + \lambda\sigma(\theta)$ between the highest mean (exploitation) and the highest variance (exploration).

Probability of improvement (PoI): The probability $P[\mathcal{P}(\theta) > f(\theta^*)]$ that evaluating θ improves the value over the current best $f(\theta^*)$, which

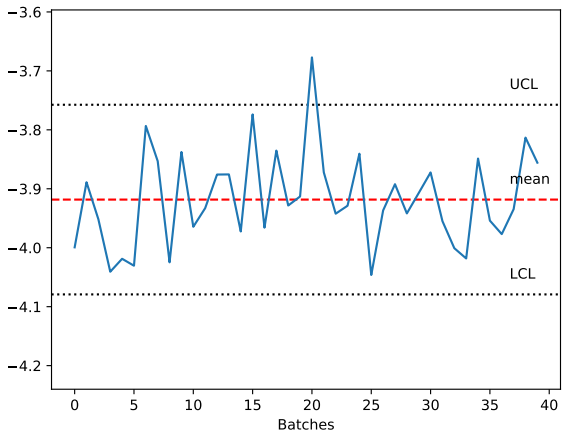


Figure 3.8: An exemplary SPC chart.

can be determined using the complementary cumulative distribution function (CDF).

Expected improvement (EI): The expected improvement $E[\max(f(\theta^*) - \mathcal{P}(\theta), 0)]$ over the current best $f(\theta^*)$.

Other functions like Knowledge Gradient [WF16] or Entropy Search [WJ17] exist, but usually contain more sophisticated components to approximate the underlying function f.

3.3 SPC and Machine Learning

In manufacturing, monitoring of processes is usually done through SPC [Dem52]. A process usually specifies the whole interaction between product, machines and operators. A process is considered **in-control** when measurements conducted during production fall into the expected distribution of values.

Figure figure 3.8 displays an **SPC chart**. A sensory output is averaged over multiple measurements (batches). It is restricted by an upper and lower **control limit** (UCL and LCL). When these limits are breached, a **control action** must be taken by an operator or process engineer to determine what caused the deviation and potentially fix it.

The control limits are calculated as:

$$\begin{aligned} \mu &\coloneqq \frac{1}{N}\sum_{i=1}^{N} m_i \\ \mathcal{B} &\coloneqq \{j \in \{1 \dots n\} : \frac{1}{N_s}\sum_{i=1}^{N_s} m_{i+j\cdot N_s}\} \\ \sigma^2 &\coloneqq \frac{1}{n}\sum_{\mu_\mathcal{B}\in\mathcal{B}} (\mu_\mathcal{B} - \mu)^2 \\ UCL, LCL &\coloneqq \mu \pm \sigma \cdot l_\sigma \end{aligned} \tag{3.25}$$

over a series of measurements m grouped into n batches of size N_s. By the application of the central limit theorem (CLT) the distribution over $\mathcal{B}$ is the normal distribution $\mathcal{N}(\mu, \sigma)$ [KK17].

It is apparent that the actual limits are dependent on the value of N_s. It is usually specified by a process engineer based on the application and is a compromise between time to reaction (low N_s, few samples have passed until a discrepancy triggers a control limit) and specificity of the control (high N_s, the increased flattening suppresses false positives caused by singular outliers).

l_σ is similarly chosen by the process engineer. It is usually called the **sigma value** and is a simple mean to control the sensitivity of the control limits. It is usually in the range of 3 to 6 σ.

SPC is usually applied to single or low dimensional physical measurements like electrical current or metric distance. When handling the huge dimensionality of images a historical gold standard to compress their information into a low dimensional metric does not exist [MWC11]. Managing the high complexity of measurements that consist of visual data in terms of SPC represents a huge research gap [Tra+22]. Newer research suggests methods for monitoring image data based on regions of interest [OSS21] or generating control charts for 3D point clouds [SC19], but fall short of general applicability.

3.4 Drift and OOD Detection

In section 2.2.2 on page 13 I introduced the problem of verifying that data observed during production is drawn from the same distibution $\mathcal{S}$ as the training data $\mathbb{S}^T$ as a fundamental challenge to any industrial applications of ML. Now I will introduce different approaches towards measuring this dissimilarity in the context of deep learning for computer vision.

To illustrate the issue, the set of all observable samples $\mathbb{S}$ is seen as a random distributions over all possible feature vectors, with each sample being a singular observable instance x drawn from $\mathcal{S}$ (denoted as $x \sim \mathcal{S}$). If all samples in a subset $\mathbb{S}' \subset \mathbb{S}$ are drawn randomly from $\mathbb{S}$ (denoted as $\mathbb{S}' \sim \mathbb{S}$, assuming an unbiased selection process) they follow the same distribution as their superior set:

$$\mathbb{S}' \sim \mathbb{S} \Leftrightarrow \mathbb{S}' \sim \mathcal{S} \Leftrightarrow \forall x \in \mathbb{S}' : x \sim \mathcal{S} \tag{3.26}$$

In the scope of this thesis, out-of-distribution (OOD) and drift detection are seen as two manifestations of the same problem: An OOD sample x' is (as the name implies) a sample that is not drawn from the same distribution as the training set $\mathbb{S}^T \sim \mathcal{S}$ [CC19]:

$$x' \in \mathbb{S}^{OOD} \Leftrightarrow x' \not\sim \mathcal{S} \tag{3.27}$$

Data drift occurs when there is a whole shift over a whole subset $\mathbb{S}'$ of samples observed during operations[BS18]:

$$isDrift(\mathbb{S}') \Leftrightarrow \mathbb{S}' \not\sim \mathcal{S} \tag{3.28}$$

Both falsify a different statement in equation (3.26): OOD aims to find singular samples that disprove $x \sim \mathbb{S}$ while drift measures the shift in overall sample distribution, testing $\mathbb{S} \sim \mathcal{S}$ directly. If either is present, the assumption of generalization (equation (2.2) on page 13) is violated.

There are also many closely related problems, like novelty detection [PAD18] (finding interesting samples, *e.g.* for active learning), anomaly detection [Pan+21] (finding samples with dissimilarities towards the training data, but missing the classifier present in other scenarios) or certainty metrics (usually in the domain of adversarial learning [SG18; Fei+17]). All of them aim to find samples that do not follow the underlying data distribution of $\mathbb{S}^T$.

3.4.1 Basic Drift Detection

I define a drift detector $d(x)$ as a function that correlates with the dissimilarity between a sample and the training data (shortened to d). When the average d is higher than expected it should indicate systemic drift. OOD samples should be visible as singular, unusually high spikes in d. Its value can also be seen as the **confidence** that a classifier $M(x)$ trained on $\mathbb{S}^T$ is able to produce a reliable result on x.

For low dimensional data, defining d is fairly straight forward. Standard statistical tests often suffice to handle both cases. On a per-sample basis, the mahalanobis distance [Lee+18] does so. For drift detection on data streams the Kolmogorov-Smirnov test [Rei+16] or p-Values [Jor+17] might be used. More sophisticated methods use unsupervised models to capture more abstract correlations that basic statistical measures can not capture, *e.g.* margin density drift detection (MD3) [SK17].

3.4.2 Sample Distance for Images

For images, finding a suitable metric d is harder. The vector representation of a digitized image (its pixels) is by definition high dimensional. Correlations often not confined to a singular dimension/ a fixed position in all images: Their vital information is usually encoded in higher level features that are abstract, non-linear and may have multiple concrete representations for the same object. For example an image shifted by one pixel would be indistinguishable from its original by the human eye, but every value in its representation would be assigned to another dimension inside its basic feature vector.

Thus, an ideal $d(x)$ must select meaningful high level features $f'(x)$ generic enough to assign similar values to images with equivalent content but so specific that any dissimilarity towards $\mathbb{S}^T$ must result in a high output value. There are two basic ways to generate such features: Model centric methods that use an intermediate representation $f'(x) : g(f'(x)) = M(x)$ of the classification model M. Or data based approaches, which learn an adequate low-dimensional feature representation $f'(x) : x \approx g(f'(x))$ of $\mathbb{S}^T$.

Classifier Output

The most abstract representation based on M is its output itself. Classification output is usually represented by values modeling the posterior probabilities $M(x) \hat{=} p(y = c|x)$ of the label y belonging to class c. Since $\mathbb{S}^T$ usually has clearly defined labels, $\forall x \in \mathbb{S}^T : p(y = c|x) \in \{0, 1\}$. Thus:

$$M(x) \notin \{0, 1\} \rightarrow x \not\sim \mathcal{S} \tag{3.29}$$

allowing d to be simply based on the distance from these values, *e.g.*:

$$d(x)_{co} \coloneqq \|M(x) - \overrightarrow{0.5}\|^2 \tag{3.30}$$

This approach has one fatal flaw: The negation does not hold:

$$M(x) \in \{0, 1\} \not\Rightarrow x \sim \mathbb{S}^T \tag{3.31}$$

Especially when in the area of extrapolation, the model may assign very high confidence scores for potentially wrong results (indicated in section section 2.2.2). Despite that, the approach usually solves as a good baseline for drift and OOD detection [HG16].

Distribution Over Intermediate Layers

To receive features more abstract, one can use the distribution $\mathcal{F}$ of values over an intermediate hidden layer $f'(x)$ of M. This assumes that

$$x \sim \mathcal{S} \leftrightarrow f'(x) \sim \mathcal{F} \tag{3.32}$$

holds on the more abstract layers inside the network. It assumes that only meaningful features are extracted by M and are encoded in its inner layers. Thus, it is invoked layers preceding the output, typically before final pooling [Jia+18; PM18]. Intuition suggests that the classificator already filtered out unnecessary data at the higher layers, leaving only information used to classify the sample. [Abd+19] suggest finding the best suitable layer by applying an additional OOD dataset.

One possibility to quantify $f'(x) \sim \mathcal{F}$ is the Mahalanobis distance [Lee+18]:

$$d(x)_{Maha} \coloneqq \sqrt{(x-\mu)^T \Sigma (x-\mu)} \tag{3.33}$$

where mu is the mean vector and Σ the covariance matrix, assuming a normal distribution of $\mathcal{N}^n(\mu, \Sigma)$ for $\mathcal{F}$. It is basically a measurement of how many standard deviations x is away from the mean μ.

Other approaches stem from the field of adversarial sample detection [Goo15]. They try to evaluate whether samples are close to out-off-class samples, *e.g.* by counting k-nearest neighbors [PM18]:

$$d_{kNN}(x) := \frac{1}{N} \sum_l^N \frac{|\{c \in \Gamma_{l,k} : c \neq arg\,max(f(x))\}|}{k} \tag{3.34}$$

Here $\Gamma_{l,k}$ are the labels of the k nearest samples to the embedding layer $f_l(x)$ drawn from a calibration set $\mathbb{S}^E$. This approach is very similar to [MW17] which uses additional regularization on the embedding layer.

This metric was further modified in [Jia+18] by calculating only the ratio

of distance d between the k-th out-of-class ($\hat{\gamma}_{l,k}$) and the k-th in-class ($\gamma_{l,k}$) sample:

$$d_{dist}(x, l) := \frac{d(f_l(x), \hat{\gamma}_{l,k})}{d(f_l(x), \gamma_{l,k})} \tag{3.35}$$

The problem of these methods is that they are specifically tailored towards adversarial attacks, completely disregarding the extrapolation area [UC21].

A theoretical shortcoming of intermediate layers is their missing guarantee that information removed by $f'(s)$ is also unimportant to unknown samples. For example, classification might disregard important features that fall completely out of the distribution of $\mathbb{S}^T$ because they simply do not exist in there [Meh+21]. Thus they might capture small drift well but fail when encountering OOD samples (*e.g.* showing new defect categories). This is shown through experiments done in section 5.3.1 on page 76.

3.4.3 Data Based Approaches

Another approach towards defining M_d would be to ignore the features created by the classification model M altogether and derive them from the data itself. To measure similarity between a sample x and an overall distribution $\mathcal{S}$, one must break $\mathcal{S}$ into meaningful features L that allow to model the whole distribution.

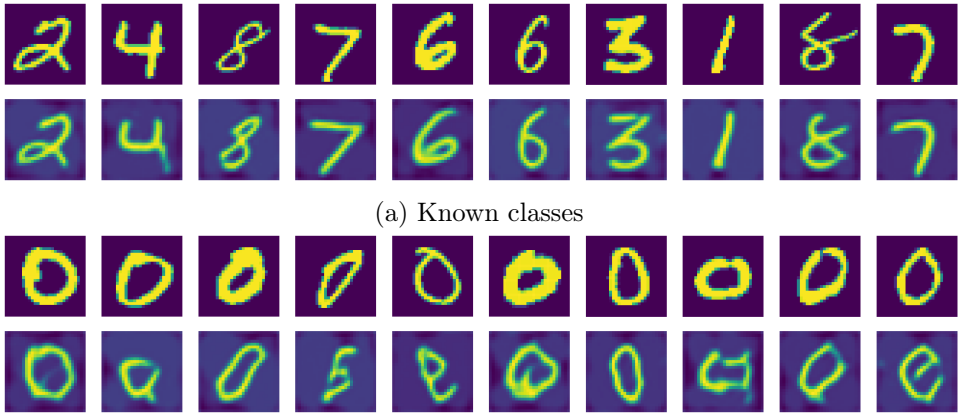
For complex data, this can be done through unsupervised learning. Recent theoretical research implies that this improves overall stability and uncertainty estimation, even more so than direct classification on concrete OOD samples [Hen+19].

Auto Encoders

Such an unsupervised method is provided by the AE [HZ93]: To reduce the input dimensionality, an encoder model *enc* is trained. It maps the data to a minimal representation L (called the latent space) which has a much smaller dimension than s, while a corresponding decoder model *dec* is tasked to properly reconstruct s from L. By selecting a size for L that is less than the amount of samples in $\mathbb{S}^T$ it is forced to find a good feature representation for $\mathbb{S}^T$, avoiding the memorization problem [BW21].

Thus, the loss is defined as:

$$\mathcal{L}(\theta_{enc,dec}, \mathbb{S}^T) \coloneqq \sum_{(x,y)\in\mathbb{S}^T} \mathrm{E}(dec(enc(x)), x) \tag{3.36}$$



(a) Known classes

(b) Unknown class

Figure 3.9: Reconstruction of an Auto-Encoder trained on the MNIST dataset. All samples containing zeroes have been omitted from the training set, resulting in the models disability to reconstruct them properly.

The same reconstruction error may be used to measure the dissimilarity between the sample and the original dataset. AEs falls under the same generalization constraints introduced for most ML methods in figure 3.9. Thus, the d-metric may be defined as:

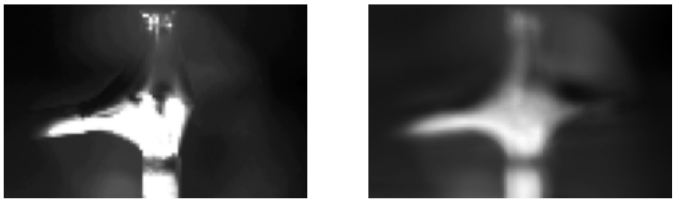
$$d_{AE}(x) \coloneqq \mathrm{E}(dec(enc(x)), x) \tag{3.37}$$

For anomaly detection, this formulation is especially useful: Singular anomalies (*e.g.*. scratches on a surface) can be localized due to their local reconstruction error. It can be calculated on a per-pixel by removing the summation in the calculation of $d_{AE}(x)$[Cho+20b]:

$$d_{AE}(x)[i,j] \coloneqq |dec(enc(x))[i,j] - x[i,j]| \tag{3.38}$$

PCA

One of the more well known methods to construct *enc* and *dec* is by using the principal component analysis (PCA)[LLZ14]. It deconstruct the sample data s into principal components (or eigenvectors) w that are ordered by the amount of variance they can encode.



(a) An image and its reconstruction.

(b) The first ten eigen-vectors of the PCA.

Figure 3.10: Example of a PCA encoding with $L = 10$.

$$\begin{aligned} \hat{X} &= X - \sum_{i=i}^{n-1} X w[n] w[n]^T \\ w[n] &\coloneqq \arg\max_{\|w\|=1} \|\hat{X} w\|^2 \end{aligned} \tag{3.39}$$

calculates the n-th component, where X is the matrix representation of $\mathbb{S}^T$ with the mean of each sample shifted to zero and $w[n]$ is the n-th eigenvector of X.

This allows to encode a sample x into a smaller representation

$$\begin{aligned} \mu_x &\coloneqq \frac{1}{\dim(x)} \sum_{i}^{\dim(x)} x[i] \\ enc_{PCA} &\coloneqq (s - \mu_x) W_L \end{aligned} \tag{3.40}$$

of latent dimension L, where W_L is the matrix representation of the first L components. The sample can be reconstructed through inversing the operation:

$$dec_{PCA} \coloneqq (enc_{PCA}(x) W_L^T) + \mu_x \tag{3.41}$$

which is illustrated in figure 3.10.

This naive approach has a few shortcomings of its own. It can't capture

effects that are (close to) linear like increased image brightness, since they can be perfectly encoded in μ_x without these effects represented in $\mathbb{S}^T$. They are also not able to encode non-linear transformations like shifts or rotations, since *enc* is a linear transformation itself.

Deep Auto Encoders

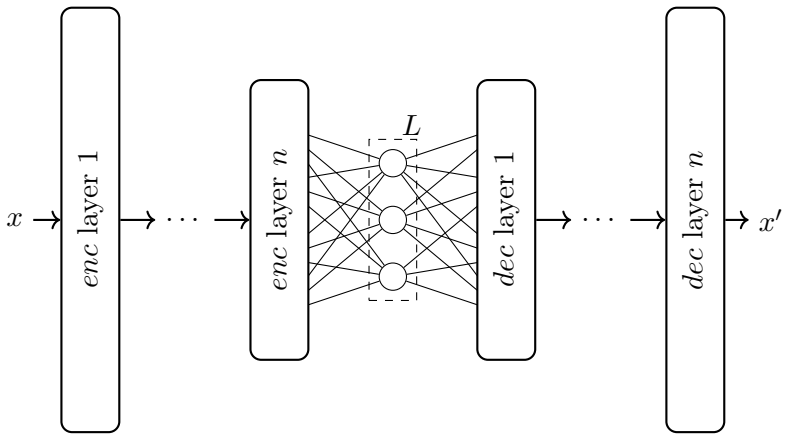


Figure 3.11: Auto-Encoder (AE) built from two deep ANNs.

To overcome these limitations, a more sophisticated approach is required. In recent years ANNs have been employed, using up- and down-sampling layers to reduce and recover the dimensions of x [ZP17]. Both the output of the encoder network and the input to the decoder are implemented through dense layers (see figure 3.11).

Instead of using the reconstructing error, it is also possible to employ the latent space as an abstract representation of the input sample x. Thus similar methods to those introduced in section 3.4.2 on page 40 may be applied. Instead of working on the classification model, the distribution over $enc(x)$ is compared to identify OOD samples. This can be done by using distance to clusters [Ayt+18], density estimation [Aba+19] or even comparing with singular samples from the same latent representation [Gon+19]. Even more promising is using the Mahalanobis distance (see equation (3.33) on page 41)[Den+18] to calculate a statistical distance between x and $enc(x)$.

3.5 Automatic Camera Calibration

After detecting drift in the system, it is often necessary to remove such drift. This requires the sensor to be set up in a way that its captured data resembles the samples from $\mathbb{S}^T$ again. To do so, the intrinsic camera parameters like focal length or field of view have to be set to the appropriate values. Additional, its positional orientation has to be corrected. This is usually termed calibration.

When operating a ML system, it is often not clear what the ideal parameters are. A human might not be able to deduce from the images alone if the calibration is correct. Thus some form of reference or automatic calibration is needed. Classicaly automated calibration was done with a reference pattern like a checkerboard [DF18] or an otherwise textured planar object [HKH16]. Both methods were improved by applying ANNs [CLX21; BG20].

Estimating good parameters from images alone has been made possible through modern machine learning (ML). This is either done by estimating specific parameters like field-of-view [Bog+18], radial distortion [Cha+20] or spacial translation and rotation [Zha+21b]. Further approaches use additional sensors for fusing data like a 3d sensor [Wu+21; Sch+17], but this is usually not available. A method that estimates arbitrary parameters and uses captured images alone, as proposed in this thesis, is not known to the author.

3.6 Pool Based Active Learning

Industrial manufacturing may produce millions of images a week, most of which don't hold any novelties beneficial for the training. Thus, a smart selection of important samples is required. This aligns with pool based scenarios in **active learning** [CAL94]: They assume a large pool of unlabeled samples $\mathbb{S}$. Based on a confidence metric c samples are selected for labeling by an **oracle**. It usually denotes a human with domain knowledge that is tasked to assign ground truth to selected samples.

Algorithm 2 on the following page specifies the main loop of active learning. Observed samples are evaluated on the current model using a query function c. Any samples with a confidence lower than the threshold t_c are labeled by the oracle and added to the training set $\mathbb{S}^T$. Usually, the process of finding beneficial samples and improving the model is repeated until a certain convergence criterion or fixed amount of samples is reached. Active learning often leads to the same evaluation performance on fewer samples [Ren+21b].

Algorithm 2: Basic active learning loop.

$\theta_M \leftarrow \textsc{Train}(\mathbb{S}^T, M)$
while $\neg$FINISHED **do**
  $\mathbb{A} \leftarrow \emptyset$
  for $x \in \mathbb{S} \setminus \mathbb{S}^T$ **do**
    $c_x \leftarrow c(x; \theta_M)$
    if $c_x < t_c$ **then**
      $\hat{y} \leftarrow \textsc{Oracle}(x)$
      $\mathbb{S}^T \leftarrow \mathbb{S}^T \cup \{(x, \hat{y})\}$
    end
  end
  $\theta_M \leftarrow \textsc{Train}(\mathbb{S}^T, M)$
end

The design of c determines the query strategy: Uncertainty-based querying measures how certain the model is to assign the correct class. Simple approaches usually take the per class output of the model itself, *e.g.* margin sampling (output value of highest minus second highest class) or entropy reduction (calculating the information entropy over the whole output) [Set09]. This often neglects the relationship between samples, carrying the risk of selected samples being to similar, thus not covering the whole space of beneficial samples. Density-based querying aims to find diverse samples by enforcing a homogeneous distribution of samples, the so-called core-set [Phi17]. Hybrid query strategies like [Kim+21b] try to take both factors into account.

3.7 Explainable AI

Explainable AI (XAI) [Xu+19] is a rather novel field in artificial intelligence (AI). With models becoming more and more complex, amassing millions of parameters, it is no longer possible to determine what features of an input contributed to their explicit output. The ML model is usually considered a *black-box*, with its inner workings obscured to an observer. XAI tries to provide additional information to a user to better understand the reasoning behind a decision.

Especially when encountering deviations from our own human reasoning or encountering bias, as described in section 2.2.2 on page 13, XAI can help to improve trust into deep ML models [vHF22]. There are two major approaches towards explaining models [LPK20]: Either model agnostic methods that

construct simpler models from correlating model input and output, treating the classifiation model as a black-box, *e.g.* SHAP [LL17] or LIME [RSG16]. Or model specific ones like GradCAM [Sel+17] or Saliency Maps [SVZ14]. The later try to augment and locate features representations inside the model to determine whether the model bases its predictions on the actual data or some arbitrary artifacts.

In this thesis, I will introduce a new method based on AE reconstruction. There are only few works that graze this approach: [BZ23] only looks if an variational Auto-Encoder (VAE) produces interpretable features in the domain of music generation. [Agu+23] creates AEs from decision trees to increase their interpretability, while [Ba-+23] also only tries to explain the auto encoder itself. [KC21] creates explanation based on the latent space, but not its reconstruction. According to [SF23] there exists no XAI method based on AEs, except [Yan+18] where generated point clouds are visualized to illustrate training convergence.

3.8 MLOps

In software development, the paradigm of unified software development and IT operations (DevOps) has risen in importance over the last decade. It sees both development and operations as one interconnected process that needs to be automated, encompassing software versioning, code reviews, testing alongside building, deployment and monitoring. This is covered by a vast amount of practices, tools and software products [GP22].

In the past, the operational side of machine learning was often not considered. Research usually evaluates systems only up to the initial training. Only recently, the term of MLOps [GSM21] has been coined to develop tools that unify the whole ML development process, including operations. The entire process is broken into stages, although these are defined differently by each implementation [SS23]. Thus I will depict my own definition of an MLOps workflow in figure 3.12 on the following page. It consists of the following stages:

Data Management: Data has to be gathered from various sources like active production, samples generated in a lab environment or existing datasets. It has to be pre-processed and stored to be available for the subsequent steps.

Labeling: Ground truth has to be provided by a domain expert or other

means. The workflow should provide tools to efficiently do so.

Architecture Selection: An adequate model architecture must be provided based on the task, nature of available data and other constraints.

Training: The selected model must be trained with the data. To do so, the hard and software required must be provided automatically. During and at the end of training its performance should be monitored and evaluated, resulting in the decision to either deploy the new model or trigger additional data gathering.

Deployment: The trained model must be deployed to an endpoint where other applications can access and infer it.

Monitoring: The availability, performance and stability must be monitored. If these are not guaranteed anymore, warnings must be triggered. These can result in a retraining of the model, but may also require other actions to be taken.

New software products like AzureML [Tea16] or Databricks [RCF19] are in active development to cover these requirements. They often focus a lot on the labeling and training steps of an machine learning project and only provide rudimentary solutions for model supervision. Thus, a better solution is required, especially in the manufacturing domain where long term operations is the primary focus of almost any software project.

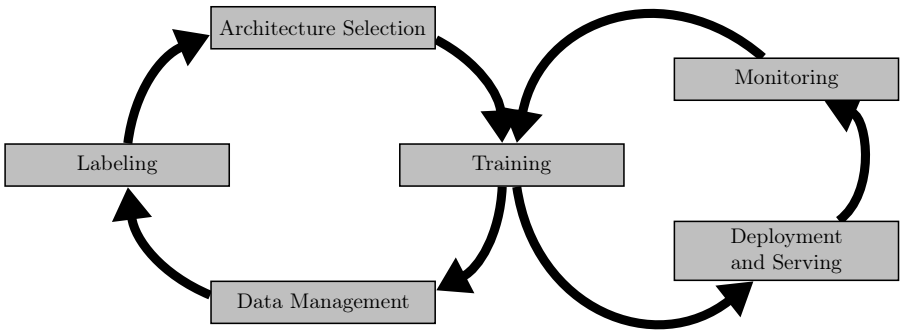


Figure 3.12: Example for a definition of stages inside an MLOps workflow.

There is not a lot of research into ML operations specific to manufacturing. Only a few approaches go past the scope of only devising a classification model. [Sch18] tried to predict the necessity of quality controls based on data

collected over multiple machines of a printed circuit board (PCB) assembly line, utilizing cloud computing. Other approaches may acknowledge the dynamic environment of manufacturing, but focus on quality prediction [TM23] or hardware solutions [HHZ22].

3.9 Conclusion

In this chapter I outlined the state-of-the-art in machine learning through ANNs. I presented multiple topics that concern my approach towards industrial ML in one way or the other: Be it optimization, active learning or explainable AI. Additionally, I showed the approaches available from an operations point of view: SPC and MLOps. To conclude, I want to emphasize that a unified approach for manufacturing ML does not exist yet. All these methods are used disconnected from each other.

In the upcoming chapter I will present the visual machine learning control (VMLC) framework that will bring operations and ML together. It does so by using techniques like OOD/ drift detection combined with self calibration and SPC, all presented in this chapter. VMLC provides a new solution for the operational part of machine learning in the manufacturing environment.

Chapter 4

Proposed Solution

> The most effective way to improve productivity is to eliminate work.
>
> *William E. Conway, Jr.*

In this chapter, I will introduce a unified solution to the problems introduced in section 2.3. It will be an extension to the basic inference pipeline most visual machine learning (ML) applications are based on:

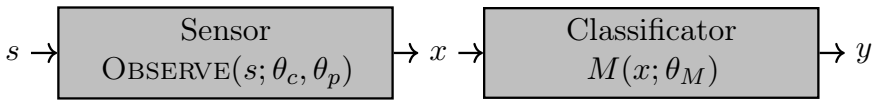


Figure 4.1: Basic inference pipeline.

A physical sample s is observed by a **sensor** (the camera), which is configured by the calibration vector θ_c, producing a measurement (image) x. This step is modeled in the function OBSERVE$(s;\theta_c,\theta_p)$. θ_c contains intrinsic values to the measurement like focal length or position of the camera. θ_p encompasses all unknown external factors like a dirty lens, particles in the air or environmental light. They resemble any influence that changes the outcome of measurement x.

Afterwards, the trained model $M(x;\theta_M)$ is invoked, producing a classification result y. In the usual case, this completes the inference pipeline illustrated in figure 4.1.

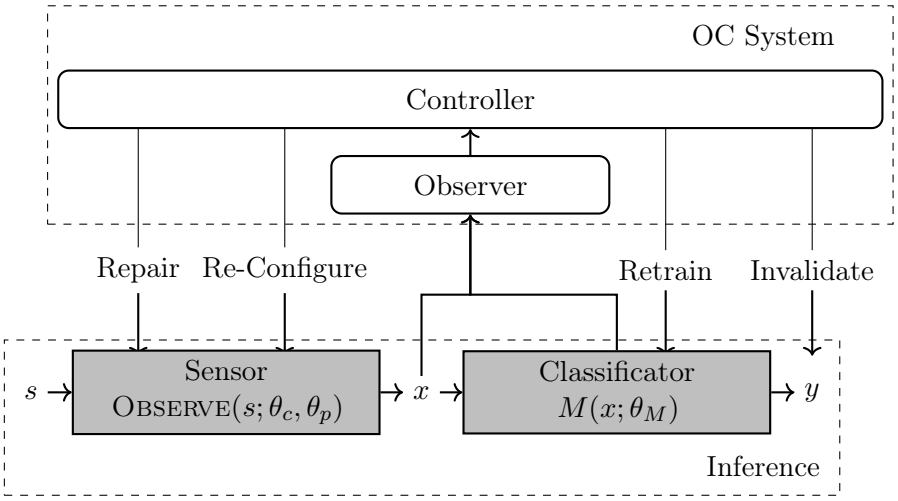


Figure 4.2: Observer controller architecture over the base classification system.

4.1 Observer Controller Architecture

In section 2.3 on page 21 I specified requirements for a ML system to handle the challenges of industrial operations: It must be verifiable, explainable, repairable and optimizable. This falls deeply in line with the organic computing (OC) paradigm [Sch+10]. OC draws inspiration from emergent systems that have so called **self-x** characteristics, like self-explaining, self-healing and self-optimization. These are basically reformulations of my own requirements.

Thus, I decided to derive a ML system the OC way: By implementing a so called **observer**/ **controller** structure [SM05]. These structures contain two components:

Observer: An agent observes both the available data about the environment (input data, *e.g.* the image) and the internal state of the system (internal values inside the model). It creates a representation the controller can act upon.

Controller: The controller decides which actions are required to keep the system in a desirable state. It might trigger a reconfiguration of the system by changing internal parameters, request certain actions to occur or act as a safeguard that keeps the system from doing undesirable actions.

It is built on top of the existing system and enables it to implement the various self-x capabilities.

Figure 4.2 on the previous page illustrates how the observer/ controller architecture should interact with the basic classification system consisting of sensor and model. It has various actions available to react upon its observations:

Re-Configure: If the problem is with a misconfiguration caused by sensor drift or other intrinsic issues (*e.g.* wrong configuration file loaded for current product type), it can either rely on an **operator** to fix the sensor parameters or try to do so itself (a possible solution to do so is presented in section 4.4 on page 63).

Repair: If the problem is not resolved through configuration alone, it must be influenced by outside factors. Since these stem from the physical world outside the systems control, it cannot interact with them directly. Thus it must further rely on human intervention and possible investigation into the issue. In the process, the human becomes the physical extension of the system, helping it to achieve its self-organization. Either the undesirable behavior lies with the system, be it faulty equipment, mechanical errors or other perturbations. Or it is caused by problems outside the systems borders, like a change in process or material. Such issues are investigated by a **process engineer**.

Retrain: If the problem is not with the sensor but outside factors, it has to be decided by the process engineer whether the process has to be corrected or the classification model has to adapt to the new challenges. If true, a reparametrization of the model is required. It must be retrained with new data that stems from the current situation, containing images of the new product, altered imaging process or novel, yet undiscovered classes. To do so, it can provide the relevant data observed by the observer, but has no information about the ground-truth required for training. Thus, a human is once more required. This time, a domain expert is needed to function as human-in-the-loop [CFE22], leading to a continuous self-optimization through repeated cycles of live operation and model improvement.

Invalidate: Actions that necessitate a corrective action on the system usually also implicate that the current classification is not to be trusted, since the current state is unstable. Hence, the controller needs to invalidate any classification that may be compromised to prevent faulty products to be processed further. Keeping these products and their data aside

for future investigation also helps to improve the underlying system and the whole manufacturing process.

The alert reader might have spotted that there is a multitude of people involved in various control actions. They are typical to any Lean manufacturing process [For96] and are as follows:

Operator: The person that operates the machine. This usually means maintaining components, monitoring if everything works as expected and fixing errors during operations.

Process engineer: A person that monitors the whole process over multiple manufacturing steps. Her goal is to increase yield, reduce faults and allover optimize the process.

Domain expert: An expert that is capable to decide over the defining qualities of a certain operation or control. This might be a quality assurance (QA) expert specifying quality requirements, a physicist that knows how certain properties are reflected in the data or even the process engineer himself.

This leads to the following hierarchy of control actions and required roles available to the system:

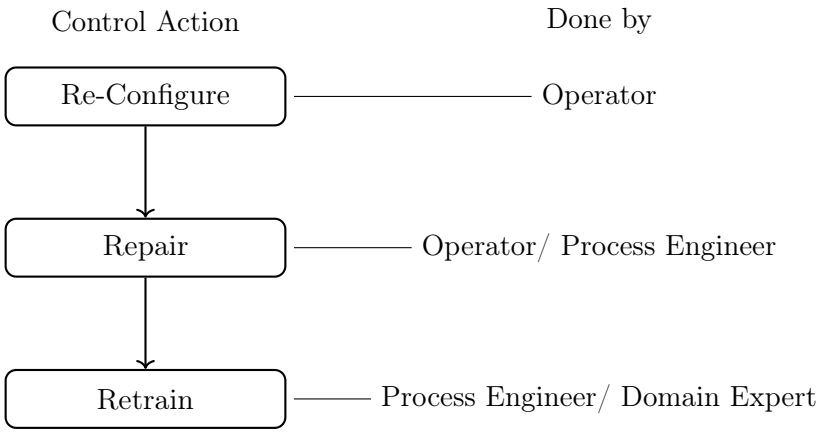


Figure 4.3: Hierarchy of control actions handled by various actors.

The general idea behind locking certain actions behind others is derived from the LEAN paradigm [WRJ90]: Errors should be fixed where they originated from. ML demands a stable process. If it is not stable, there is no

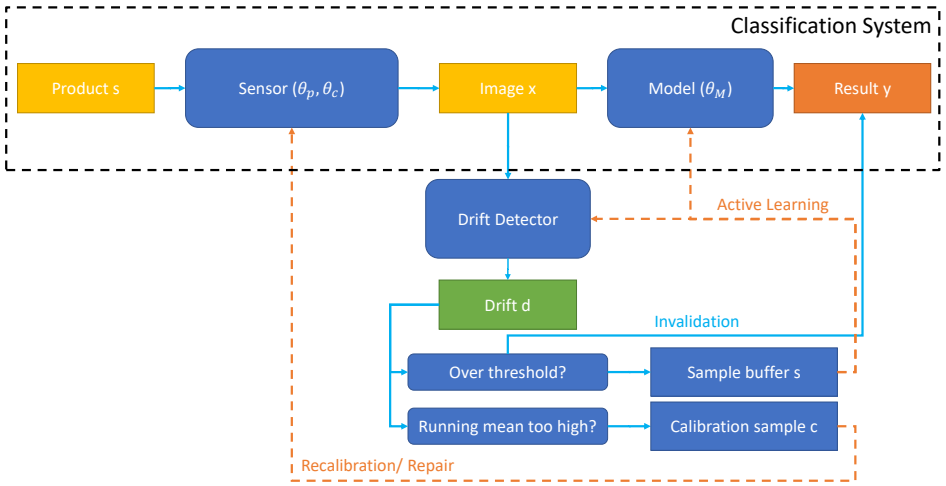


Figure 4.4: Core classification system with added process control.

benefit in improving the classification model, which is usually a costly affair in both time and money (domain experts are usually more costly than operators, retraining requires time to label further data and often triggers various time consuming processes like auditing or halting production for deployment).

Additionally, an observer controller system needs to be as little invasive as possible. If it triggers unnecessary control actions, trust in the system will soon degrade.

4.2 The VMLC Architecture

To implement this architecture, I propose visual machine learning control (VMLC), shown in figure 4.4, first published in [Mas23b]. VMLC introduces the following components:

- A **distance model** $M_d(x; \theta_d)$ that estimates the **sample distance** d for each observation x.
- A criterion to determine whether the value of d is out of distribution.
- A **sample buffer** $\mathbb{B}$ that stores any out-of-distribution (OOD) samples for further inspection.
- A method to determine if drift is present and handle accordingly.

In the upcoming sections, these components will be discussed in depth:

4.2.1 Distance Model

The main target of the observer is to find a simplified representation of the systems state that encapsulates the stability of the system. In this thesis, I assumed that all training samples $x \in \mathbb{S}^T$ came from a stable system and follow the underlying distribution $\mathcal{S}$. Thus, any sample similar to this set is considered from a stable state, while a dissimilar one suggests instability.

To measure this similarity, the distance metric d is introduced as the output of the distance model $d(x) = M_d(x; \theta_d)$. It should enforce the following inequality on all data observed during production:

$$\forall x \sim \mathcal{S},\, x' \not\sim \mathcal{S} : d(x) < d(x') \tag{4.1}$$

Any sample that is not from the same distribution as $\mathbb{S}^T$ should be assigned a high value, while similar ones shall receive low values in d score.

It is basically an anomaly detection built upon the base classification system. Other systems usually either only do the first, sacrificing classification accuracy in the process. Or they classify without any additional control imposed, risking wrong results through an uninformed classifier.

4.2.2 Out of Distribution Criterion

To find an adequate, model independent value at which to reject samples as OOD, I first define

$$\mathcal{D} \sim \{x \in \mathbb{S}^T : d(x)\} \tag{4.2}$$

as the distribution of $d(x)$ over the training data $\mathbb{S}^T$. When one assumes $\mathbb{S}^T$ to be of the same distribution as another data set $\mathbb{S}' \sim \mathcal{S}$, $\mathcal{D}$ must be the same across this set as well (since x represents the same random variable in both cases).

This allows me to define the conditional probability $p(d(x)|x \sim \mathcal{S})$ of observing a certain value of $d(x)$ during stable operations:

$$p(d(x)|x \sim \mathcal{S}) = P[\mathcal{D} \leq d(x)] = F_{\mathcal{D}}(d(x)) \tag{4.3}$$

through applying the cumulative distribution function (CDF) $F_{\mathcal{D}}$ on $\mathcal{D}$. It should be noted that this gives no information about the probability of $x \not\sim \mathcal{S}$ due to the distribution of $d(x)$ being unknown. But it specifies the likability of a certain value to be observed during stable operation. If a value is unlikely to occur on usual data, it is a strong indicator that the current sample is

OOD. This is expressed by setting a probability threshold p_{ood} where all samples are rejected if the inverse CDF is $1 - F_{\mathcal{D}}(d(x)) > p_{ood}$.

To receive a concrete value of $d(x)$ when this occured, the specification limit (SL) d_{max} is calculated by inversion:

$$d_{max} \coloneqq F_{\mathcal{D}}^{-1}(1 - p_{ood}) \tag{4.4}$$

Through this approach, the observer can be configured with a probability threshold independent from its respective distance model $M_d(x; \theta_d)$.

4.2.3 Sample Buffer

Gathering samples that need to be further investigated or included in future training is vital to improve the system's stability. Thus any rejected sample that violates the specification limit is added to the sample buffer $\mathbb{B}$. In general this means the image taken by the system, but may also include the physical product for physical examination (*e.g.* a failure in process resulted in unknown physical deformations). This may be especially helpful if new defects arise or the imaging device shows unpredictable behavior that needs fixing.

Samples in the buffer need to be labeled by a so called **oracle**, a domain expert that can assign ground truth for a future re-training. When the model is trained with new data, $\mathbb{B}$ is added to the training set and used for both the new model M as well as the drift detector M_d.

4.2.4 Drift Detection

In VMLC, drift is seen as any instability in the process. Whether it stems from outside sources or the system itself is not relevant to detect it. As soon as it emerges, its cause has to be investigated. Thus drift is defined as the deviation of the mean $\overline{d}$ of d values from the expected mean μ over a fixed amount of time.

This approach is equivalent to statistical process control (SPC), which was introduced in section section 3.3. To calculate the control limits in accordance with equation section 3.3, $\mathcal{B}_{\mu}(\mathcal{D}, N_s)$ specifies the mean of $\mathcal{D}$ batched into

groups of N_s samples:

$$\begin{aligned} \mu &\coloneqq \frac{1}{|\mathcal{D}|}\sum_{d}^{\mathcal{D}} d \\ \sigma^2 &\coloneqq \frac{1}{|\mathcal{B}_\mu(\mathcal{D}, N_s)|}\sum_{\mu_d \in \mathcal{B}_\mu(\mathcal{D}, N_s)} (\mu_d - \mu)^2 \\ UCL, LCL &\coloneqq \mu \pm \sigma \cdot l_\sigma \end{aligned} \tag{4.5}$$

The key difference to most SPC applications is that the stability of the process is not compared with an arbitrary past period, but rather the data distribution during training. We assume that system stability was given during the training and any deviation from this state might lead to diminished classification performance.

I furthermore specify the **performance** *acc* of a classification model M as its likability to correctly assign the correct label $\hat{y}$ to a sample from a given set $\mathbb{S}$:

$$acc(M, \mathbb{S}) \coloneqq p(M(x) = \hat{y} | x, \hat{y} \in \mathbb{S}) \sim \mathcal{L}(\mathbb{S}) \tag{4.6}$$

It can be assumed that *acc* is at its highest when the generalization assumption formulated in equation (2.2) on page 13 holds. Thus, based on equation (4.1) on page 56 the distance metric should also fulfill the following inequality:

$$\begin{aligned} & acc(M, \mathbb{S}^d) < acc(M, \mathbb{S}^T) \\ \Rightarrow & \mathbb{S}^d \not\sim \mathbb{S}^T \\ \Rightarrow & \frac{1}{|\mathbb{S}^d|}\sum_{x \in \mathbb{S}^d} d(x) > \frac{1}{|\mathbb{S}^T|}\sum_{x \in \mathbb{S}^T} d(x') \end{aligned} \tag{4.7}$$

Thus, as long as $\mathcal{B}_\mu(\mathcal{D}, N_s) \sim \mathcal{N}(\mu, \sigma)$ is probable the classifier accuracy should be at its maximum. Any drop in performance should be immediately result in an increase in d, making it an estimation of the classification performance without requiring the ground truth.

4.2.5 Algorithmic Description

Algorithm 3 renders the complete VMLC system in algorithmic form. In line 1 and 2, the initial models are trained on the available training data $\mathbb{S}^T$. Other variables are set up in line 3 through 7: The sample buffer $\mathbb{B}$ is initially empty since no productive samples have been encountered yet. $\mathcal{D}$ is modeled

Algorithm 3: Main algorithmic loop of VMLC.

```
   /* Initialize variables.                                          */
 1 θ_M ← Train(θ_M, 𝕊^T, M)
 2 θ_d ← Train(θ_d, 𝕊^T, M_d)
 3 𝔹 ← ∅
 4 𝒟 ← {x ∈ 𝕊^T : M_d(x; θ_d)}
 5 μ, σ ← 𝒩(𝒟)_{N_s}
 6 UCL ← μ + l_σ σ
 7 d_max ← F_𝒟^{-1}(1 − p_ood)
 8 for s_i ∈ 𝕊^P do
 9     x ← Observe(s_i; θ_c, θ_p)
10     d ← M_d(x; θ_d)
       /* Check if OOD sample is discovered.                         */
11     if d > d_max then
12         𝔹 ← 𝔹 ∪ {(x, Oracle(x))}
13         y ← 'Rejected'
14     else
15         y ← M(x; θ_M)
16     end
17     d̄ ← (1/N_s) Σ_{n=0}^{N_s} M_d(s_{i−n}; θ_d)
       /* Check if control action is required.                       */
18     if d̄ > UCL then
19         θ_c ← Re-Configure(s_i, θ_p, M_d)
20         if ¬ fixed then
21             θ_p ← Repair()
22             if ¬ fixed then
                   /* Invoke active learning.                        */
23                 𝕊^T ← 𝕊^T ∪ 𝔹
24                 θ_M ← Retrain(θ_M, 𝕊^T, M)
25                 θ_d ← Retrain(θ_d, 𝕊^T, M_d)
26                 𝔹 ← ∅
27                 𝒟 ← {x ∈ 𝕊^T : M_d(x; θ_d)}
28                 μ, σ ← 𝒩(𝒟)_{N_s}
29                 UCL ← μ + l_σ σ
30                 d_max ← F_𝒟^{-1}(1 − p_ood)
31             end
32         end
33     end
34     return (x, y)
35 end
```

after the outputs of the distance model M_d over the initial training data. The upper control limit UCL and specification limit d_{max} are set according to section 4.2.2 on page 56 and section 4.2.4 on page 57.

Lines 8 to 10 introduce the main loop: Each sample s entering the machine is first observed by the sensor calibrated by θ_c with outside influence θ_p. In the case of a vision system, this means a image is taken by a camera of the physical object. The dissimilarity between the known data $\mathbb{S}^T$ and the observed input x is measured by the distance model M_d parameterized by θ_d.

In line 11 through 16, the OOD criterion is invoked. If a sample violates the threshold, it is labeled as rejected and added to the buffer. Otherwise, it is labeled as class y based on the output of the classification model M.

To detect drift in the system this similarity value is averaged over the last N_s samples, represented by the variable $\overline{d}$ in line 17. This may be implemented through a moving average or batch-wise, invoking this part of the algorithm only after N_s samples have passed. Since d is a distance metric, no lower control limit is enforced.

If the upper control limit (UCL) is surpassed a control action is triggered, represented by lines 19 to 31. The first step is to fix the system by re-configuration, effectively finding a better configuration θ_c under the present sample s_i and perturbation θ_p. How this may be achieved is described in section 4.4 on page 63.

In line 20 the system is inspected whether the issue still persists. This may be done through a manual investigation or further evaluation of $d(x)$ on subsequent samples. If the metric is not returned to expected values, the problem needs escalation and the next step in the control hierarchy is invoked in lines 23 to 30.

If the system is deemed un-fixable by direct intervention alone, a full retrain of both models is required. To do so, the data gathered in $\mathbb{B}$ is added to the train set and emptied afterwards. This resembles the approach of pool based active learning introduced in section 3.6 on page 46. Based on the nature of the issue (*e.g.* a new product design is introduced where no productive data yet exists) additional data has to be gathered for retraining before operations may resume. Finally, all limits are re-evaluated on the new model configuration so VMLC continues to work with the new data included.

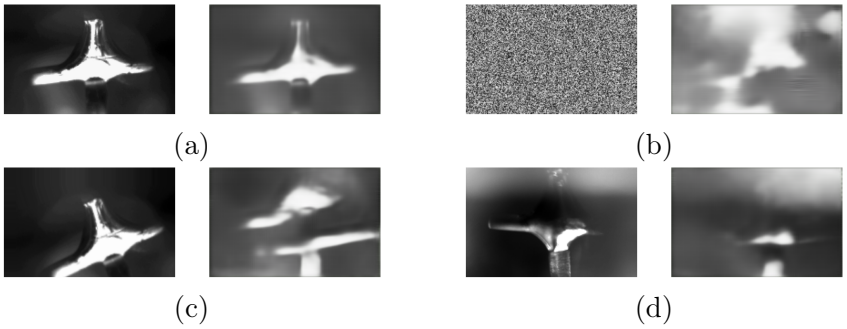


Figure 4.5: Successful reconstruction of an (a) similar sample as well as (b) random noise, a (c) transformed sample and a (d) OOD image captured under different lighting conditions,all of them not reconstructed properly.

4.3 Classifier Coupled Auto-Encoder as Distance Model

The distance model M_d has to accurately identify if $s \sim \mathbb{S}^T$ is probable. The less so, the higher its output d. Section section 3.4 presented viable candidates for this role. After evaluating these methods on actual manufacturing images, the Auto-Encoder (AE) method described in section 3.4.3 on page 45 proved to be the most promising [Mas23a].

To measure the probability of $x \sim \mathcal{S}$, an AE is trained to transform input data $x \in \mathbb{S}^T \sim \mathcal{S}$ into a low-dimensional latent representation L (the encoder part). From this, the decoder tries to recreate the original input $x' \approx x$.

Figure figure 4.5 shows the reconstruction of an AE on actual manufacturing data. It should be noted how every input produces an output that resembles a welding. But only when its input is similar to the training set an image close to the actual image is created. All other reconstructions, no matter if constructed from manipulated data, images captured under different lighting conditions or random noise all deviate significantly from their input. By applying an error function like mean squared error (MSE) between both images this dissimilarity can be condensed into a single scalar value.

Despite these promising features the basic AE approach has some shortcomings when employed as distance metric: There is no dependency between the classifier M and its distance model M_d. They are based on the same data, but there is no guarantee whatsoever that the features derived by the encoder *enc* are deciding for the classification model and vice versa.

In this thesis, I present a novel approach that combines both model

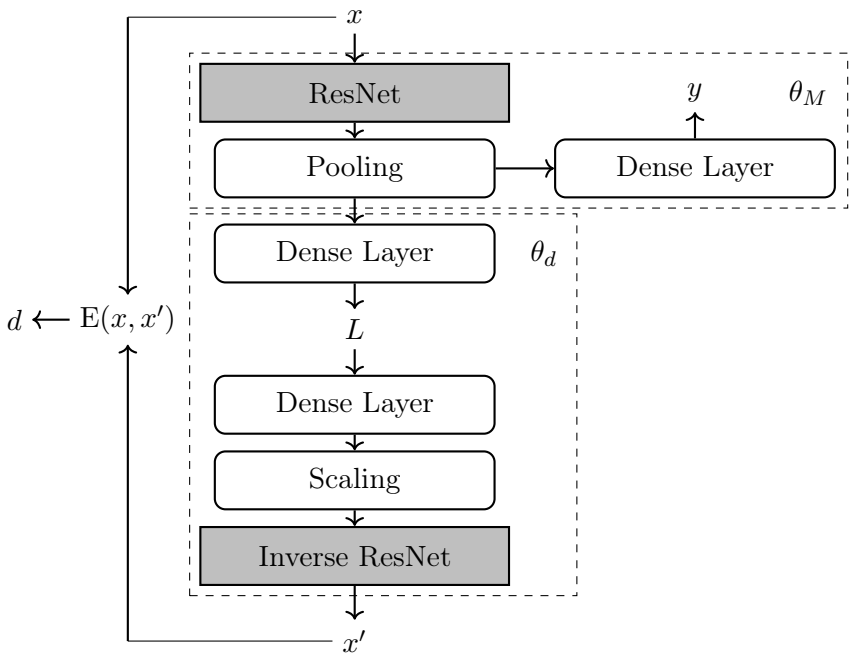


Figure 4.6: AE built upon the features learned by the classificator.

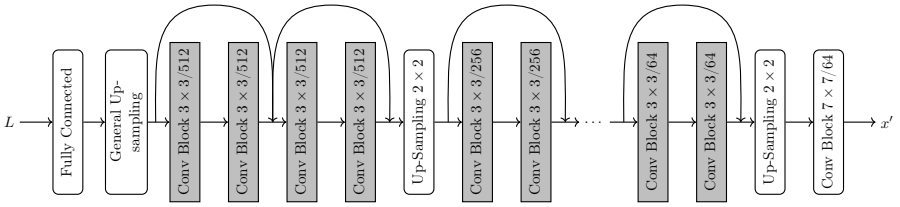


Figure 4.7: Inversed Residual Network (ResNet) architecture as decoder model.

and data based uncertainty to tackle this problem: A classification coupled Auto-Encoder (CCAE) that considers both data and model to not fall suspect to extrapolation while being based on abstract features of the classifier.

Figure 4.6 shows the general architecture of CCAE. It takes the lowest pre-output layer of M and constructs the latent representation L through a fully connected layer. The sample is then reconstructed by a separate decoder model that outputs a reconstruction x' with the same dimensions as x.

To build the decoder architecture, I used an inversion of the classification network as suggested in [Pu+16](see figure 4.7 for an example on a ResNet). The layer structure of the original classification model is traversed backwards. Pooling layers used to down-sample the initial data are replaced with up-

sampling layers, as described in section 3.1.3 on page 31. This design mirrors the encoder network to guarantee that it provides the same capabilities for reconstruction. It minimizes the risk that vital latent features are not properly processed by the AE, as may be by simpler architectures.

Through this approach it is of no importance whether the classification model is designed for label classification or image segmentation. Thus CCAE can be constructed for any common deep learning architecture available, as long as an inversion can be created. To not influence the classification performance, the parameters θ_M are fixed before training θ_d.

To improve performance, I implemented the structural similarity index (SSIM) loss between input and output of the AE. This is suggested by recent research on the SSIM as error function [Ber+18]. For industrial images its focus on smaller image regions and thus details may be especially helpful: Small features are usually the more important part of industrial images in contrast to overall structure, which MSE compares.

4.4 Self-Configuration

To improve the degree of self-organization of the system, capabilities to react on minor disturbances are desirable. In a time where there is extreme demands towards cost efficiency of manufacturing, resolving every issue by hand is undesirable. Especially when an operator might not know the ideal configuration parameters θ_c that resemble the conditions during model training the most.

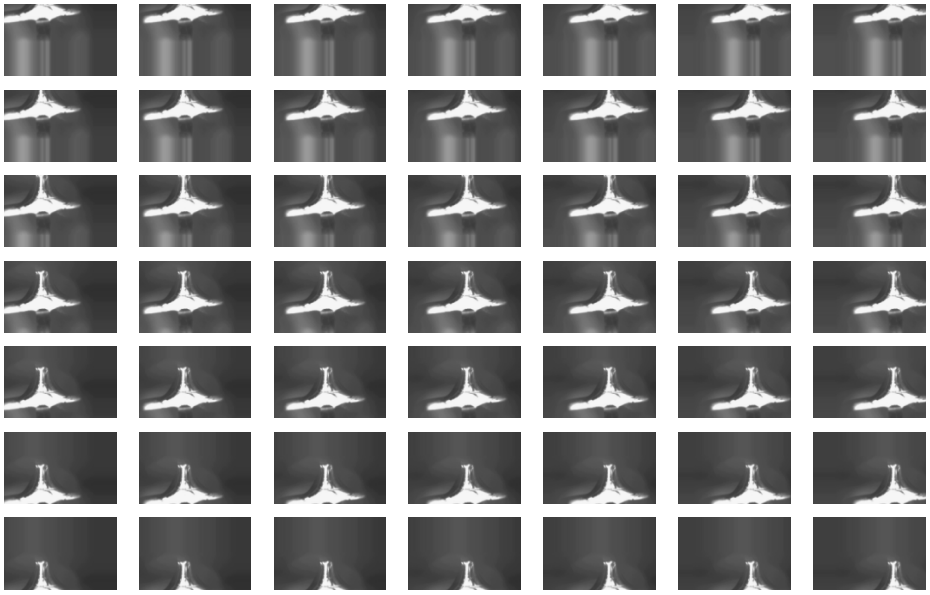
I denote the internal parameters θ_c^T and the outside influences θ_p^T as the configuration of the sensor during training time. Since M_d measures the dissimilarity between training data (and thus the setup during that period) and a sample s I assume that:

$$(\theta_c^T, \theta_p^T) \neq (\theta_c, \theta_p) \rightarrow d(\textsc{Observe}(s; \theta_c^T, \theta_p^T)) \geq d(\textsc{Observe}(s; \theta_c, \theta_p)) \quad (4.8)$$

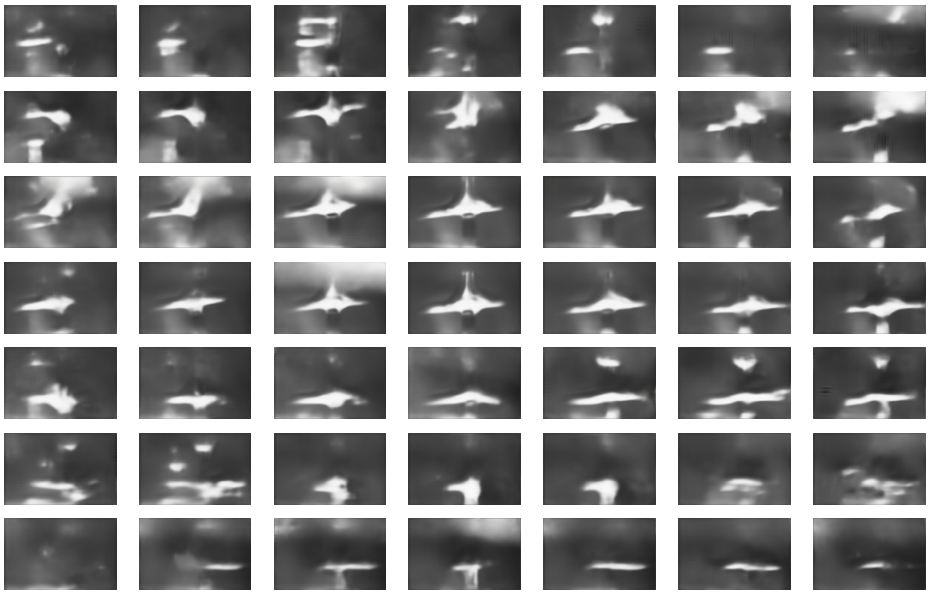
Hence d should have a clear minimum at the ideal parameterization of $\textsc{Observe}(s; \theta_c, \theta_p)$. This makes adequate system parameters recoverable by optimizing:

$$\arg\min_{\theta_c} \sum_{s \in \mathbb{S}} d(\textsc{Observe}(s; \theta_c, \theta_p)) \quad (4.9)$$

This is done under the assumption that the unknown θ_p can be negated by the appropriate setting for θ_c and the parameter optimization for the partic-



(a) Original distorted images.



(b) Auto-Encoder reconstructions.

Figure 4.8: Degrading reconstruction over distorted images.

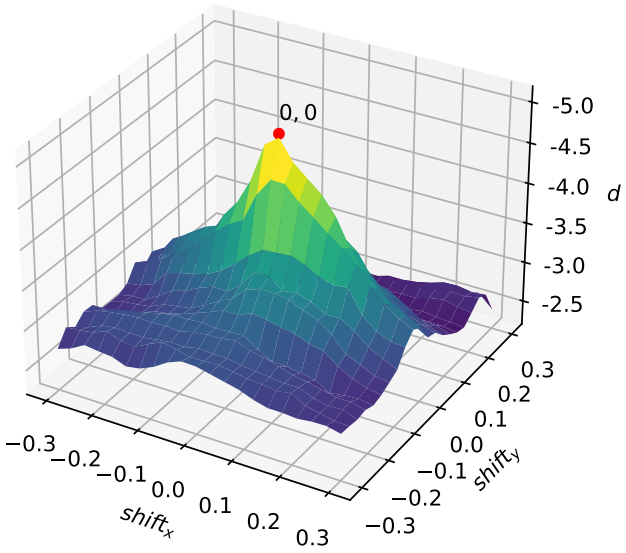


Figure 4.9: Value of d for a image translated into x and y directions with $\theta_c^T = [0., 0.]$.

ular sample s is also applicable to the subsequent samples. Experimentation with simulated drift through translating a image along its axes (figure 4.8 on the preceding page) has shown that d produces an optimizable landscape that is continuous, has no sharp fall-offs and a clear minimum at θ_c^T (see figure 4.9). This suggest that the idea to find an optimal configuration for θ_c by sampling and optimizing d over the parameter space of the camera is feasible.

Since $\textsc{Observe}(s; \theta_c, \theta_p)$ is a physical process that usually can not be adequately simulated or modeled, evaluating a specific value for θ_c requires the system to set its parameters, invoke the imaging process and calculate the d value on the image. This process is expected to be very time consuming, thus the underlying optimization method is required to sample as few times as possible.

To address these condition I employed Bayesian optimization as it was presented in section 3.2 on page 33. It is gradient free, does not assume an underlying function and works very well in an environment with few parameters but high sampling cost [Rub+19]. Possible alternatives to Bayesian optimization are discussed in section 7.2.3 on page 105.

To use Bayesian optimization for self-configuration I modified 1 on page 33

Algorithm 4: Basic Bayesian optimization.

```
/* Capture images for n_0 initial points from the
   parameter space.                                          */
```
$\mathbb{S}^0 \leftarrow \{i \in n_0 : (\theta_i, \textsc{Observe}(s; \theta_i, \theta_p^t))\}$
$\mathbb{P} \leftarrow \{\theta_i, x \in \mathbb{S}^0 : \theta_i \rightarrow d(x)\}$
while $|\mathbb{P}| \leq N$ **do**
  $\mathcal{P} \leftarrow P(d(\textsc{Observe}(s; \theta, \theta_p^t))|\mathbb{P})$
  `/* Find the next point to sample.                        */`
  $\theta_n \leftarrow \arg\max_\theta acqu(\mathcal{P}, \theta)$
  `/* Capture image` x `for configuration` θ_n`.   */`
  $x_n \leftarrow \textsc{Observe}(s; \theta_n, \theta_p^t)$
  $\mathbb{P} \leftarrow \mathbb{P} \cup \{\theta_n \rightarrow d(x_n)\}$
end
return $\arg\max(\mathbb{P})$

by replacing $f(\theta_n)$ with the distance metric d over $\textsc{Observe}((; \theta_c, \theta_p)$ see 4). Each time sampling is required, the camera takes a picture x of the physical sample s under the current configuration θ_n. Subsequently, $d(x)$ is invoked to form a scalar metric these parameters are optimized over. Note that only the calibration parameter is part of the parameter space θ since θ_p^t has to be seen as an unknown variable. It is considered constant at calibration time t (assuming no further perturbations are introduced during calibration).

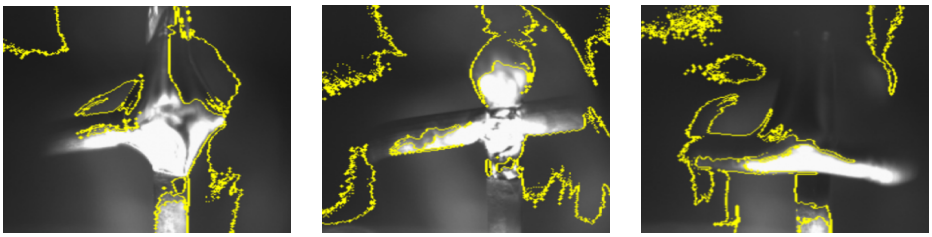
4.5 Retraining

Retraining is done based on the sample buffer $\mathbb{B}$, as already illustrated in 3 on page 59. This is done to select beneficial samples to add to the dataset, since labeling is a costly process. The high class imbalance and low inter-sample variance (see section 2.2.3 on page 17) lead to a huge mass of redundant samples, likely in the millions. To pick the few important ones is vital to not bore the domain expert and save time.

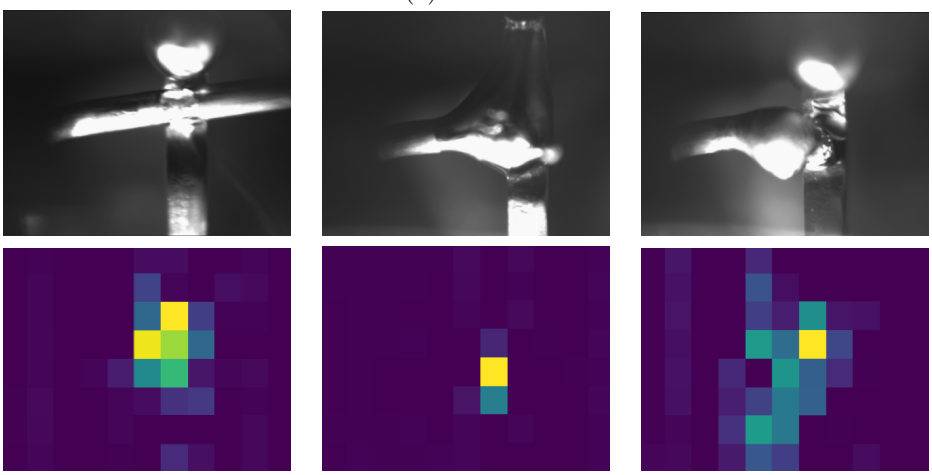
The sample buffer $\mathbb{B}$ is basically an implementation of pool-based active learning introduced in section 3.6 on page 46. Since the distance metric is already measuring dissimilarities, it should be a good indicator on interesting samples. To make the approach a hybrid one, samples should also be evenly spread out over the distribution of interesting samples. This might be done by employing density estimation over the latent space of the Auto-Encoder (AE) as in [Kim+21b].

It should be noted that each retraining on the classifier requires an simultaneous training of the distance model on the same data. This guarantees that the d metric is still based on the current classifier.

4.6 Self-Explanation



(a) LIME



(b) Grad-CAM

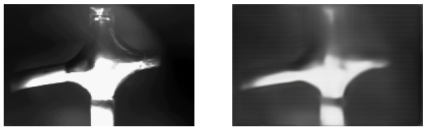
Figure 4.10: Example of explainability methods on welding data.

Figure 4.10 shows some of these methods applied to industrial data. Most visual applications focus on large scenes with lots of background noise and other objects. Thus, explanations usually focus on locating the important regions of the image. For manufacturing, it is obvious that this does not provide enough information for the operator to actual understand the reasoning of the classifier model.

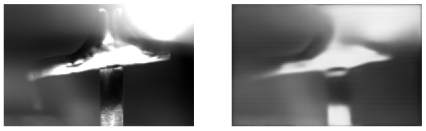
There is usually small background noise and classification is based on defined qualities or details of similar objects in contrast to distinguishing an object from other, different objects. Additionally, established explainable AI (XAI) methods are problematic in fields with high-stakes decision making,

since they can not guarantee reliable insights [VEA22].

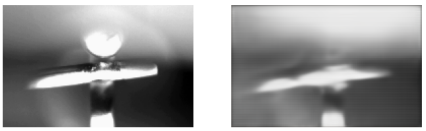
The CCAE presented in section 4.3 on page 61 presents a completely new possibility to tackle XAI: By reconstructing the input from the internal representation of the classification model, the user can *see* what the model sees. Figure 4.11 on the next page shows such reconstructions. It would be further possible to find similar images based on the latent space representation similar to [Gon+19], to show an user if the current kind of images have been encountered by the system before or if there is a significant lack of supporting data. Providing these kind of explanations could greatly boost understanding and confidence into the underlying ML classification system.



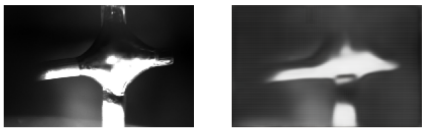
(a) Good reconstruction on unproblematic image.



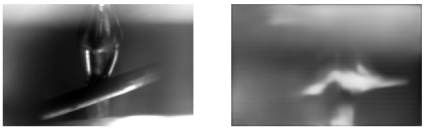
(b) The left part of the electrode is not considered, but also not important for classification.



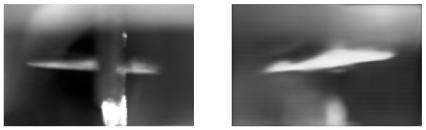
(c) The obvious weld bead is not detected due to excessive lighting but correctly labeled as *Defect* based on the electr.



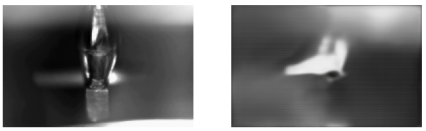
(d) The classification does not know samples with this kind of weld but labels correctly by accident.



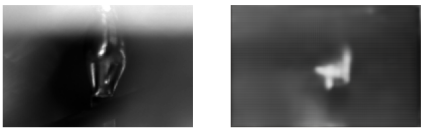
(e) This diagonally bent electrode is not known to the system. It reconstruct something close to an *Ok* sample, thus the miss-classification.



(f) Backwards bent electrode which is also unknown and reconstructed to an *Ok* sample.



(g) The electrode is missing all together, which is unseen in training data.



(h) Missing parts and low lighting.

Figure 4.11: Reconstruction of samples and derived explanations on their validity.

4.7 Conclussion

In this chapter, I presented my approach towards ML operations in an industrial environment: VMLC. By utilizing an observer controller structure based on combined OOD and drift detection the system is able to detect an unstable state or anomalous samples, handle them in an automatic and defined manner and make self improvements through the application of active learning and self-configuration. By leaning towards established methods like SPC, I make sure that the system is easily integrated into existing manufacturing workflows and structures.

To test how how well the system performs and if it can stand up to my claims, I thoroughly examined it through both experimentation and a real-world prototype deployed in a productive manufacturing line. These experiments as well as their outcome will be described in the upcoming chapter.

Chapter 5

Experiments

> Without data, you're just
> another person with an opinion.
>
> *W. Edwards Deming*

To verify the viability of the framework derived in chapter 4, I had to thoroughly evaluate and compare each component on data resembles the manufacturing domain. Thus, section 5.1 on the following page presents the selected datasets and how they where prepared in terms of cross-validation, subset selection and determining ground truth for out-of-distribution (OOD) samples. Section 5.2 on page 74 will present how drift, control actions and human labeling was simulated for testing.

Subsequently, section 5.4 on page 86 demonstrates how well each singular component of visual machine learning control (VMLC) performs under these conditions. Lastly, section 5.5 on page 91 inspects the full VMLC system in action, showing how the active monitoring keeps the underlying classification system stable even when encountering continuous drift or a complete shift in the sample domain.

Each experiment is done across multiple methods on both datasets. All functionality was implemented in Python using Tensorflow 2.10 [Mar+15] concerning machine learning (ML) aspects. Additional numerical operations where handled by Numpy [Har+20]. Each experiment was repeated 5 times by using cross-validation over its respective datasets [Sch93]. Using a greater number of runs was considered, but discarded due to time constraints (each run involves the complete training of both the classification and the distance models). Standard deviation over runs is denoted as $\pm$ in most tables.

5.1 Datasets Used

Experimentation was done on two different datasets: The Modified National Institute of Standards and Technology Database (MNIST) [Lec+98] data and the real-world manufacturing data gathered for the weld detection application described in section 2.1 on page 6.

5.1.1 MNIST

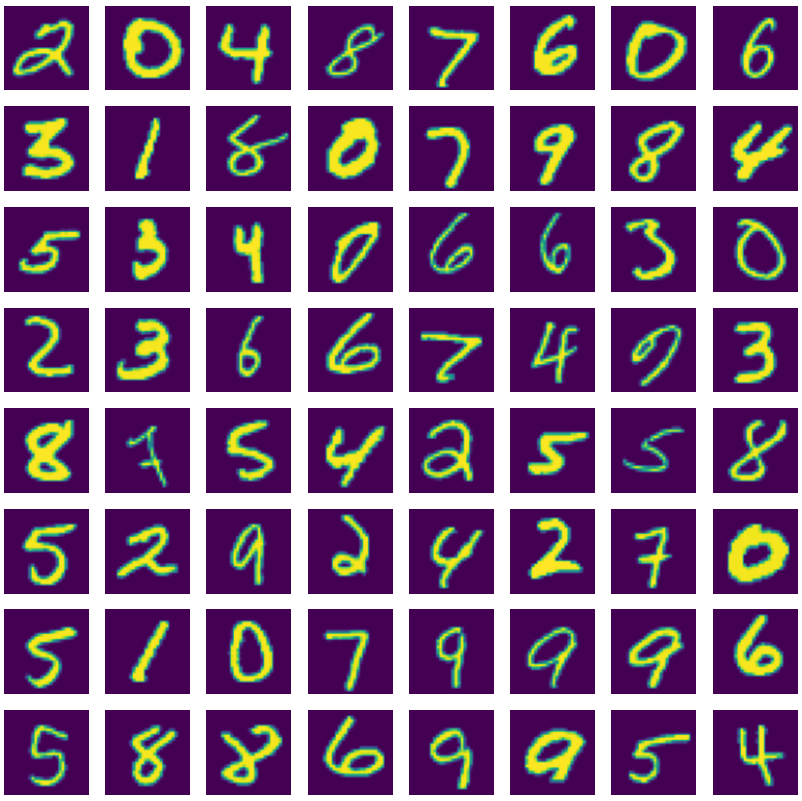


Figure 5.1: Samples from the MNIST dataset.

MNIST (illustrated in figure 5.1) contains 60,000 training ($\mathbb{S}^T$) and 10,000 test ($\mathbb{S}^P$) examples of handwritten digits. A fifth of $\mathbb{S}^T$ was further split by 5-fold cross-validation to generate the disjunct evaluation/ calibration set $\mathbb{S}^E$. Each sample consists of an image of dimension 28×28 and the class label $\hat{y} \in \{0 \dots 9\}$ representing the digits value.

This dataset is a standard benchmark used in the majority of classification model evaluations [BSI19]. Due to its closeness to actual manufacturing data

(low inter-sample variance) it is a good starting point to verify the general applicability of the presented methods.

To create a true OOD set $\mathbb{S}^{OOD}$, specific classes were omitted from the training and evaluation set, namely the digits 0, 1 and 2:

$$\begin{aligned} \mathbb{S}^{OOD} &\coloneqq \{(x,y) \in \mathbb{S}^T \cup \mathbb{S}^P : y \in \{0,1,2\}\} \\ \mathbb{S}^T &\leftarrow \{(x,y) \in \mathbb{S}^T : y \in \{3 \dots 9\}\} \\ \mathbb{S}^P &\leftarrow \{(x,y) \in \mathbb{S}^P : y \in \{3 \dots 9\}\} \end{aligned} \tag{5.1}$$

5.1.2 Weld Inspection Data

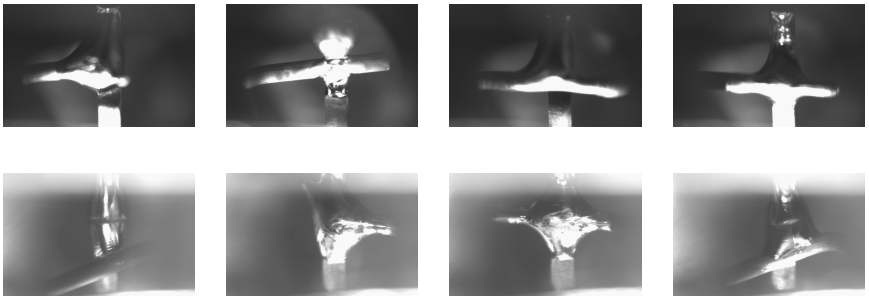
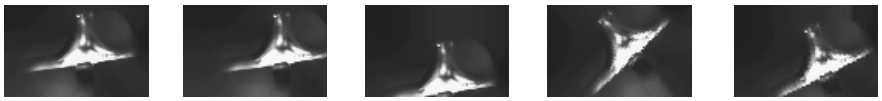


Figure 5.2: Images taken from two different camera setups. The lower row are the OOD samples.

To get experiments more in line with real world use cases, data gathered from the weld inspection introduced in section 2.1 on page 6 was used. It consist of around 14700 training samples, with $\approx 4\%$ of them containing defects of various types. Each image is in grayscale and of dimensions 188×120. Labels are a binary classification of either *Defect* or *Ok*.

To populate $\mathbb{S}^{OOD}$, 7000 additional images were taken by an additional camera setup. It is part of the same application, inspecting the second of two contacts. Its lighting properties and spatial orientation differ significantly, while still displaying the same type of object. This closely reflects a real world example of changing the setup during live production. A comparison of these images can be seen in figure 5.2.

To generate distinct sets for validation, the data was split into three disjunct sets: The training set $\mathbb{S}^T$ (containing 40% of data), an evaluation set $\mathbb{S}^E$ (using 10% of data) used for calibration and at the end of the training and the actual test data $\mathbb{S}^P$ (the remaining 50%).



θ_p	[0.,0.,0.,0.]	[**0.2**,0.,0.,0.]	[0.,**0.2**,0.,0.]	[0.,0.,**0.2**,0.]	[**0.1**,**0.1**,**0.1**]

Figure 5.3: The various kinds of artificial perturbation applied to the input images.

5.2 Simulated Factory Environment

To properly evaluate the various possible system configurations I had to run an exhaustive number of tests. This was impossible on the real application, as it was needed for production. Thus, functions that required real world interactions had to be simulated. They had to simulate the real conditions during deployment as close as possible while being easily reproducible and fast in execution.

5.2.1 Observation Function

For my experiments, it was clear that a real camera system was not available. But to test my framework, I had to be able to produce images suspect to continuous drift. To do so, I implemented the function $\textsc{Observe}(s; \theta_c, \theta_p)$ as a virtual camera, simulating various kinds of sensor drift:

Algorithm 5: Simulated observation function with artificial drift p.

$\theta'_p \leftarrow \theta_p + \theta_c$
$[p_{sx}, p_{sy}, p_{rot}, p_b] \leftarrow \theta'_p$
$s \leftarrow \textsc{Translate}(s; p_{sx} \cdot h, p_{sy} \cdot w)$
$s \leftarrow \textsc{Rotate}(s; p_{rot} \cdot 180°)$
$s \leftarrow \min(\max(s + p_b \cdot 255, 0), 255)$
return s

θ_p is split into various scalar values that define the degree of specific operations: Rotate rotates the image around its center while Translate shifts them in the respective x and y direction. h and w denominate the height and width of input image s. p_b adjusts the brightness, keeping final values between 0 and 255. Figure 5.3 visualizes the effects of θ_p on a singular image. Pixels that stem from a region outside the image (due to shift or rotation) were padded with the value of the closest pixel.

The benefits of using artificial drift are obvious: Since θ_p is known, the actual amount of drift can be measured. To do so, I defined d_R as the actual

amount of modification present in the system:

$$d_R \coloneqq |\theta_p - \theta_c| \tag{5.2}$$

5.2.2 Recalibration Function

Since θ_p and θ_c are defined as each others opposites in algorithm 5 on the previous page, an optimal calibration θ_c can always be found:

$$\theta_c = -\theta_p \Rightarrow x = \textsc{Observe}(x; \theta_c, \theta_p) \Leftrightarrow d_R = 0 \tag{5.3}$$

Thus, the self-configuration method introduced in section 4.4 on page 63 can be used to find an optimal configuration to minimize d_R. If it is not available or optimization fails, the Repair action is invoked.

5.2.3 Repair Function

In the real world, the Repair action would imply a significant effort in time and money. Everything is possible, ranging from a simple fix of an mechanical issue to an elaborate cause analysis. For the simulation, implementing a repair action is much more simple: By resetting θ_p and θ_c to $\overrightarrow{0}$, every issue in the system can be mitigated (if not stemming from OOD samples).

This assumes a **perfect repair** where an ideal state is recovered through a corrective action. In a real world scenario repair may introduce new perturbations into the system or configuration might be done sub-optimal. For keeping simplicity during experiments, I assumed this to not be the case.

5.2.4 Oracle Function

When requiring new labeled samples for a retraining, they have to be selected and sent to a domain expert. This is usually done offline, resulting in a significant delay between the first occurrence of an issue and its resolution through retraining.

Thus, I only used labeled samples from the available data during experiments. The oracle can than be implemented by simply returning the correct label for each image. In the real world assigned labels might be noisy due to miss-classification, especially if strong drift is already present. This is a general problem in machine learning and shall not be covered by this thesis. For an exhaustive survey on this topic consider [Son+22].

5.3 Distance Model

At the center of the system is the distance model M_d. To evaluate possible implementations of M_d, two possible scenarios had to be considered:

Out of Distribution: Is the detector able to correctly classify whether a sample $s \in \mathbb{S}^{OOD}$? The detector is only trained on samples from $\mathbb{S}^T$, containing no OOD data per definition. Additionally, the cutoff point d_{max} from which a sample is deemed OOD should be easily derivable.

Systematic Drift: Does the detector produces values that correlate with the actual drift present in the data? This is done by introducing artificial drift measured in d_R.

The experiments conducted in this section reproduce and extend on results already published in [Mas23a].

5.3.1 Evaluated Distance Metrics

To have a thorough comparison between different approaches, the following metrics where selected to implement the distance model M_d:

Uncertainty: The output of the classification model M, measured as described in equation (3.30) on page 40. This is the most basic approach towards model uncertainty. It serves as a baseline for any other method evaluated. [HG16]

Mahalanobis: The Mahalanobis distance described in equation (3.33) on page 41. Σ and μ where calculated by invoking the pre-pooling layer of the classification model M over every sample of $\mathbb{S}^E$. [Lee+18]

kDistance: The k-Nearest neighbor distance specified by equation (3.35) on page 42 with $k = 10$. The underlying intermediate layer is the same as for the Mahalanobis metric. [Jia+18]

PCA: The reconstruction error of a principal component analysis (PCA) as described in section 3.4.3 on page 43. It uses principal components derived from $\mathbb{S}^T$. [LLZ14]

Auto-Encoder: SSIM based reconstruction error of an Auto-Encoder, consisting of an encoder based on a 50 layer Residual Network (ResNet) and an 34-layer inverse ResNet as decoder. [ZP17]

Classification coupled Auto-Encoder (CCAE): An Auto-Encoderusing the same architecture. The weights of the encoder are shared from the the classifier, as described in section 4.3 on page 61. [Mas23b]

Mahalanalobis AE: Mahalanobis distance over the latent space L of the Auto-Encoder. [Den+18]

Mahalanalobis CCAE: Mahalanobis distance over the latent space L of the classification coupled Auto-Encoder.

This reflects a broad selection of the methods introduced in section 3.4 on page 38, having methods that are based on the classifier (Uncertainty, Mahalanobis, kDistance), based on the dataset (PCA, Auto-Encoder, Mahalanobis AE) and the new method proposed in section 4.3 on page 61 (CCAE, Mahalanobis CCAE). Each classifier was trained for 20 epochs on $\mathbb{S}^T$.

The size of the latent space L as well as the primary components of PCA where determined by the scree plot method [Cat66]. It was set to 25 for the welding data and to 10 for MNIST accordingly.

5.3.2 OOD Detection

To evaluate the OOD detection performance for each metric, every method was calibrated on $\mathbb{S}^E$ by calculating:

$$\begin{aligned} \mathcal{D} &= \{x \in \mathbb{S}^E : M_d(x; \theta_d)\} \\ d_{max} &= F_{\mathcal{D}}^{-1}(p_{ood}) \end{aligned} \tag{5.4}$$

where $F_{\mathcal{D}}^{-1}(p_{ood})$ is the inverse cumulative distribution function (CDF) of $\mathcal{D}$. A classification of x was considered correct if:

$$(x, y) \in \mathbb{S}^{OOD} \Leftrightarrow M_d(x; \theta_d) \geq d_{max} \tag{5.5}$$

Figure 5.4 on page 79 shows the receiver operating characteristic (ROC) [Han+89] for all evaluated metrics. It plots the true positive rate (TPR) against the false positive rate (FPR) for all possible threshold values d_{max}. Its axis are specified as:

$$\begin{aligned} TPR &:= \frac{|\{x \in \mathbb{S}^{OOD} : M_d(x; \theta_d) \geq d_{max}\}|}{|\mathbb{S}^{OOD}|} \\ FPR &:= \frac{|\{x \in \mathbb{S}^{P} : M_d(x; \theta_d) \geq d_{max}\}|}{|\mathbb{S}^{P}|} \end{aligned} \tag{5.6}$$

	Method	d_{max}	AUC	$\mathbb{S}^P$	$\mathbb{S}^{OOD}$
Welding	Uncertainty	$0.000_{\pm 0.00}$	$0.662_{\pm 0.13}$	$\mathbf{0.978}_{\pm 0.01}$	$0.543_{\pm 0.37}$
	kDistance	$7.449_{\pm 4.41}$	$0.939_{\pm 0.03}$	$0.968_{\pm 0.00}$	$0.024_{\pm 0.05}$
	Mahalanobis	$59.038_{\pm 16.39}$	$0.744_{\pm 0.05}$	$0.253_{\pm 0.17}$	$0.898_{\pm 0.11}$
	PCA	$0.914_{\pm 0.02}$	$0.998_{\pm 0.00}$	$0.972_{\pm 0.00}$	$0.998_{\pm 0.00}$
	Auto-Encoder	$-4.970_{\pm 0.44}$	$\mathbf{1.000}_{\pm 0.00}$	$0.757_{\pm 0.25}$	$\mathbf{1.000}_{\pm 0.00}$
	CCAE	$-3.506_{\pm 0.13}$	$0.997_{\pm 0.00}$	$0.958_{\pm 0.02}$	$\mathbf{1.000}_{\pm 0.00}$
	Mahalanobis AE	$6.836_{\pm 0.06}$	$\mathbf{1.000}_{\pm 0.00}$	$0.969_{\pm 0.01}$	$\mathbf{1.000}_{\pm 0.00}$
	Mahalanobis CCAE	$7.829_{\pm 0.21}$	$0.858_{\pm 0.03}$	$0.968_{\pm 0.00}$	$0.208_{\pm 0.07}$
MNIST	Uncertainty	$0.024_{\pm 0.05}$	$0.742_{\pm 0.05}$	$\mathbf{0.963}_{\pm 0.01}$	$0.666_{\pm 0.20}$
	kDistance	$5.575_{\pm 0.40}$	$0.349_{\pm 0.09}$	$0.902_{\pm 0.03}$	$0.002_{\pm 0.00}$
	Mahalanobis	$49.054_{\pm 8.45}$	$\mathbf{0.924}_{\pm 0.01}$	$0.953_{\pm 0.00}$	$0.651_{\pm 0.08}$
	PCA	$2.741_{\pm 0.01}$	$0.623_{\pm 0.00}$	$\mathbf{0.973}_{\pm 0.00}$	$0.181_{\pm 0.01}$
	Auto-Encoder	$-0.857_{\pm 0.01}$	$0.579_{\pm 0.06}$	$0.968_{\pm 0.00}$	$0.021_{\pm 0.01}$
	CCAE	$-0.977_{\pm 0.05}$	$0.764_{\pm 0.09}$	$0.970_{\pm 0.00}$	$0.175_{\pm 0.13}$

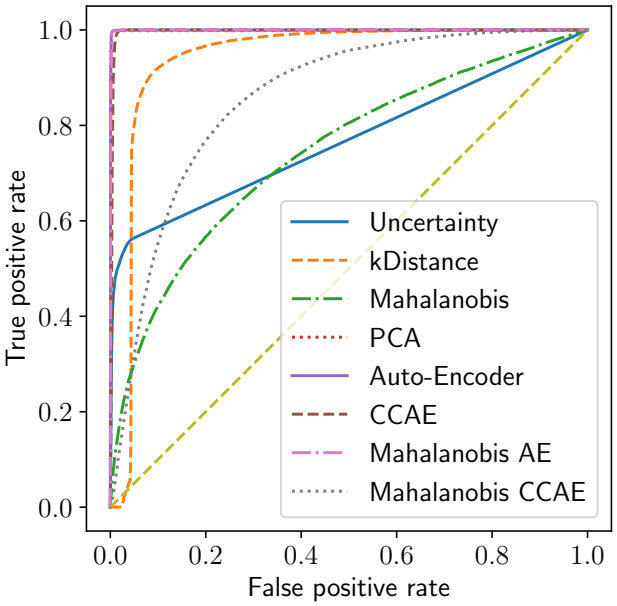
Table 5.1: Median OOD detection accuracy with $p_{ood} = 0.97$ (6 σ) and AUC scores.

It is immediately apparent that any Auto-Encoder (AE) based solution (PCA included) gives close to perfect results on the welding data, while any method based on the sole classifier has worse results. The actual per-class classification accuracy denoted in table 5.1 further solidifies this notion. It also contains the area under curve (AUC) score, which is the definite integral under the ROC curve. This is in stark contrast to the MNIST results, which seem to prefer model based approaches.

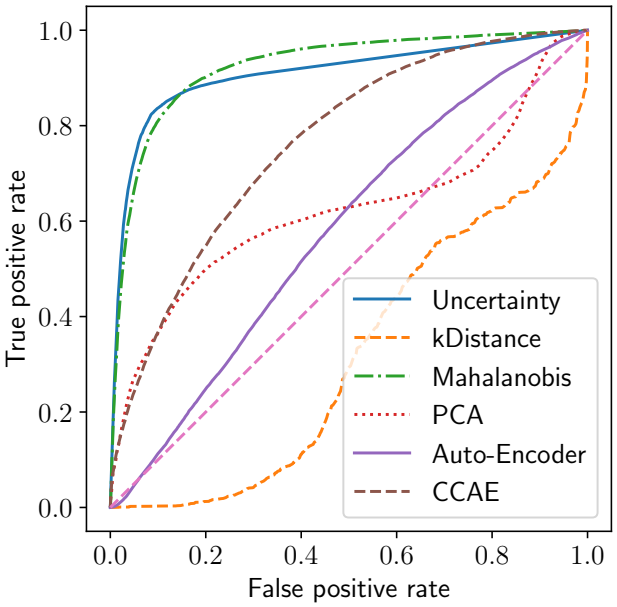
The kDistance metric presents itself as especially unreliable: It seems to correlate very heavily with the underlying class distribution and thus suffers under the apparent class imbalance, brandishing a high AUC score while incapable to properly separate the OOD samples. This might be credited to AUC not considering how probable the threshold is for an ideal ROC score [Mus20].

Plotting the distribution of values for each set exemplifies the applicability of each method, as was done in figure 5.5 on page 82. Reconstruction based methods produce easily separable distributions, with each sets values close to a normal distribution. In contrast, all metrics directly based on the classifier have strong overlaps, especially in the lower regions of d.

It is interesting how much the results of MNIST differ. Due to the



(a) Welding data



(b) MNIST

Figure 5.4: ROCs for evaluated metrics averaged over 5 cross-validation runs.

low dimensionailty of the dataset, the basic AE can easily derive a perfect representation, leading to sample distributions not properly separated (see figure 5.6 on page 83). The CCAE, while still imperfect, improves tremendously on this problem.

5.3.3 Drift Correlation

An ideal distance metric M_d should capture the average dissimilarity between the known samples of $\mathbb{S}^T$ and the currently observed samples $\mathbb{S}^P$, as suggested in section 4.2.4 on page 57. Since I induced artificial drift, its presence can be measured in d_R. An ideal distance metric should correlate with this drift, so that:

$$d_R \sim \overline{d} \tag{5.7}$$

It should be noted that this is always averaged over multiple samples, since this is the requirement to successfully apply the statistical process control (SPC) method.

Figure 5.7 on page 84 shows plots of each $\overline{d}$ metric against d_R. To generate these test samples, θ_p was set to random values drawn from $\mathcal{N}(0, 0.05)$ and was then evaluated by each metric with a sample size of $N_s = 1000$. This was done for 200 configurations of θ_p.

To measure the dependency between the two values, I used the Pearson correlation [Sed12]:

$$\rho_{d_R,\overline{d}} \coloneqq \frac{\sum(d_R - \mu_{d_R})(\overline{d} - \mu_{\overline{d}})}{\sqrt{\sum(d_R - \mu_{d_R})^2(\overline{d} - \mu_{\overline{d}})^2}} \tag{5.8}$$

with μ_{d_R} and $\mu_{\overline{d}}$ being the mean of these values over all configurations. ρ may take values from -1 to 1. 1 indicates strong linear correlation, -1 negative correlation and 0 that the two values are completely uncorrelated.

Furthermore, I postulated in equation (4.7) on page 58 that decreases in performance should result in an significant increase in $\overline{d}$. To test whether the proposed metrics are capable to do so, I formulated it as another classification problem:

$$acc(p(\mathbb{S}^P; \theta_p)) < acc(\mathbb{S}^P) \Leftrightarrow \sum_{x \in \mathbb{S}^P} d(p(x; \theta_p)) > \sum_{x \in \mathbb{S}^P} d(x) \tag{5.9}$$

This means that any θ_p that leads to a reduction in classification accuracy should be assigned a higher d value than one that does not. Figure 5.8 on

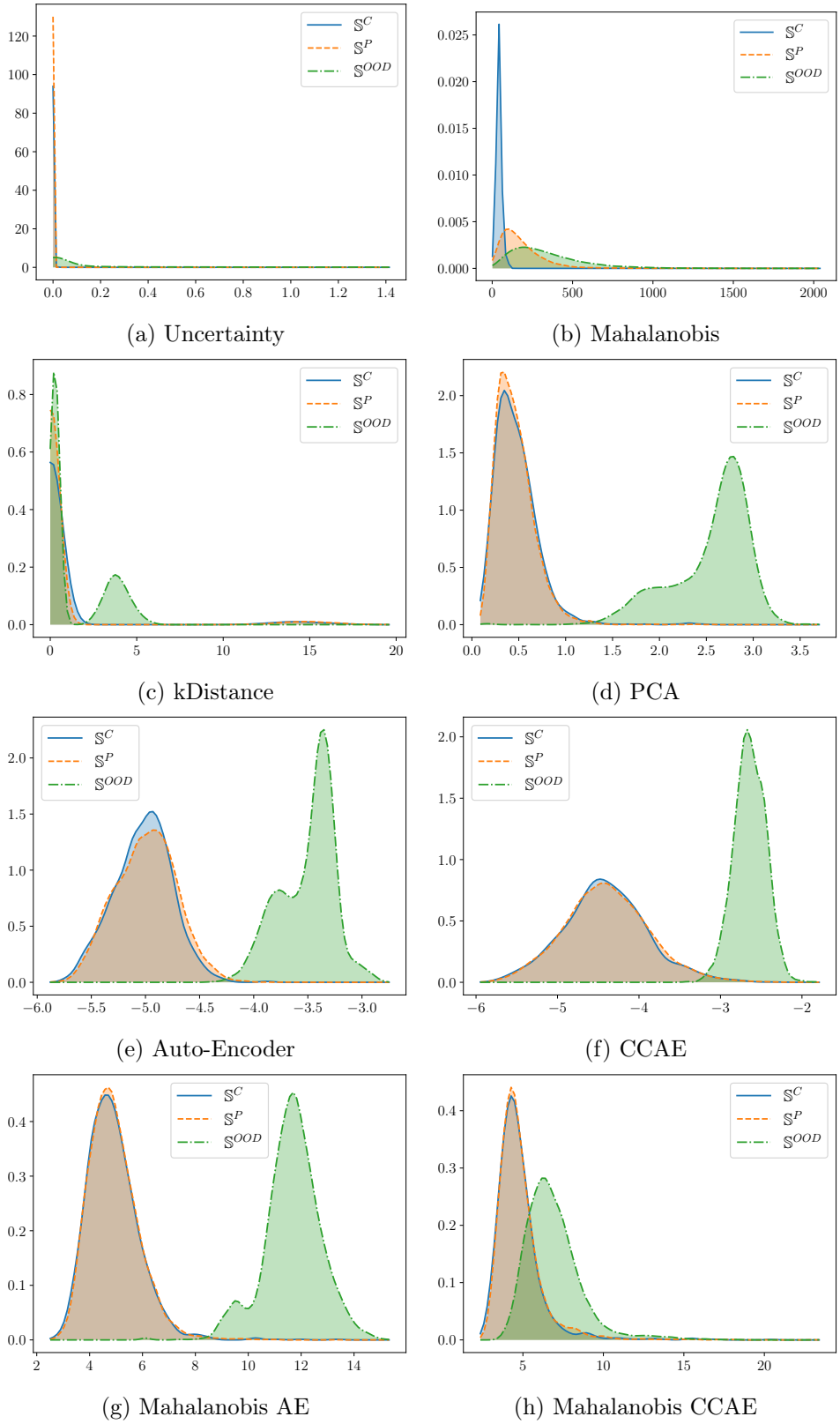
	Method	ρ	AUC on miss-classification
Welding	Uncertainty	$0.356_{\pm 0.08}$	$\mathbf{0.861}_{\pm 0.13}$
	kDistance	$0.065_{\pm 0.24}$	$0.484_{\pm 0.24}$
	Mahalanobis	$0.556_{\pm 0.06}$	$0.822_{\pm 0.09}$
	PCA	$0.610_{\pm 0.04}$	$0.839_{\pm 0.05}$
	Auto-Encoder	$0.623_{\pm 0.09}$	$0.685_{\pm 0.18}$
	CCAE	$0.670_{\pm 0.08}$	$0.674_{\pm 0.18}$
	Mahalanobis AE	$\mathbf{0.724}_{\pm 0.04}$	$0.785_{\pm 0.03}$
	Mahalanobis CCAE	$0.514_{\pm 0.02}$	$0.858_{\pm 0.08}$
MNIST	Uncertainty	$0.678_{\pm 0.04}$	$0.737_{\pm 0.37}$
	kDistance	$0.107_{\pm 0.24}$	$0.576_{\pm 0.33}$
	Mahalanobis	$\mathbf{0.702}_{\pm 0.06}$	$\mathbf{0.738}_{\pm 0.38}$
	PCA	$0.307_{\pm 0.10}$	$0.571_{\pm 0.30}$
	Auto-Encoder	$0.359_{\pm 0.07}$	$0.533_{\pm 0.28}$
	CCAE	$0.621_{\pm 0.07}$	$0.644_{\pm 0.33}$

Table 5.2: Pearson correlation ρ between $\overline{d}$ and d_R as well as AUC when detecting decreases in classification performance.

page 85 shows the corresponding ROCs. It should be noted that *acc* is always given as class-averaged accuracy to regard the high class imbalance present in industrial data.

Table 5.2 shows both correlation between simulated drift and classification accuracy. Here the differences in MNIST and real-world data are apparent once more: The AE based methods are a clear winner on industrial data. Uncertainty may produce better AUC scores, but suffers once more from its binary nature due to a very sharp threshold, unable to reflect the continuous drift.

For MNIST, Mahalanobis or the CCAE might be the weapons of choice. This might be once more attributed to the low dimensionality of the data, making reconstruction all to easy.



(a) Uncertainty

(b) Mahalanobis

(c) kDistance

(d) PCA

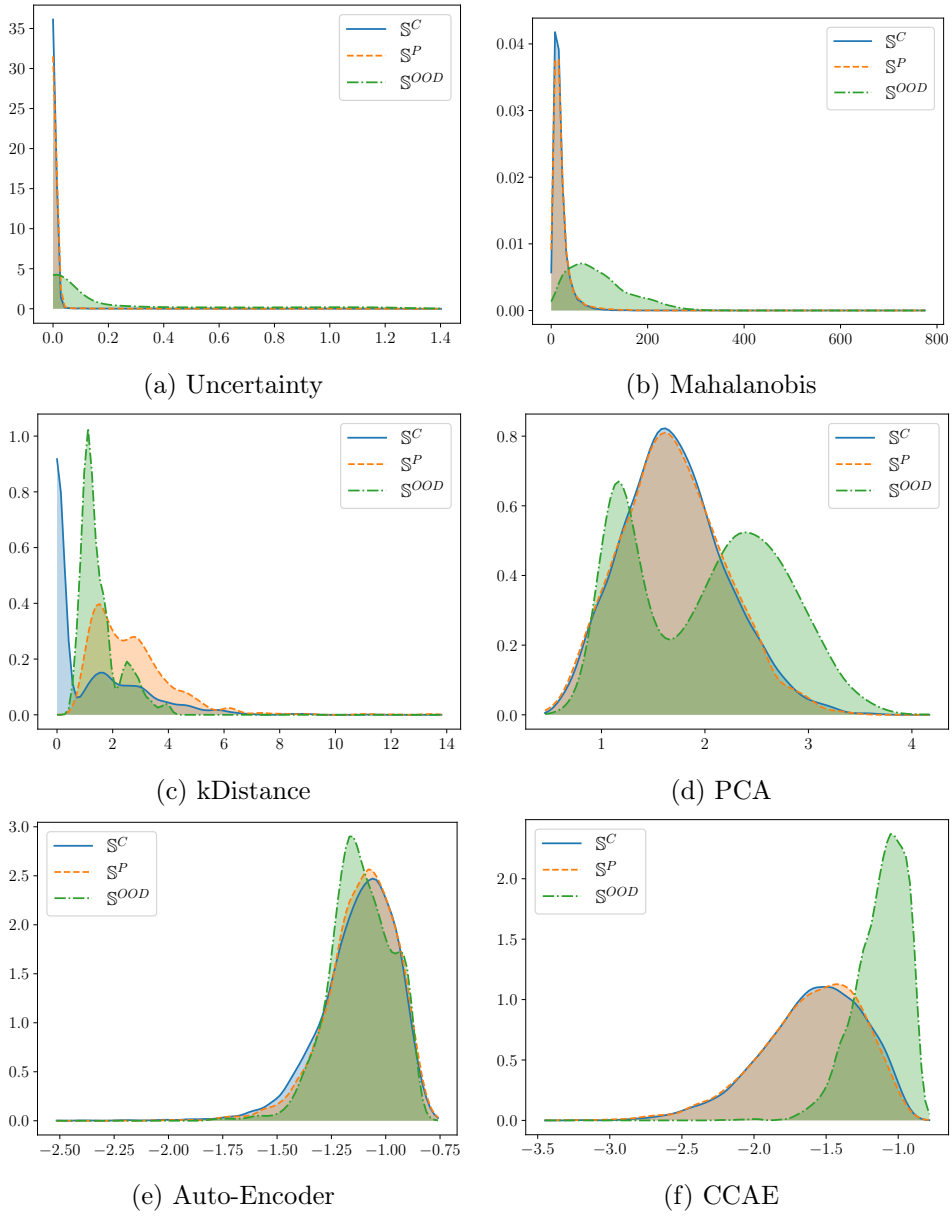
(e) Auto-Encoder

(f) CCAE

(g) Mahalanobis AE

(h) Mahalanobis CCAE

Figure 5.5: Kernel density estimates (KDEs) of the d metric distribution $\mathcal{D}$ for the various implementations of M_d on the welding dataset.



(a) Uncertainty

(b) Mahalanobis

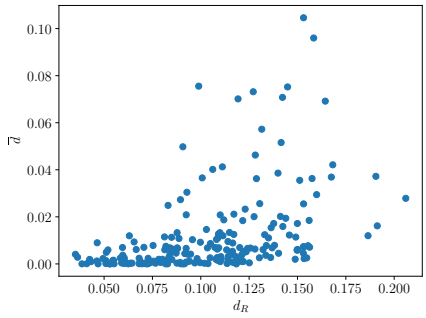
(c) kDistance

(d) PCA

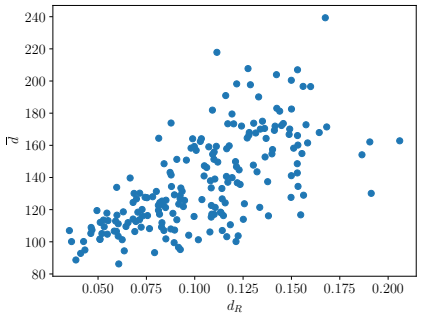
(e) Auto-Encoder

(f) CCAE

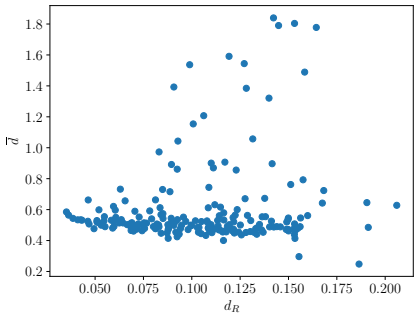
Figure 5.6: KDEs of the d metric distribution $\mathcal{D}$ for the various implementations of M_d on MNIST.



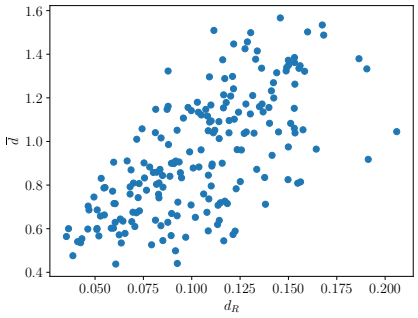
(a) Uncertainty



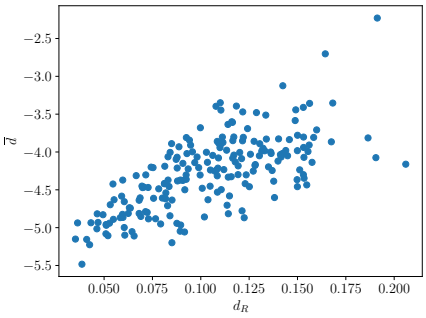
(b) Mahalanobis



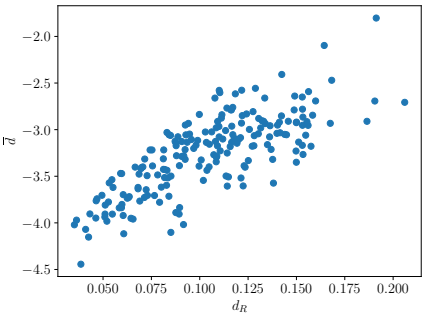
(c) kDistance



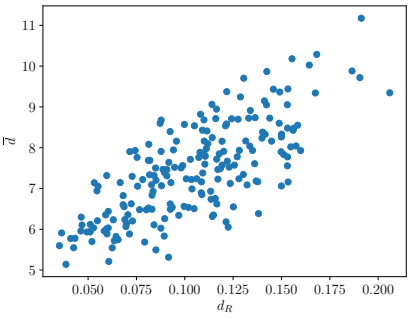
(d) PCA



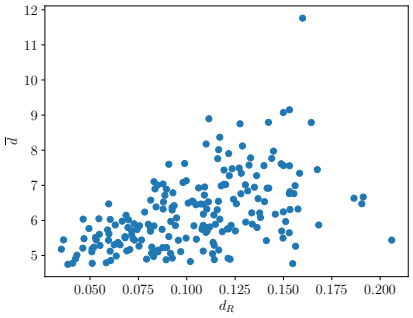
(e) Auto-Encoder



(f) CCAE

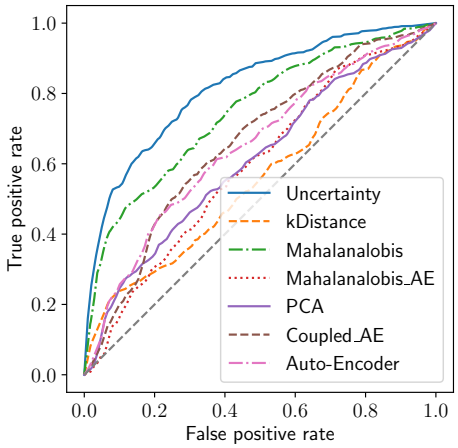


(g) Mahalanobis AE

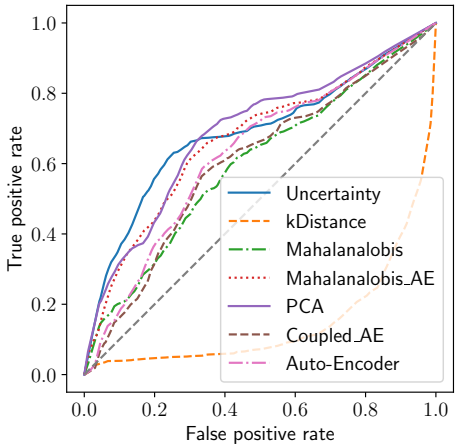


(h) Mahalanobis CCAE

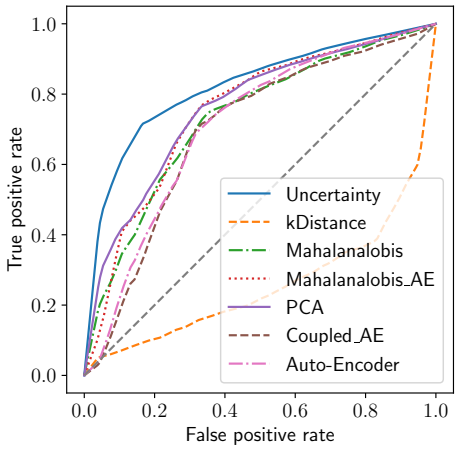
Figure 5.7: Correlation plots between the artificial drift d_R and the assigned mean d value for the welding data.



(a) Only based on mislabeled *Ok* class.



(b) Only based on mislabeled *Defect* class.



(c) Based on any loss in performance.

Figure 5.8: ROCs averaged over 5-fold cross validation for detecting degradation in classification accuracy on the welding dataset.

5.4 Component Evaluation

Based on the distance metric d, other components are derived for VMLC. Each of these components was experimented on individually to determine their general applicability. Due to time constraints, the tested implementations of M_d where restricted to Uncertainty and the CCAE.

5.4.1 Self-Calibration

	n	d_R	$p(acc_r > 0.95)$	acc_r
Welding	1	$0.164_{\pm 0.02}$	$0.205_{\pm 0.12}$	$0.758_{\pm 0.06}$
	5	$0.089_{\pm 0.01}$	$\mathbf{0.375}_{\pm 0.17}$	$0.878_{\pm 0.05}$
	20	$0.066_{\pm 0.01}$	$0.370_{\pm 0.19}$	$\mathbf{0.907}_{\pm 0.04}$
	100	$\mathbf{0.063}_{\pm 0.01}$	$0.350_{\pm 0.15}$	$0.899_{\pm 0.07}$
MNIST	1	$0.153_{\pm 0.02}$	$0.330_{\pm 0.05}$	$0.766_{\pm 0.05}$
	5	$0.114_{\pm 0.01}$	$0.560_{\pm 0.10}$	$0.886_{\pm 0.03}$
	20	$0.099_{\pm 0.01}$	$0.685_{\pm 0.12}$	$0.928_{\pm 0.02}$
	100	$\mathbf{0.089}_{\pm 0.01}$	$\mathbf{0.805}_{\pm 0.13}$	$\mathbf{0.948}_{\pm 0.03}$

Table 5.3: Results of Bayesian optimization over parameters.

To test the self-calibration method introduced in section 4.4 on page 63, I monitored if and how fast the distance metric is able to recover its initial parameters from a random configuration of θ_c. To do so, a Bayesian optimizer received the average d value over n samples observed with θ_c, based on the AE reconstruction metric. It was allowed to sample 50 times with the first 5 sample points chosen at random. The probability of improvement acquisition function was used as it yielded best results during testing.

I then looked on how close the accuracy of the classifier with the suggested best parameters was compared to an evaluation on the original images. For all 5 cross validations, I did 50 runs to a total of 250 runs per configuration. The Bayesian optimizer was allowed to sample up to 50 times.

Table 5.3 shows the results of these tests. The performance is given as:

$$acc_r \coloneqq \frac{acc(\textsc{Observe}(\mathbb{S}^P; \theta_c, \theta_p)}{acc(\mathbb{S}^P)} \tag{5.10}$$

with $\theta_p = \overrightarrow{0}$. Additionally, the average value of d_R was given after recalibration. It shows that the method comes close to recovering the perfect

	Method	OOD det.	*acc* w.o. Rejected	*acc* Retrained
Welding	Baseline (No sampling)	-	0.796	0.796
	Random sampling	0.506	-	0.896
	Uncertainty	0.808	0.890	0.918
	CCAE	**0.973**	**0.938**	**0.926**
MNIST	Baseline (No sampling)	-	-	0.696
	Random sampling	0.315	-	**0.953**
	Uncertainty	0.429	-	0.942
	CCAE	**0.674**	-	**0.953**

Table 5.4: Out-of-distribution sample detection and class balanced classification accuracy when confronted with OOD data.

parameters, but does not reconstruct them perfectly ($d_R > 0$). The amount of samples plays a huge role up to a certain point. Results on $n > 20$ are far better than with a lower batch size. Higher sizes do not seem to yield better results.

$p(acc > 0.95)$ addresses the ratio of runs that may be considered a successful recovery of parameters, with more then 95% acc_r after recalibration. It shows that only a third find the optimal parameters for the welding data. This shows that the method works for some cases, but may require further improvement for practical application.

Tests where initially also done using the Mahalanobis AE metric, but results where not satisfying. This suggests that the reconstruction error yields a smoother optimization surface that allows the Bayesian optimization to work properly.

5.4.2 Active Learning

Whether the use of $\mathbb{B}$ was really beneficial was tested by confronting the system with samples from the OOD dataset $\mathbb{S}^{OOD}$. Averaged accuracy over all samples $\mathbb{S}^P \cup \mathbb{S}^{OOD}$ was measured as a baseline. It had a contingent of 500 samples it was allowed to gather, then was retrained using $\mathbb{S}^T \cup \mathbb{B}$. Averaged accuracy was measured afterwards.

Table 5.4 shows the results of these experiments, along with the performance when simply rejecting samples. The advantage over randomly selected data is quite apparent, at least for the manufacturing data. On MNIST, there was no improvement over this method, since the 500 samples already

were sufficient to cover the missing classes in $\mathbb{S}^T$ (approximately 50 samples per missing class). It is interesting that using basic uncertainty even resulted in worse results on MNIST, since it pushed selection more towards known classes.

5.4.3 Explanations from Reconstruction

In section 4.6 on page 67 I suggested that the AE reconstruction could be used to implement explainable AI (XAI). Evaluating those claims is rather hard, since they rely on a high amount of suggestiveness: How well does one method illustrate the models decision over the other?

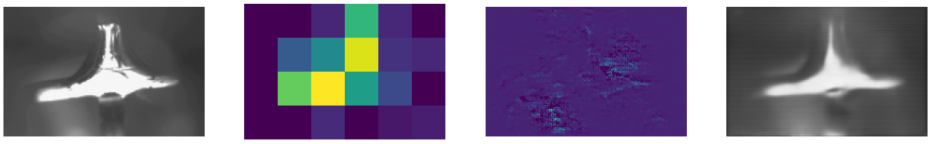
Thus, I relied on a simple side-by-side comparison between my method and two popular model agnostic ones: layer-wise relevance propagation (LRP) [Bin+16] and GradCAM [Sel+17]. LIME [RSG16] was excluded since it produced results that where rather random and not interpretable (see figure 4.10 on page 67).

Figure 5.9 on the following page illustrates the results of these experiments. There are a few issues that arise with the standard XAI methods:

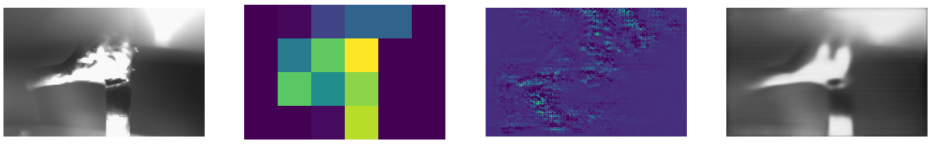
- Localization of explanations (high values on the heatmap) usually correspond with the central are of the part, which is to be expected.
- This also is true for mislabeled samples. The explanation gives no indication why the mislabeling happened on (e) and (f).
- Non-rateable samples are captured well (g).
- Samples where the classification relied on misguided assumption((d), (e) and (f)) are not distinguishable from heatmaps for *Ok* samples.

AE reconstruction handles these scenarios better. It is obvious when the classification model relied on correct assumption or not ((a), (b), (c) versus (d), (e), (f)). In my own opinion it far better enables a peak into the model than the other approaches. From these results, an operator should be able to tell whether to trust the assigned class with ease.

It becomes even more apparent on the MNIST data with omitted classes of 0 to 2: The localization methods at least show that the classification model is looking at the correct regions of the image. But the give no indication when the model is wrong in a fatal way. In contrast, the CCAE gives understandable reasoning on where the classifier saw indications for its wrong prediction (see (c), (d) and (e)).

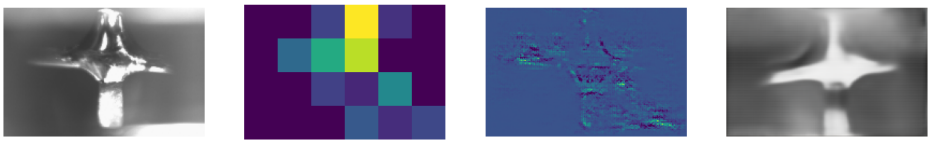


(a) Explanations where the model was correct (*Ok* sample).



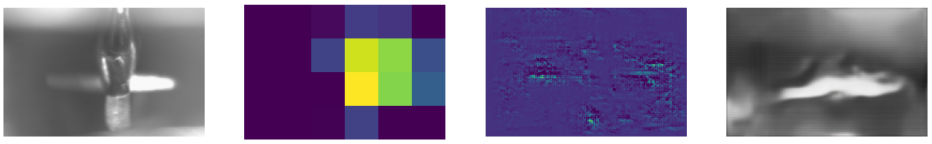
(b) Another correctly classified sample (*Ok* sample).

(c) Correctly classified *Defect* sample.



(d) OOD sample where the model was correct by accident (most samples where rated *Ok* by default, *Ok* sample).

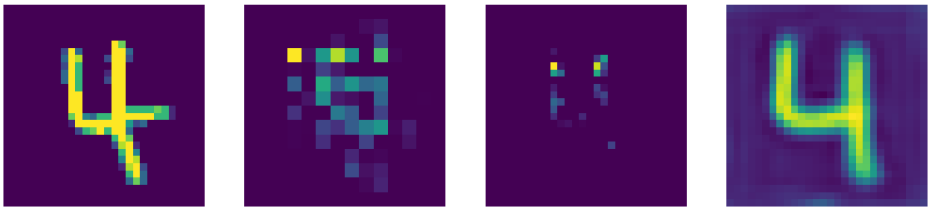
(e) Incorrectly classified sample (*Defect* sample).



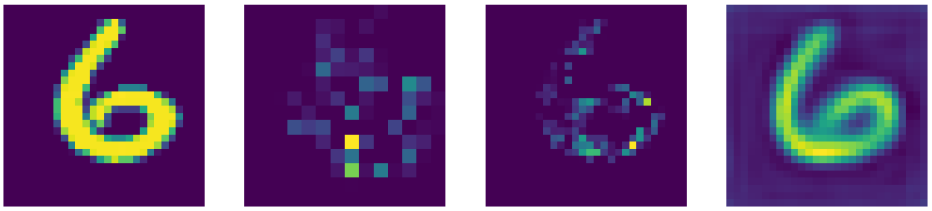
(f) Another incorrectly classified sample (*Defect* sample).

(g) Sample that is not fit for classification.

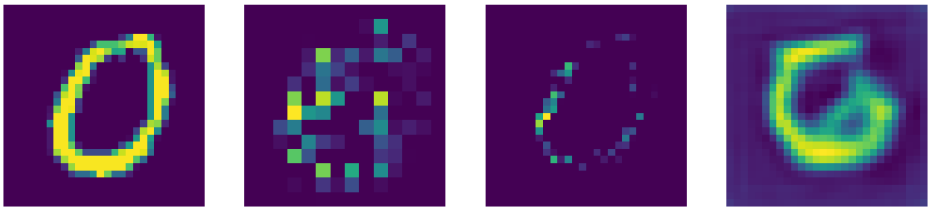
Figure 5.9: Various welding samples and their explanations. From left to right: Original image, GradCAM, LRP and AE reconstruction.



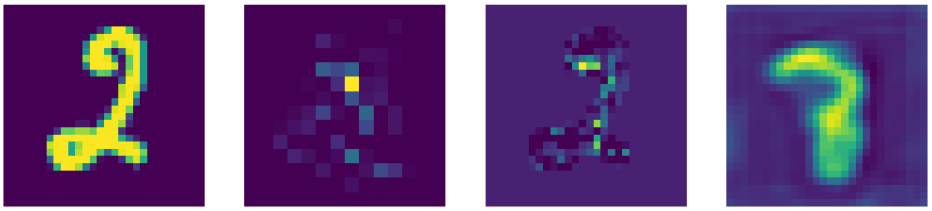
(a) Correctly classified 4.



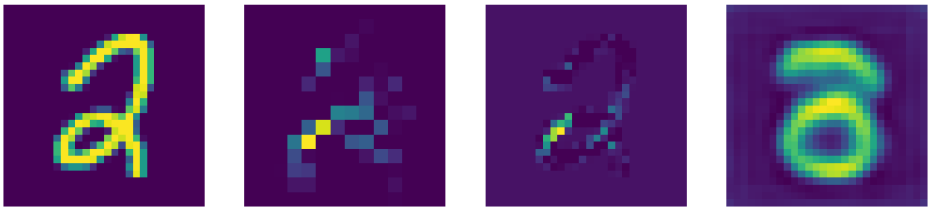
(b) Correctly classified 6.



(c) 0 that was labeled a 6.



(d) 2 that was labeled a 7.



(e) 2 that was labeled an 8.

Figure 5.10: Various samples from MNIST and their explanations. From left to right: Original image, GradCAM, LRP and AE reconstruction.

5.5 Simulated Runs

After showing that the singular components of VMLC are actually suitable for the task and data, I did an evaluation of the full system under conditions reflecting actual operations (see section 5.2 on page 74). VMLC was implemented as described in chapter 4 on page 51 using the CCAE reconstruction error as metric. The classification and distance model were both trained on the respective $\mathbb{S}^T$. Thresholds for M_d were calibrated on the evaluation set $\mathbb{S}^E$ and all tests conducted on $\mathbb{S}^P$.

The samples in $\mathbb{S}^P$ were processed in batches of size N_s (= 50 for evaluation, = 200 for illustrations). Each batch is manipulated according to the current θ_p, which is either increased by a fixed amount per batch or added with a random amount.

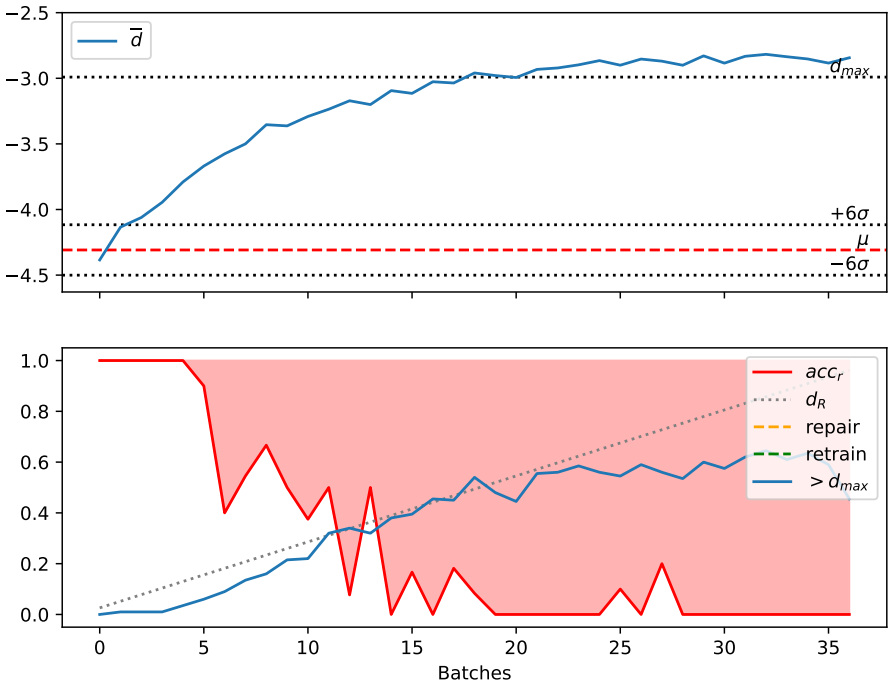
Figure 5.11 on the next page illustrates how the system performance degrades when encountering such artificial data drift. Accuracy is measured as the average accuracy over both classes (*Defect* and *Ok*). It can be seen that the d metric captures the actual drift measured by d_R. It is also interesting to see that specification limit (SL) ($\hat{=}d_{max}$) is far apart from the upper control limit (UCL), but degradation may already occur at lower levels of θ_p.

Figure 5.12 on page 93 shows how active VMLC handles the artificial drift. By resetting the parameters according to section 5.2.3 on page 75 the accuracy drop can be mitigated. The system is steered back into a stable state and operational functionality is guaranteed.

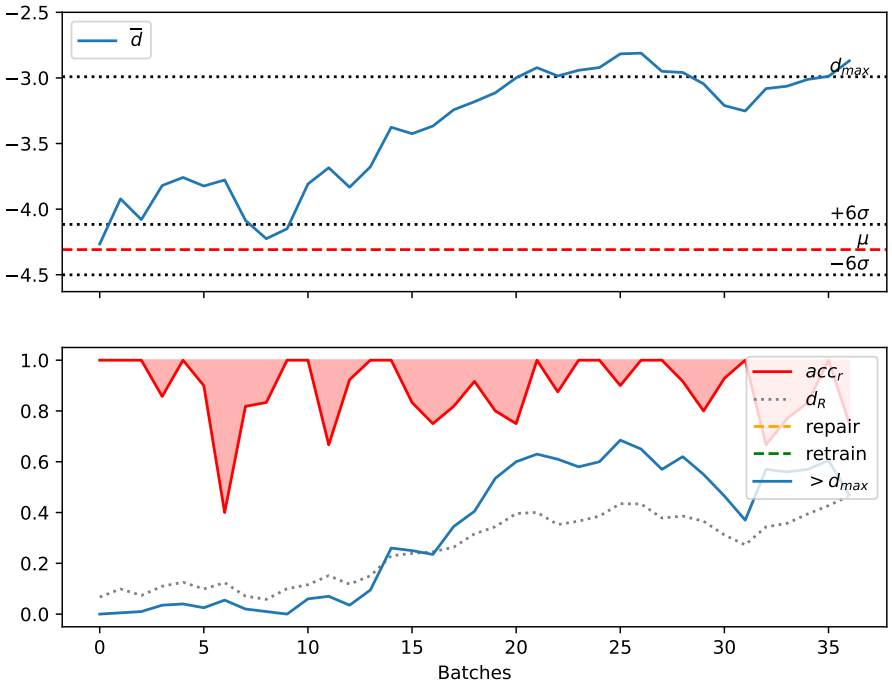
This is also true when using the re-calibration mechanic presented in section 4.4 on page 63. By optimizing over the last 5 samples, the system is able to recover the close-to ideal parameters and remain stable. It should be noted that the approach is far from optimal and does not find the optimal parameters. This usually results in further corrective actions in the next batch (see figure 5.13 on page 94).

To have a proper quantitative evaluation, I tested these scenarios multiple times using 5-fold cross validation. All tests where conducted on both datasets (MNIST and the welding data) and by applying a random increment to θ_p each batch. To have a comparison with traditional means of monitoring the classification model, VMLC was also applied using the Uncertainty metric.

Table 5.5 on page 95 displays the results of these evaluation runs (which were already published in [Mas23b]. $\overline{d_R}$ measures the average amount of drift per sample. It is shown that VMLC achieves results close to the achievable baseline when repairing after each batch (a small amount remains since drift

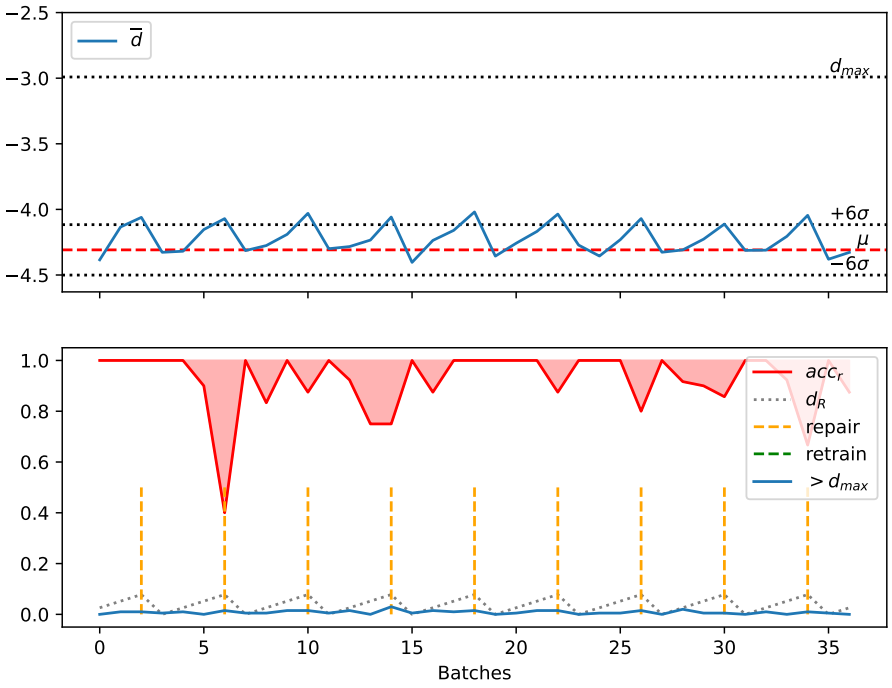


(a) Continuously increasing drift.

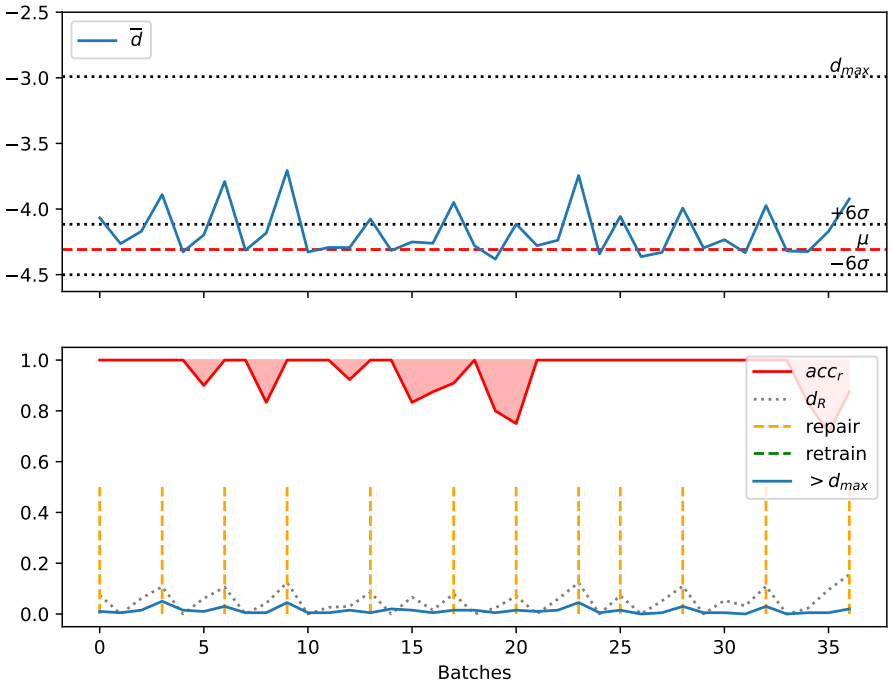


(b) Drift changing randomly.

Figure 5.11: The systems performance when not reacting on drift.

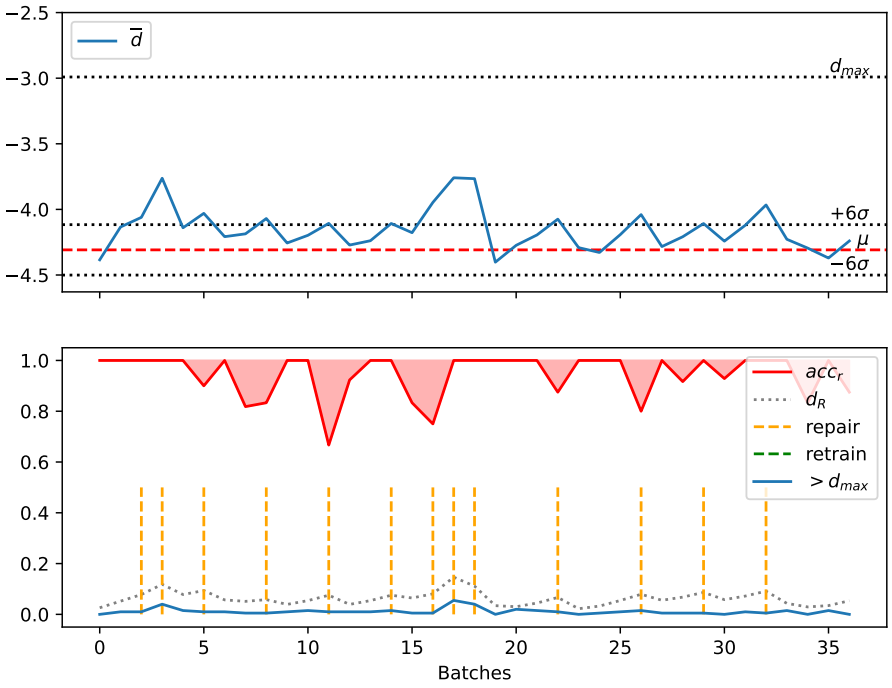


(a) Continuously increasing drift.

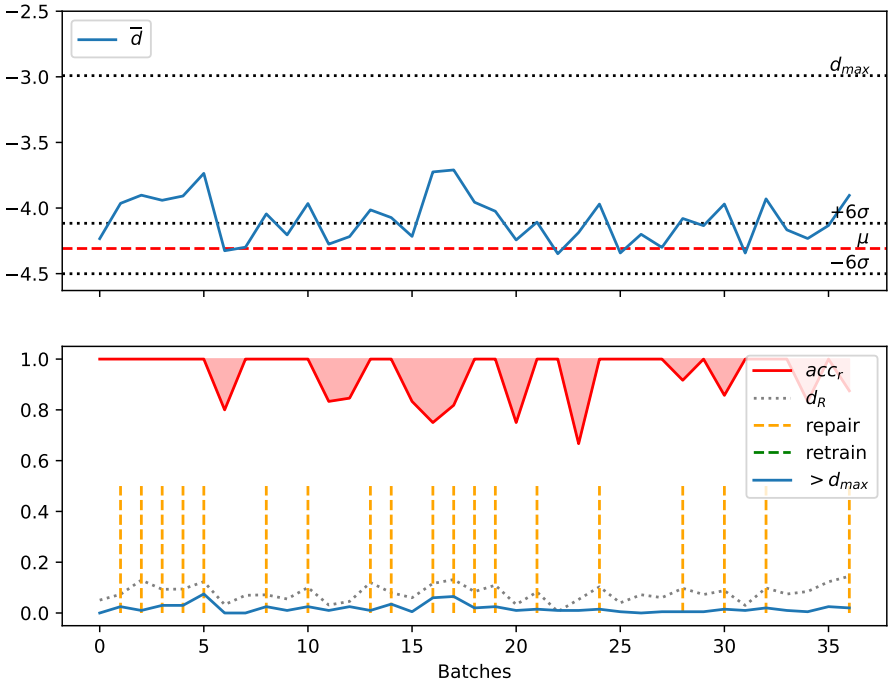


(b) Drift changing randomly.

Figure 5.12: The systems performance when applying a REPAIR action when $\overline{d} > UCL$.



(a) Continuously increasing drift.



(b) Drift changing randomly.

Figure 5.13: The systems performance when re-calibrating using Bayesian optimization when $\overline{d} > UCL$.

	Method	n_r	Accuracy			$\overline{d_R}$
			Averaged	OK	Defect	
Welding	No intervention	0	0.635	0.764	0.537	0.145
	Fixed Interval	14	0.930	0.992	0.869	0.024
	VMLC (Uncertainty)	14	**0.962**	**0.997**	**0.926**	**0.005**
	Fixed Interval	**4**	0.849	0.945	0.753	0.069
	VMLC (AE)	**4**	**0.962**	**0.997**	0.925	0.006
	Always repair	147	0.965	0.999	0.931	0.003
	Baseline (no drift)	-	0.971	0.999	0.944	0.000
MNIST	No intervention	0	0.762	-	-	0.139
	Fixed Interval	18	0.971	-	-	0.028
	VMLC (Uncertainty)	18	**0.993**	-	-	**0.005**
	Fixed Interval	**4**	0.929	-	-	0.055
	VMLC (AE)	**4**	0.982	-	-	0.027
	Always repair	138	0.993	-	-	0.003
	Baseline (no drift)	-	0.994	-	-	0.000

Table 5.5: Performance when handling artificial drift on $\mathbb{S}^P$.

is always incremented on a new batch). Performance (measured in class averaged accuracy due to class imbalance) is also maintained close to the theoretical maximum. The Auto-Encoder performs slightly worse on MNIST, but both show that the system is capable to provide a stable prediction accuracy under apparent drift.

But this alone does not guarantee a real benefit by the system. To do so, it must also keep the amount of average Repair actions triggered (measured by n_r) to a minimum. And here is where the more precise AE setup really shines: It only needs a fraction of corrective actions while maintaining equal performance and average drift compared to the uncertainty metric. Using VMLC instead of scheduled repairs of the same amount (denoted as *fixed interval*) shows that the prediction of systemic drift greatly improves the effectiveness of these interventions. Repairs are only requested when actually required, reducing maintenance cost and preventing false alarms, which are frustrating for operators and decrease their trust in the system.

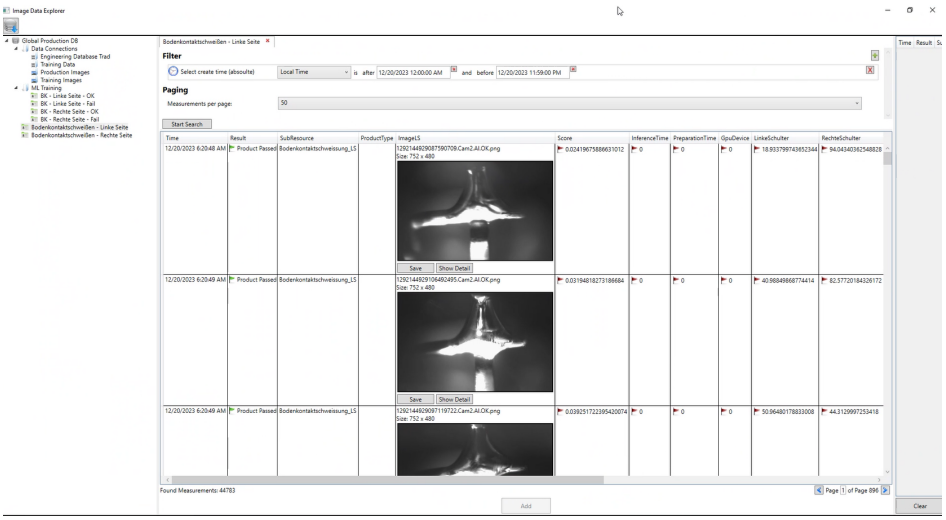


Figure 5.14: Screen-shot of the search functionality inside the *image data explorer*.

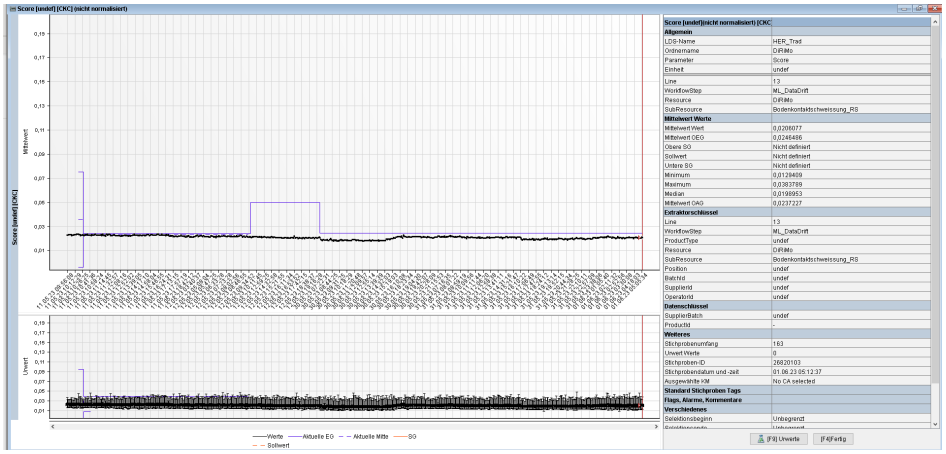


Figure 5.15: SPC chart rendered by LineWorks SPACE.

5.6 Real-World Application

Since the inspiration for this thesis sprang from a practical application, its results had to be implemented back into the manufacturing line. This meant that the components had to be integrated into the software solutions used on the shop floor.

To deploy the distance metric, it was integrated into the classification model. Through sharing most of its parameters, performance was good enough to run both models simultaneously. The distance metric was sent

together with the image over Apache NiFi [Apa], a tool to handle extract, load and transform (ETL) operations on data. Both where stored inside a database located in the cloud. This basically mimics the image buffer $\mathbb{B}$.

To access the samples for inspection and selection of labeling candidates, a visual tool named *image data explorer* was developed to search images based on their distance metric (see figure 5.14 on the previous page). It is able to sort images into new training sets and send them to an external labeling tool, namely AzureML [Tea16].

The monitoring of control limits was done by a dedicated software, LineWorks SPACE by Camline [Cam]. Figure 5.15 on the preceding page shows how the system displays the measures and shows control limits. Variance is usually very low, with no unnecessary control actions triggered to date.

Explainability as proposed in this thesis is not yet implemented, but highly requested for the future. The rather abstract concept of sample dissimilarity is hard to grasp when looking at the assigned d values alone. To improve on the understanding of this metric, the reconstructions of the current image shall be displayed on the graphical interface inside the inspection system as well as inside the image data explorer.

Adding all these improvements and control structures to the bare-bones classification system improved performance and trust significantly. It helped to transform a perceived and practical failure towards a highly reliable system over the course of a year. After multiple re-iterations of labeling and model training trough in-production active learning the classifier was able to produce stable and reliable results. Through providing SPC based on the distance metric, trust in the system could be established. This resulted in the planned roll-out to similar manufacturing lines.

5.7 Conclusion

In this chapter, I described my test on the VMLC system. By testing each component separately (OOD detection, drift detection, self-calibration, active learning and explainability), I showed that each of them is viable in their context. Proving their interplay in both a simulated and real world environment illustrates that the system is truly capable to improve operations of a deployed visual ML system. To properly argument my decisions and give suggestions on best practices for future applications, I will further discuss these results in the upcoming chapter.

Chapter 6

Discussion

> Data are just summaries of thousands of stories - tell a few of those stories to help make the data meaningful.
>
> *Dan Heath*

In this chapter, I summarize the results of the evaluation and discuss the viability of the system. I also give some insights into its real-world implementation and show how the theoretical findings translate into practical application.

6.1 Experimental Results

The experiments I conducted in chapter 5 on page 71 clearly demonstrate the viability of the visual machine learning control (VMLC) framework. An interpretation of each result was given its respective section. In this section, I want to further elaborate on some overarching topics:

6.1.1 Performance of Distance Metrics

In the experiments I have done, Auto-Encoders (AEs) emerged as the clear winners. Both using their reconstruction error or comparing their latent space proved viable. This seems to stem from their ability to recognize the extrapolation space, as formulated in section 2.2.2 on page 13.

Experiments also support the notion of equation (3.31) on page 41 when using the *Uncertainty* metric based on the model output: It is quite accurate when it detects out-of-distribution (OOD) samples, but also produces a high

rate of false negatives. Another drawback of this metric is, due to the logistic function in its final layers, that all samples are extremely pushed towards the values of 0 and 1. This results in any measurable deviation from those values to be already OOD, with no space for ambivalence in between. This makes defining useful thresholds and interpretations for its d metric rather hard.

Classification coupled Auto-Encoder (CCAE) results in a significant improvement over the plain Auto-Encoder (AE). Despite this, the Mahalanobis distance across the latent space seems to be the most potent, surpassing all other metrics in almost every category. Using it on the CCAE diminishes this advantage, which has to be expected: The encoder model almost perfectly resembles the classifier, making this version basically a re-implementation of the Mahalanobis distance on the pre-pooling layer (with an additional dense layer in between). For future implementations, I highly suggest to use the Mahalanobis distance on an generic Auto-Encoder (AE) instead of its reconstruction error.

6.1.2 Two Model System

A question that often is asked by fellow colleagues is: *Why not only use the distance metric to detect defects?* This would mean to do simple anomaly detection by training an AE on non-defect samples only (which there are plenty of), reformulating the classification as a one-class problem [SAK21]. This approach is also taken by a lot of professional computer vision (CV) providers like Cognex [Chu+20].

Figure 6.1 on the following page clearly illustrates why this approach is flawed: The two distributions are not separated enough to be precise enough for real production. This is elevated by the fact that not every deviation from the norm must be a defect per se. Quality assurance (QA) has often to determine how much anomality of a given product is deemed tolerable. Using a distinct classifier better reflects these requirements and allowed averaged detection accuracy of up to 99.7% on the full dataset for the real world application.

6.1.3 Influence of Data

The discrepancies of some results in relation to the dataset applied show that data matters. Re-evaluating methods on data outside established collections seems to be important to create more reliable and trustworthy machine learning (ML) applications. Especially since data quality is imperative to

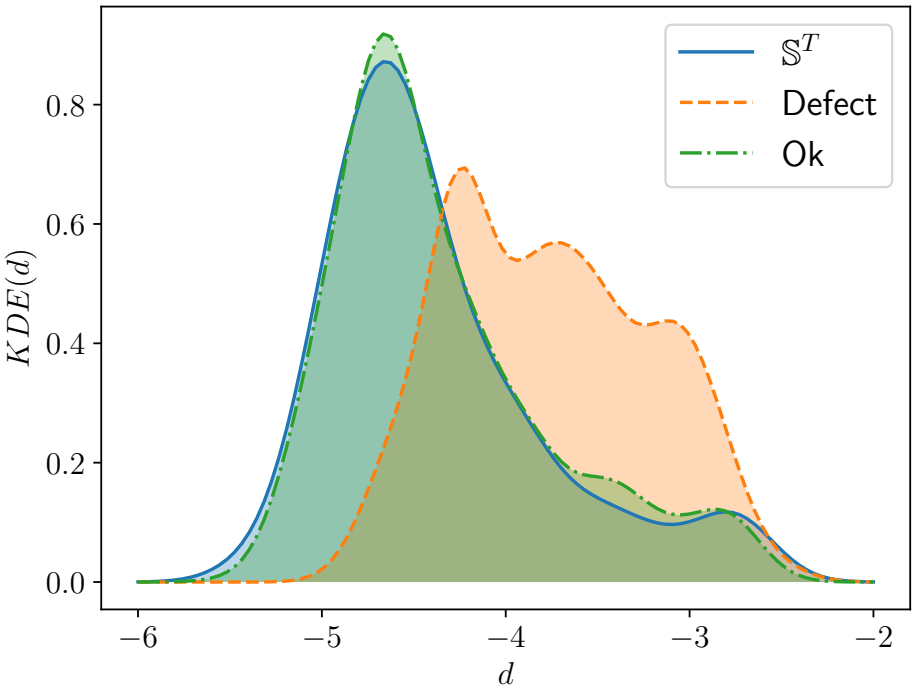


Figure 6.1: Kernel density estimate (KDE) of d values separated by classes by an auto-encoder when only trained on the *Ok* class.

high classification performance [Fen+21].

Despite deviations in distance model performance, the overall system performs well on both data sets. This shows that the approach is generally applicable and replacing model output based uncertainty with more sophisticated methods beneficial to achieve better monitoring on visual data. This is done under the constraint of low inter-sample variance, as the expressiveness of the AE reconstruction might diminish with more diverse samples. For the industrial use-case, this is sufficient. For other domains, applicability of VMLC has yet to be proven.

6.2 Fulfillment of Requirements

So does VMLC fulfill the requirements formulated in section 2.3 on page 21? Experiments have shown that, indeed, it does so:

- The system is **verifiable** through the implementation of an observer/ controller architecture. Through employing a distance model M_d any irregularities are captured, be it systematic drift, OOD data or an

unstable process. statistical process control (SPC) makes it possible for an operator to see at a single glance whether the system is operating as expected.

- The system is **repairable** by suggesting adequate control actions. It is basically a implementation of a human-automation symbiosis [Rom+16], using various actors to return itself towards a stable state. In addition, the novel idea of self-recalibration provides an automated mean to repair a subset errors caused by miss-configuration.
- The system is **optimizable** by collecting interesting samples through active learning, relying on its distance metric. It provides guidance to retrain itself, mitigating the problems of low inter-sample variance and high class imbalance.
- The system is **explainable** through the novel approach of using the reconstruction of M_d combined with CCAE as a *peek into the classification model*. It makes the d metric graspable for the average operator and illustrates misconceptions the model may have.

All these claims have been proven through the various experiments, showing that at least for the scenario provided in this thesis, VMLC is a feasible solution towards operating ML on manufacturing quality controls.

Chapter 7

Summary

> Perhaps the best test of a man's intelligence is his capacity for making a summary.
>
> *Lytton Strachey*

To finalize this thesis, I will give a quick summary over the complete work. The chapter includes an overview of the work done, insights into future work and possible improvements and ultimately highlight the contributions made through my research.

7.1 Summary

In this thesis, I formulated the research question:

How can an existing machine learning (ML) classification model used for visual quality control safely operate in an automated manufacturing environment, considering requirements like stability, explainability and maintainability?

To answer it, I introduced the domain of visual quality control for industrial manufacturing. By looking both at a real world project and theoretical considerations, I showed why machine learning (ML) operations is non-trivial for this particular field of application. Through explaining the nature of manufacturing and its data with the extrapolation problem at its core, I

formulated requirements to any application that aims to successfully operate a ML classification system in live production.

The key component to do so is the well established statistical process control (SPC) method that is applied to most measurements in manufacturing to detect unstable processes and systems. I postulated that to implement it for visual machine learning applications, a distance metric is required. it must be able to determine whether current data samples resemble the distribution of training data learned by the classificator. Subsequently, I provided different existing methods and tools to quantify this dissimilarity. Furthermore, I presented various methods to react on a detected system instability.

At the heart of this thesis is the visual machine learning control (VMLC) framework that unifies and expands these methods. It uses an observer/ controller architecture to keep the underlying classification system in a stable state. The observer is implemented through a distance model. To improve upon existing Auto-Encoders (AEs) designed for this task, I introduced classification coupled Auto-Encoder (CCAE) by basing the encoder on the classification model itself. For the controller, I suggested various control actions both done automatically and by relying on human interaction. This included novel approaches towards automated camera calibration and explainability.

The system was thoroughly evaluated on both MNIST and industrial data specifically gathered for this thesis. I was able to show that CCAE reconstruction improves upon plain AEs as distance metric, although not beating Mahalanobis distance based on AE features. Self-calibration, active learning and CCAE based explainability all proved feasible in a manufacturing context. Finally, I showed how the VMLC system significantly improves stability under a minimal amount of control actions.

In conclusion, I illustrated how the system was implemented for the original use case and showed further improvements and research directions on the topic.

7.2 Shortcomings and Outlook

In this section, I will briefly describe topics where this work might be extended or which where left out. For each i will give a short explanation why they where excluded or in what direction they should be taken.

7.2.1 Other Generative Methods

A question that arouse frequently in the scientific community was: *Why not use other generative methods like Generative Adversarial Networks (GANs) [GSS14] or transformers [Liu+23]?* Such a model can learn a typical representation of an image under some constraints and then use this to compare with the current sample. As these kind of models are seen as more powerful due to their recent ground-breaking applications in zero-shot learning or text-to-image generation [Rad+21].

And herein lies the problem. Results that are *more realistic* to a human often fare worse in terms of similarity towards the actual ground truth (the input image). This is described as the perception-distortion trade-off in recent research [BM17]. Humans lay more emphasis on *details*, which are basically artifacts to the reconstruction.

Say we have a process t that removes information from an input x:

$$t(x) = x' : x \in \mathbb{R}^n, x' \in \mathbb{R}^m, n > m \tag{7.1}$$

Then there exists a multitude of inverse transformations $\mathbb{T} \coloneqq \{t^{-1} : t(x) = x'\}$. AEs searches for the optimal decoder to its known encoder t:

$$\min_{t^{-1} \in \mathbb{T}} \sum_{x \in \mathbb{S}^T} \mathrm{E}(x, t^{-1}(t(x))) \tag{7.2}$$

Other methods select the optimal t^{-1} by some other criteria, for example the output of the discriminator in adversarial learning or the current patch for transformers. This might result in a generated x' that looks better to the human observer, but is less accurate in its reconstruction.

7.2.2 Other Methods to Handle OOD

In this thesis, I evaluated a broad selection of approaches towards measuring classification uncertainty. Of course there exist many other approaches that have not been considered for various reasons.

Many of these methods focus on modifying the model itself to generate more reliable classification results. This is done through regularization [Hua+22; Kam+20] or by adding synthesized samples to the training [Du+22; Ver+19]. These techniques are not considered in the scope of this thesis, since they modify the classification model and break the boundary between the inference process and the observer-controller architecture.

The same applies to approaches from the field of generalized open-set recognition [GHC20b], where out-of-distribution (OOD) samples are used to condition the model (*e.g.* [Moh+20]). Additionally, due to the low variance issue of industrial data (see section 2.2.3 on page 18), it is questionable if this kind of data is even available for industrial applications.

In terms of actual architecture used for the AE distance metric, other implementations are possible. Variational Auto-Encoders (VAEs) [KW14] are often suggested, where the latent dimension L is replaced by a normal distribution $L_{VAE} \sim \mathcal{N}(\mu, \sigma)$. This allows to generate more probable samples from the latent space, as the variance is stored in an additional parameter vector. Reconstruction is done on a randomly drawn sample from this latent distribution and gradient calculation is done through the reparameterization trick [KSW15]. Extensions for anomaly detection exist, for example by measuring the propability of reconstruction [AC15]. As the modeling of the latent space in a variational manner did not result in any improvements in [Mas23a], it was omitted from the evaluation.

7.2.3 Real-World Limitations of Self-Recalibration

It has been shown in section 5.4.1 on page 86 that the idea of an self-calibrating system through a distance model is working. Bayesian optimization proveides good solutions with reasonable amounts of required samples. There are still other methods of derivative-free optimization methods, either from the class of genetic algorithms (GAs) [Hol92] like covariance matrix adaptation evolution strategy (CMA-ES) [Han06] and particle swarm optimization [KE95] or sampling based methods like simulated annealing [BT93] or using subgradients [NB01]. Evaluating and comparing these methods on the recalibration problem could be an interesting research question in itself.

For this work, I considered it sufficient to prove the general applicability of the approach. Taking the step from its theoretical functionality towards a practical implementation should be considered with a grain of salt: Although the Bayesian optimization needs quite few samples, in a physical environment each sampling might take a long time: The sensor has to setup its parameters for each sample point to take an image. This implies multiple constraints on the method:

- Not every parameter might be managed by the sensor, *e.g.* an external light source or a mechanical positioning device like a robot. Thus only a subset of parameters might be automatically configured, resulting in

no confergence during calibration.

- Not every systemic drift might be fixed by recalibration. If there is smear on the lens or problems in a preceding process, stability might not be achievable.

- The drift value is only stable when averaged over multiple samples. The experiments have shown that more than one image has to be taken to reliably find the best parameters. This might require the machine to be specifically built for this scenario or an optimization method has to be found that can optimize over subsequent samples. Another possibility would be to use specific samples where the achievable d value is known (called *golden samples* in manufacturing) to calibrate the sensor more easily.

7.2.4 Active Learning Selection

The field of active learning was only briefly explored in section 4.5 on page 66. The general selection of interesting samples was solely based on the distance metric. There exists a multitude of methods to further improve selection strategy like TypiClust [HDW22], Learning Loss [YK19] or P4tal [Yi+22] (see [Ren+21a] for a general overview).

These methods are usually evaluated on common benchmark datasets like CIFAR-10 [Kri09a]. It is possible that these active learning strategies behave different on manufacturing data. Evaluating against such data would have expanded the scope of this work too much, but is a possibility for further research. For the evaluation done for this thesis, I considered showing the applicability of active learning based on the distance metric as sufficient. The goal was to prove its benefit, will finding the best method to do so is a target for potential future research.

7.2.5 Segmentation Problems

Single or multi-label classification is one major application for computer vision inside quality control. Another is semantic segmentation, basically classification on a per-pixel basis. In the manufacturing context, this usually means to detect and measure impurities, particles or damages based on an image (see figure 7.1 on the next page for an example from another ML project done during this thesis). State of the art ML models have greatly

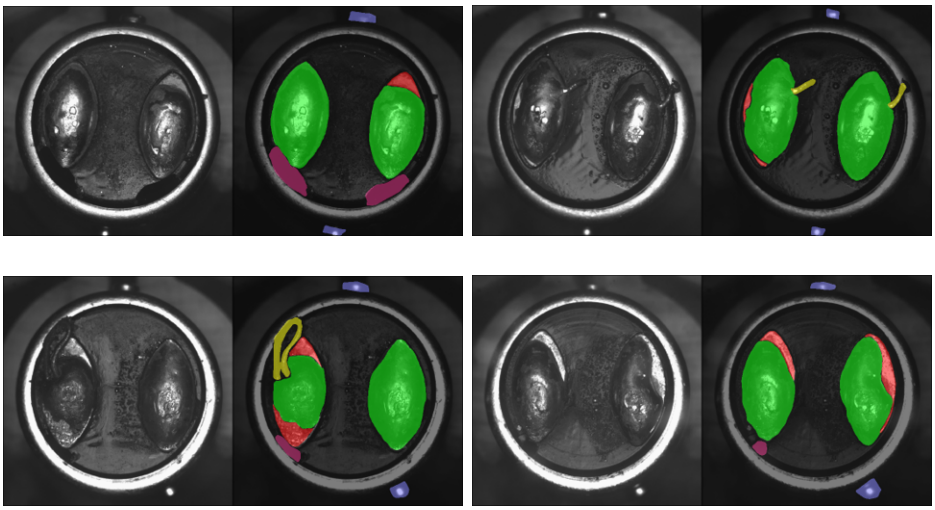


Figure 7.1: Images and segmentation ground-truth for soldering quality control. It segments good (green) from bad soldering (red), as well as break-outs in the socket (purple), uncut wires (yellow) and socket pins (blue).

improved the performance of these applications [Sov+20; Jin+21] and allow new applications, for example guiding an autonomous robot [TAG19].

Due to the general design of the AE based distance metric it can be easily adapted to contemporary segmentation architectures like DeepLab [Che+17] or UNets [RFB15]. Figure 7.2 on the following page illustrates the reconstruction on the afore-mentioned task, allowing to detect miss-alignments or disturbances in the imaging system.

Due to time constraints, the applicability of VMLC could not be tested on this kind of data. This would be a great starting point for future research and help the system to be even more adaptable to various manufacturing tasks.

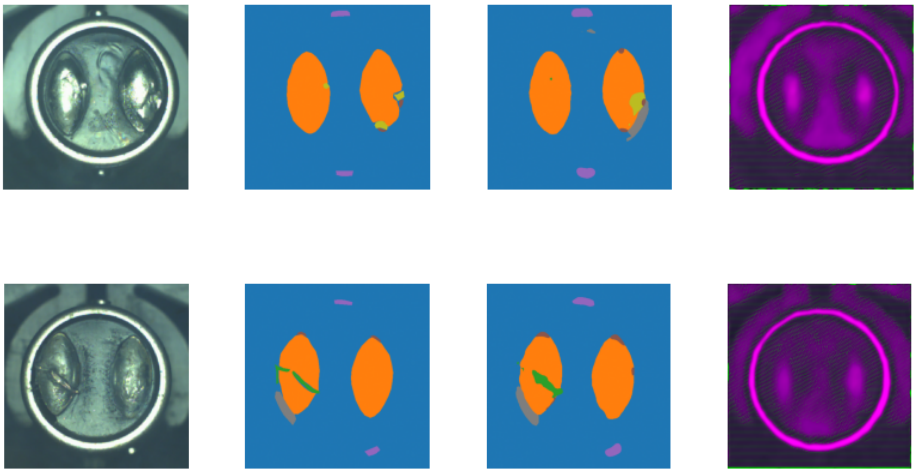


Figure 7.2: Input image, ground truth, assigned labels by the classification network and reconstruction on the solder data.

7.3 Contributions

Despite all these possible improvements and open questions, this work provides a feasible solution for productive application of visual machine learning, tested both theoretically and in a practical application. Through this thesis, I made the following contributions:

- I presented VMLC as a novel framework to create a stable and universal that handles all requirements of a dynamic manufacturing environment.
- I suggested a viable implementation for SPC on visual data, which did not exist before.
- I introduced CCAE, an improvement over the naive approach of using AEs to monitor data dissimilarities.
- I created a new classification dataset with a total of over 21000 labeled samples. Many existing methods where re-evaluated on this data with partially novel results.
- I implemented a new method to automatically recover calibration parameters for a camera system based on previously recorded data.
- I designed a new CCAE-based method for explainable AI (XAI) which might have huge potential for other applications.

Index

Bibliography

[Aba+19] Davide Abati et al. "Latent Space Autoregression for Novelty Detection." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[Abd+19] Vahdat Abdelzad et al. *Detecting Out-of-Distribution Inputs in Deep Neural Networks Using an Early-Layer Output*. 2019. arXiv: 1910.10307 [cs.LG].

[AC15] Jinwon An and Sungzoon Cho. "Variational autoencoder based anomaly detection using reconstruction probability." In: *Special Lecture on IE* 2.1 (2015), pp. 1–18.

[Agu+23] Diana Laura Aguilar et al. "Towards an Interpretable Autoencoder: A Decision-Tree-Based Autoencoder and its Application in Anomaly Detection." In: *IEEE Transactions on Dependable and Secure Computing* 20.2 (2023), pp. 1048–1059. DOI: 10.1109/TDSC.2022.3148331.

[Ama93] Shun-ichi Amari. "Backpropagation and stochastic gradient descent method." In: *Neurocomputing* 5.4-5 (1993), pp. 185–196.

[Apa] Apache. *Apache Nifi main page*. URL: www.nifi.apache.org/ (visited on 12/30/2023).

[Ari+20] Jorge F Arinez et al. "Artificial intelligence in advanced manufacturing: Current status and future outlook." In: *Journal of Manufacturing Science and Engineering* 142.11 (2020), p. 110804.

[Aru+17] Kai Arulkumaran et al. "Deep reinforcement learning: A brief survey." In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38.

[Ayt+18] Caglar Aytekin et al. "Clustering and unsupervised anomaly detection with l 2 normalized deep auto-encoder representations." In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–6.

[Ba-+23] Abdulrahman H. Ba-Alawi et al. "Development of transparent high-frequency soft sensor of total nitrogen and total phosphorus concentrations in rivers using stacked convolutional auto-encoder and explainable AI." In: *Journal of Water Process Engineering* 53 (2023), p. 103661. ISSN: 2214-7144. DOI: https://doi.org/10.1016/j.jwpe.2023.103661. URL: https://www.sciencedirect.com/science/article/pii/S2214714423001782.

[BBC20] Paola Baruchelli, Francesco Botto, and A Cimatti. "Overview on maturity of AI innovations in manufacturing." In: *EIT Digit* (2020).

[Ben+06] Yoshua Bengio et al. "Greedy layer-wise training of deep networks." In: *Advances in neural information processing systems* 19 (2006).

[Ber+18] Paul Bergmann et al. "Improving unsupervised defect segmentation by applying structural similarity to autoencoders." In: *arXiv preprint arXiv:1807.02011* (2018).

[BG20] Hauke Brunken and Clemens Gühmann. "Deep learning self-calibration from planes." In: *Twelfth International Conference on Machine Vision (ICMV 2019)*. Vol. 11433. SPIE. 2020, pp. 980–990.

[Bin+16] Alexander Binder et al. "Layer-wise relevance propagation for deep neural network architectures." In: *Information science and applications (ICISA) 2016*. Springer. 2016, pp. 913–922.

[BKS23] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics." In: *Machine Learning* 112.10 (2023), pp. 3713–3747.

[Bla16] Eugene H Blackstone. "Sufficient data." In: *The Journal of Thoracic and Cardiovascular Surgery* 152.5 (2016), pp. 1235–1236.

[BM17] Yochai Blau and Tomer Michaeli. "The Perception-Distortion Tradeoff." In: *CoRR* abs/1711.06077 (2017). arXiv: 1711.06077. URL: http://arxiv.org/abs/1711.06077.

[Bog+18] Oleksandr Bogdan et al. "DeepCalib: A deep learning approach for automatic intrinsic calibration of wide field-of-view cameras." In: *Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production*. 2018, pp. 1–10.

[Bom+21] Rishi Bommasani et al. "On the opportunities and risks of foundation models." In: *arXiv preprint arXiv:2108.07258* (2021).

[Bro+20] Tom Brown et al. "Language Models are Few-Shot Learners." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[BS18] Roberto Souto Maior Barros and Silas Garrido T. Carvalho Santos. "A large-scale comparison of concept drift detectors." In: *Information Sciences* 451-452 (2018), pp. 348–370. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2018.04.014.

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[BSI19] Alejandro Baldominos, Yago Saez, and Pedro Isasi. "A survey of handwritten character recognition with mnist and emnist." In: *Applied Sciences* 9.15 (2019), p. 3169.

[BT93] Dimitris Bertsimas and John Tsitsiklis. "Simulated annealing." In: *Statistical science* 8.1 (1993), pp. 10–15.

[BW21] Gerrit van den Burg and Chris Williams. "On memorization in probabilistic deep generative models." In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 27916–27928.

[BZ23] Nick Bryan-kinns and Bingyuan Zhang. *Exploring Variational Auto-Encoder Architectures, Configurations, and Datasets for Generative Music Explainable AI*. Oct. 2023. arXiv: 2311.08336 [cs.AI].

[CAL94] David Cohn, Les Atlas, and Richard Ladner. "Improving generalization with active learning." In: *Machine learning* 15 (1994), pp. 201–221.

[Cam] Camline. *Camline LineWork SPACE product page*. URL: `www.camline.com/products/lineworks/lineworks-space/` (visited on 12/30/2023).

[Cas96] Kenneth R Castleman. *Digital image processing*. Prentice Hall Press, 1996.

[Cat66] Raymond B. Cattell. "The Scree Test For The Number Of Factors." In: *Multivariate Behavioral Research* 1.2 (1966), pp. 245–276.

[CC19] Raghavendra Chalapathy and Sanjay Chawla. "Deep learning for anomaly detection: A survey." In: *arXiv preprint arXiv:1901.03407* (2019).

[CFE22] Chelsea Chandler, Peter W Foltz, and Brita Elvevåg. "Improving the applicability of AI for psychiatric applications through human-in-the-loop methodologies." In: *Schizophrenia Bulletin* 48.5 (2022), pp. 949–957.

[Cha+20] Chun-Hao Chao et al. "Self-supervised deep learning for fisheye image rectification." In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 2248–2252.

[Che+17] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[Che+20] Hao Chen et al. "Learned resolution scaling powered gaming-as-a-service at scale." In: *IEEE Transactions on Multimedia* 23 (2020), pp. 584–596.

[Che+23] Yibo Chen et al. "Iterative Prompt Refinement for Mining Gene Relationships from ChatGPT." In: *bioRxiv* (2023).

[Cho+20a] Michał Choraś et al. "Machine Learning – The Results Are Not the only Thing that Matters! What About Security, Explainability and Fairness?" In: *Computational Science – ICCS 2020*. Ed. by Valeria V. Krzhizhanovskaya et al. Cham: Springer International Publishing, 2020, pp. 615–628. ISBN: 978-3-030-50423-6.

[Cho+20b] Jun Kang Chow et al. "Anomaly detection of defects on concrete structures with the convolutional autoencoder." In: *Advanced Engineering Informatics* 45 (2020), p. 101105.

[Chu+20] Hwehee Chung et al. "Unsupervised anomaly detection using style distillation." In: *IEEE Access* 8 (2020), pp. 221494–221502.

[CLX21] Ben Chen, Yuyao Liu, and Caihua Xiong. "Automatic checkerboard detection for robust camera calibration." In: *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2021, pp. 1–6.

[CT65] James W Cooley and John W Tukey. "An algorithm for the machine calculation of complex Fourier series." In: *Mathematics of computation* 19.90 (1965), pp. 297–301.

[Cui+19] Yin Cui et al. "Class-balanced loss based on effective number of samples." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9268–9277.

[CW17] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks." In: *2017 ieee symposium on security and privacy (sp)*. IEEE. 2017, pp. 39–57.

[DBM97] Edward L DeLosh, Jerome R Busemeyer, and Mark A McDaniel. "Extrapolation: the sine qua non for abstraction in function learning." In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 23.4 (1997), p. 968.

[Dem52] W. Edwards Deming. *Elementary principles of the statistical control of quality; a series of lectures*. 1952.

[Den+18] Taylor Denouden et al. *Improving Reconstruction Autoencoder Out-of-distribution Detection with Mahalanobis Distance*. 2018. arXiv: `1812.02765 [cs.LG]`.

[DF18] Alexander Duda and Udo Frese. "Accurate Detection and Localization of Checkerboard Corners for Calibration." In: *BMVC*. 2018, p. 126.

[Din+13] Shifei Ding et al. "Evolutionary artificial neural networks: a review." In: *Artificial Intelligence Review* 39 (2013), pp. 251–260.

[Din+20] Han Ding et al. "State of AI-based monitoring in smart manufacturing and introduction to focused section." In: *IEEE/ASME Transactions on Mechatronics* 25.5 (2020), pp. 2143–2154.

[DJ12] Steven B. Damelin and Willard Miller Jr. *The Mathematics of Signal Processing*. USA: Cambridge University Press, 2012. ISBN: 1107601045.

[DK16] Pavel Nikolaevich Druzhkov and Valentina Dmitrievna Kustikova. "A survey of deep learning methods and software tools for image classification and object detection." In: *Pattern Recognition and Image Analysis* 26 (2016), pp. 9–15.

[Doc] Tensorflow API Docs. *Tensorflow SSIM API documentation*. URL: `www.tensorflow.org/api_docs/python/tf/image/ssim` (visited on 12/30/2023).

[Don+14] Chao Dong et al. "Learning a deep convolutional network for image super-resolution." In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13*. Springer. 2014, pp. 184–199.

[Du+22] Xuefeng Du et al. "VOS: Learning What You Don't Know by Virtual Outlier Synthesis." In: *CoRR* abs/2202.01197 (2022). arXiv: `2202.01197`. URL: `https://arxiv.org/abs/2202.01197`.

[DY99] Yu-Hong Dai and Yaxiang Yuan. "A nonlinear conjugate gradient method with a strong global convergence property." In: *SIAM Journal on optimization* 10.1 (1999), pp. 177–182.

[Edd+14] Douglas Eddy et al. "A robust surrogate modeling approach for material selection in sustainable design of products." In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Vol. 46285. American Society of Mechanical Engineers. 2014, V01AT02A091.

[Fei+17] Reuben Feinman et al. "Detecting adversarial samples from artifacts." In: *arXiv preprint arXiv:1703.00410* (2017).

[Fen+21] Giuseppe Fenza et al. "Data set quality in Machine Learning: Consistency measure based on Group Decision Making." In: *Applied Soft Computing* 106 (2021), p. 107366. ISSN: 1568-4946. DOI: `https://doi.org/10.1016/j.asoc.2021.107366`. URL:

https://www.sciencedirect.com/science/article/pii/S1568494621002891.

[For96] Cipriano Forza. "Work organization in lean production and traditional plants: what are the differences?" In: *International Journal of Operations & Production Management* (1996).

[Fra18] Peter I. Frazier. *A Tutorial on Bayesian Optimization*. 2018. arXiv: 1807.02811 [stat.ML].

[Gaw+23] Jakob Gawlikowski et al. "A survey of uncertainty in deep neural networks." In: *Artificial Intelligence Review* 56.Suppl 1 (2023), pp. 1513–1589.

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[GHC20a] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. "Recent advances in open set recognition: A survey." In: *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2020), pp. 3614–3631.

[GHC20b] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. "Recent advances in open set recognition: A survey." In: *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2020), pp. 3614–3631.

[GJP95] Federico Girosi, Michael Jones, and Tomaso Poggio. "Regularization theory and neural networks architectures." In: *Neural computation* 7.2 (1995), pp. 219–269.

[GK20] Hossein Gholamalinezhad and Hossein Khosravi. "Pooling methods in deep neural networks, a review." In: *arXiv preprint arXiv:2009.07485* (2020).

[GL21] Alexander Gillert and Uwe Freiherr von Lukas. "Towards Combined Open Set Recognition and Out-of-Distribution Detection for Fine-grained Classification." In: *VISIGRAPP (5: VISAPP)*. 2021, pp. 225–233.

[Gon+19] Dong Gong et al. "Memorizing Normality to Detect Anomaly: Memory-Augmented Deep Autoencoder for Unsupervised Anomaly Detection." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[Goo15] Ian J Goodfellow. "Shlens J Szegedy C Explaining and harnessing adversarial examples." In: *3rd International Conference on Learning Representations*. 2015.

[GP22] Michael Gall and Federico Pigni. "Taking DevOps mainstream: a critical review and conceptual framework." In: *European Journal of Information Systems* 31.5 (2022), pp. 548–567.

[GSM21] Tuomas Granlund, Vlad Stirbu, and Tommi Mikkonen. "Towards regulatory-compliant MLOps: Oravizio's journey from a machine learning experiment to a deployed certified medical product." In: *SN computer Science* 2.5 (2021), p. 342.

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." In: *arXiv preprint arXiv:1412.6572* (2014).

[Guo+17] Chuan Guo et al. "On calibration of modern neural networks." In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.

[Han+89] John A Hanley et al. "Receiver operating characteristic (ROC) methodology: the state of the art." In: *Crit Rev Diagn Imaging* 29.3 (1989), pp. 307–335.

[Han06] Nikolaus Hansen. "The CMA evolution strategy: a comparing review." In: *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms* (2006), pp. 75–102.

[Har+20] Charles R. Harris et al. "Array programming with NumPy." In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[HDW22] Guy Hacohen, Avihu Dekel, and Daphna Weinshall. "Active Learning on a Budget: Opposite Strategies Suit High and Low Budgets." In: *ArXiv* abs/2202.02794 (2022).

[He+15] Kaiming He et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[He+16] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. 2016. DOI: 10.48550/ARXIV.1603.05027. URL: https://arxiv.org/abs/1603.05027.

[Hei92] Renée Panozzo Heilbronner. "The autocorrelation function: an image processing tool for fabric analysis." In: *Tectonophysics* 212.3-4 (1992), pp. 351–370.

[Hen+19] Dan Hendrycks et al. "Using self-supervised learning can improve model robustness and uncertainty." In: *Advances in neural information processing systems* 32 (2019).

[HG16] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks." In: *International Conference on Learning Representations*. 2016.

[HHR04] Miguel A Hernàn, Sonia Hernàndez-Diaz, and James M Robins. "A structural approach to selection bias." In: *Epidemiology* (2004), pp. 615–625.

[HHZ22] Pengfei Hu, Chunming He, and Yiming Zhu. "The Scheme and System Architecture of Product Quality Inspection based on Software-Defined Edge Intelligent Controller (SD-EIC) in Industrial Internet of Things." In: *2022 IEEE International Conference on Smart Internet of Things (SmartIoT)*. 2022, pp. 58–64. DOI: 10.1109/SmartIoT55134.2022.00019.

[HKH16] Daniel Herrera, C Juho Kannala, and Janne Heikkila. "Forget the checkerboard: Practical self-calibration using a planar scene." In: *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2016, pp. 1–9.

[HLS13] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. John Wiley & Sons, 2013.

[Hol92] John H Holland. "Genetic algorithms." In: *Scientific american* 267.1 (1992), pp. 66–73.

[HSS12] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." In: *Cited on* 14.8 (2012), p. 2.

[Hua+22] Wenjian Huang et al. "Density-driven Regularization for Out-of-distribution Detection." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 887–900.

[HZ93] Geoffrey E Hinton and Richard Zemel. "Autoencoders, minimum description length and Helmholtz free energy." In: *Advances in neural information processing systems* 6 (1993).

[IS15] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: *International Conference on Machine Learning*. 2015.

[Ji+23] Ziwei Ji et al. "Survey of hallucination in natural language generation." In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38.

[Jia+18] Heinrich Jiang et al. "To Trust Or Not To Trust A Classifier." In: *NeurIPS*. 2018, pp. 5546–5557.

[Jin+21] Zeqing Jin et al. "Precise localization and semantic segmentation detection of printing conditions in fused filament fabrication technologies using machine learning." In: *Additive Manufacturing* 37 (2021), p. 101696.

[Jor+17] Roberto Jordaney et al. "Transcend: Detecting Concept Drift in Malware Classification Models." In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 625–642. ISBN: 978-1-931971-40-9. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/jordaney.

[Jou+21] Nicolas Jourdan et al. "Machine learning for intelligent maintenance and quality control: A review of existing datasets and corresponding use cases." In: *ESSN: 2701-6277* (2021).

[Kam+20] Mohammad Mahdi Kamani et al. "Targeted data-driven regularization for out-of-distribution generalization." In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 882–891.

[KBC22] Steffen Kinkel, Marco Baumgartner, and Enrica Cherubini. "Prerequisites for the adoption of AI technologies in manufacturing – Evidence from a worldwide sample of manufacturing companies." In: *Technovation* 110 (2022), p. 102375. ISSN: 0166-4972. DOI: https://doi.org/10.1016/j.technovation.2021.102375. URL: https://www.sciencedirect.com/science/article/pii/S0166497221001565.

[KC21] Jin-Young Kim and Sung-Bae Cho. "Explainable prediction of electric energy demand using a deep autoencoder with interpretable latent space." In: *Expert Systems with Applications* 186 (2021), p. 115842.

[KE95] James Kennedy and Russell Eberhart. "Particle swarm optimization." In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[Kim+21a] Jungsuk Kim et al. "Printed circuit board defect detection using deep learning via a skip-connected convolutional autoencoder." In: *Sensors* 21.15 (2021), p. 4968.

[Kim+21b] Kwanyoung Kim et al. "Task-aware variational adversarial active learning." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8166–8175.

[Kin+14] Durk P Kingma et al. "Semi-supervised learning with deep generative models." In: *Advances in neural information processing systems* 27 (2014).

[Kin14] Diederik P Kingma. "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (2014).

[KK17] Sang Gyu Kwak and Jong Hae Kim. "Central limit theorem: the cornerstone of modern statistics." In: *Korean journal of anesthesiology* 70.2 (2017), pp. 144–156.

[Kri09a] Alex Krizhevsky. "Learning Multiple Layers of Features from Tiny Images." In: (2009), pp. 32–33. URL: `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[Kri09b] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems* 25 (2012).

[KSW15] Durk P Kingma, Tim Salimans, and Max Welling. "Variational Dropout and the Local Reparameterization Trick." In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015.

[KT19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of NAACL-HLT*. 2019, pp. 4171–4186.

[KW14] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes." In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014. arXiv: `http://arxiv.org/abs/1312.6114v10 [stat.ML]`.

[Lan+10] Jyhshyan Lan et al. "An investigation of neural network classifiers with unequal misclassification costs and group sizes." In: *Decision Support Systems* 48.4 (2010), pp. 582–591.

[LeC+89] Yann LeCun et al. "Handwritten digit recognition with a back-propagation network." In: *Advances in neural information processing systems* 2 (1989).

[Lec+98] Y. Lecun et al. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: `10.1109/5.726791`.

[Lee+18] Kimin Lee et al. "A simple unified framework for detecting out-of-distribution samples and adversarial attacks." In: *Advances in neural information processing systems* 31 (2018).

[Li+19] Haidong Li et al. "Research on overfitting of deep learning." In: *2019 15th international conference on computational intelligence and security (CIS)*. IEEE. 2019, pp. 78–81.

[Li+23] Bo Li et al. "Trustworthy AI: From principles to practices." In: *ACM Computing Surveys* 55.9 (2023), pp. 1–46.

[Lin+19] Hui Lin et al. "Automated defect inspection of LED chip using deep convolutional neural network." In: *Journal of Intelligent Manufacturing* 30.6 (2019), pp. 2525–2534.

[Liu+21] Xiao Liu et al. "Self-supervised learning: Generative or contrastive." In: *IEEE transactions on knowledge and data engineering* 35.1 (2021), pp. 857–876.

[Liu+23] Yang Liu et al. "A Survey of Visual Transformers." In: *IEEE Transactions on Neural Networks and Learning Systems* (2023), pp. 1–21. DOI: `10.1109/TNNLS.2022.3227717`.

[LL17] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 4768–4777.

[LLZ14] Di Li, Lie-Quan Liang, and Wu-Jie Zhang. "Defect inspection and extraction of the mobile phone cover glass based on the principal components analysis." In: *The International Journal of Advanced Manufacturing Technology* 73.9 (2014), pp. 1605–1614.

[LN89] Dong C Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization." In: *Mathematical programming* 45.1 (1989), pp. 503–528.

[LPK20] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. "Explainable ai: A review of machine learning interpretability methods." In: *Entropy* 23.1 (2020), p. 18.

[LW19] Mingyang Li and Zequn Wang. "Surrogate model uncertainty quantification for reliability-based design optimization." In: *Reliability Engineering & System Safety* 192 (2019), p. 106432.

[Mad+18] Aleksander Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks." In: *International Conference on Learning Representations*. 2018.

[Mal+03] Elias N Malamas et al. "A survey on industrial vision systems, applications and tools." In: *Image and vision computing* 21.2 (2003), pp. 171–188.

[Mar+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[Mar63] Donald W Marquardt. "An algorithm for least-squares estimation of nonlinear parameters." In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[Meh+21] Ninareh Mehrabi et al. "A survey on bias and fairness in machine learning." In: *ACM computing surveys (CSUR)* 54.6 (2021), pp. 1–35.

[MHN+13] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. "Rectifier nonlinearities improve neural network acoustic models." In: *Proc. icml*. Vol. 30. 1. Atlanta, GA. 2013, p. 3.

[Mii+24] Risto Miikkulainen et al. "Evolving deep neural networks." In: *Artificial intelligence in the age of neural networks and brain computing*. Elsevier, 2024, pp. 269–287.

[Mit97] Tom M Mitchell. *Machine learning*. Vol. 1. 9. McGraw-hill New York, 1997.

[Moh+20] Sina Mohseni et al. "Self-supervised learning for generalizable out-of-distribution detection." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5216–5223.

[Mor+12] Jose G Moreno-Torres et al. "A unifying view on dataset shift in classification." In: *Pattern recognition* 45.1 (2012), pp. 521–530.

[MPP18] Sofia Morar, Elena Pelican, and Dorin-Mircea Popovici. "Filter Application on Facial Features in a Smartphone App." In: *RoCHI*. 2018, pp. 5–11.

[Mus20] John Muschelli III. "ROC and AUC with a binary predictor: a potentially misleading metric." In: *Journal of classification* 37.3 (2020), pp. 696–708.

[MW17] Amit Mandelbaum and Daphna Weinshall. "Distance-based confidence score for neural network classifiers." In: *arXiv preprint arXiv:1709.09844* (2017).

[MWC11] Fadel Megahed, William Woodall, and Jaime Camelio. "A Review and Perspective on Control Charting with Image Data." In: *Journal of Quality Technology* 43 (Apr. 2011), pp. 83–98. DOI: 10.1080/00224065.2011.11917848.

[Nas+19] Ali Bou Nassif et al. "Speech recognition using deep neural networks: A systematic review." In: *IEEE access* 7 (2019), pp. 19143–19165.

[NB01] Angelia Nedic and Dimitri P Bertsekas. "Incremental subgradient methods for nondifferentiable optimization." In: *SIAM Journal on Optimization* 12.1 (2001), pp. 109–138.

[NH10] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines." In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[OMK20] Daniel W Otter, Julian R Medina, and Jugal K Kalita. "A survey of the usages of deep learning for natural language processing." In: *IEEE transactions on neural networks and learning systems* 32.2 (2020), pp. 604–624.

[OSS21] Yarema Okhrin, Wolfgang Schmid, and Ivan Semeniuk. "New Approaches for Monitoring Image Data." In: *IEEE Transactions on Image Processing* 30 (2021), pp. 921–933. DOI: 10.1109/TIP.2020.3039389.

[PAD18] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. "Generative probabilistic novelty detection with adversarial autoencoders." In: *Advances in neural information processing systems* 31 (2018).

[Pan+21] Guansong Pang et al. "Deep learning for anomaly detection: A review." In: *ACM computing surveys (CSUR)* 54.2 (2021), pp. 1–38.

[Phi17] Jeff M Phillips. "Coresets and sketches." In: *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017, pp. 1269–1288.

[Pla+23] Siby Jose Plathottam et al. "A review of artificial intelligence applications in manufacturing operations." In: *Journal of Advanced Manufacturing and Processing* 5 (May 2023). DOI: 10.1002/amp2.10159.

[PM18] Nicolas Papernot and Patrick McDaniel. "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning." In: *arXiv preprint arXiv:1803.04765* (2018).

[Pu+16] Yunchen Pu et al. "Variational autoencoder for deep learning of images, labels and captions." In: *Advances in neural information processing systems* 29 (2016), pp. 2352–2360.

[QWJ17] Hao Qin, Hongwei Wang, and Aylmer L Johnson. "A RFBSE model for capturing engineers' useful knowledge and experience during the design process." In: *Robotics and Computer-Integrated Manufacturing* 44 (2017), pp. 30–43.

[Rad+21] Alec Radford et al. "Learning transferable visual models from natural language supervision." In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[Ram+21] Aditya Ramesh et al. "Zero-shot text-to-image generation." In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8821–8831.

[RCF19] Wenhao Ruan, Yifan Chen, and Babak Forouraghi. "On Development of Data Science and Machine Learning Applications in Databricks." In: *World Congress on Services*. Springer. 2019, pp. 78–91.

[Rec+19] Benjamin Recht et al. "Do ImageNet Classifiers Generalize to ImageNet?" In: *International Conference on Machine Learning*. 2019, pp. 5389–5400.

[Rei+16] Denis Moreira dos Reis et al. "Fast unsupervised online drift detection using incremental kolmogorov-smirnov test." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1545–1554.

[Ren+21a] Pengzhen Ren et al. *A Survey of Deep Active Learning*. 2021. arXiv: `2009.00236 [cs.LG]`.

[Ren+21b] Pengzhen Ren et al. "A survey of deep active learning." In: *ACM computing surveys (CSUR)* 54.9 (2021), pp. 1–40.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[Rom+16] David Romero et al. "The operator 4.0: Human cyber-physical systems & adaptive automation towards human-automation symbiosis work systems." In: *Advances in Production Management Systems. Initiatives for a Sustainable World: IFIP WG 5.7 International Conference, APMS 2016, Iguassu Falls, Brazil, September 3-7, 2016, Revised Selected Papers*. Springer. 2016, pp. 677–686.

[Ros58] Frank Rosenblatt. "The Perceptron: A Probabilistic Model For Information Storage And Organization in the Brain." In: *Psychological Review* 65 (1958), (6): 386–408.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "" Why should i trust you?" Explaining the predictions of any classifier." In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[Rub+19] David Rubín de Celis Leal et al. "Efficient bayesian function optimization of evolving material manufacturing processes." In: *ACS omega* 4.24 (2019), pp. 20571–20578.

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006, pp. I–XVIII, 1–248. ISBN: 026218253X.

[SAK21] Naeem Seliya, Azadeh Abdollah Zadeh, and Taghi M Khoshgoftaar. "A literature review on one-class classification and its potential applications in big data." In: *Journal of Big Data* 8.1 (2021), pp. 1–31.

[Sal+17] Tim Salimans et al. "Evolution strategies as a scalable alternative to reinforcement learning." In: *arXiv preprint arXiv:1703.03864* (2017).

[SC19] Sue E. Stankus and Krystel K. Castillo-Villar. "An Improved multivariate generalised likelihood ratio control chart for the monitoring of point clouds from 3D laser scanners." In: *International Journal of Production Research* 57.8 (2019), pp. 2344–2355. DOI: 10.1080/00207543.2018.1518600. eprint: https://doi.org/10.1080/00207543.2018.1518600. URL: https://doi.org/10.1080/00207543.2018.1518600.

[Sch+10] Hartmut Schmeck et al. "Adaptivity and Self-Organization in Organic Computing Systems." In: *ACM Trans. Auton. Adapt. Syst.* 5.3 (Sept. 2010). ISSN: 1556-4665. DOI: 10.1145/1837909.1837911. URL: https://doi.org/10.1145/1837909.1837911.

[Sch+17] Nick Schneider et al. "RegNet: Multimodal sensor registration using deep neural networks." In: *2017 IEEE intelligent vehicles symposium (IV)*. IEEE. 2017, pp. 1803–1810.

[Sch15] Jürgen Schmidhuber. "Deep learning in neural networks: An overview." In: *Neural networks* 61 (2015), pp. 85–117.

[Sch18] Jacqueline Schmitt. "Quality inspection in manufacturing by predictive machine learning models." In: *Technical report for Collaborative Research Center SFB 876 Providing Information by Resource-Constrained Data Analysis* (2018), p. 93.

[Sch93] Cullen Schaffer. "Selecting a classification method by cross-validation." In: *Machine learning* 13 (1993), pp. 135–143.

[Sed12] Philip Sedgwick. "Pearson's correlation coefficient." In: *Bmj* 345 (2012).

[Sel+17] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization." In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[Set09] Burr Settles. "Active Learning Literature Survey." In: 2009.

[SF23] Gesina Schwalbe and Bettina Finzel. "A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts." In: *Data Mining and Knowledge Discovery* (2023), pp. 1–59.

[SG18] Lewis Smith and Yarin Gal. "Understanding measures of uncertainty for adversarial example detection." In: *arXiv preprint arXiv:1803.08533* (2018).

[Sin+21] Akhil Singh et al. "Neural style transfer: A critical review." In: *IEEE Access* 9 (2021), pp. 131583–131613.

[SK17] Tegjyot Singh Sethi and Mehmed Kantardzic. "On the reliable detection of concept drift from streaming unlabeled data." In: *Expert Systems with Applications* 82 (2017), pp. 77–99.

[SK19] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning." In: *Journal of big data* 6.1 (2019), pp. 1–48.

[SM05] Thorsten Schöler and Christian Müller-Schloer. "An Observer/Controller Architecture for Adaptive Reconfigurable Stacks." In: *Systems Aspects in Organic and Pervasive Computing - ARCS 2005*. Ed. by Michael Beigl and Paul Lukowicz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 139–153. ISBN: 978-3-540-31967-2.

[Son+22] Hwanjun Song et al. "Learning from noisy labels with deep neural networks: A survey." In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[Sov+20] Evgenii Sovetkin et al. "Encoder–decoder semantic segmentation models for electroluminescence images of thin-film photovoltaic modules." In: *IEEE Journal of Photovoltaics* 11.2 (2020), pp. 444–452.

[SS23] Marius Schlegel and Kai-Uwe Sattler. "Management of Machine Learning Lifecycle Artifacts: A Survey." In: *SIGMOD Rec.* 51.4 (Jan. 2023), pp. 18–35. ISSN: 0163-5808. DOI: 10.1145/3582302.3582306. URL: https://doi.org/10.1145/3582302.3582306.

[SVZ14] K Simonyan, A Vedaldi, and A Zisserman. "Deep inside convolutional networks: visualising image classification models and saliency maps." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. ICLR. 2014.

[Tab+19] Domen Tabernik et al. "Segmentation-Based Deep-Learning Approach for Surface-Defect Detection." In: *Journal of Intelligent Manufacturing* (May 2019). ISSN: 1572-8145. DOI: 10.1007/s10845-019-01476-x.

[Tab+20] Domen Tabernik et al. "Segmentation-based deep-learning approach for surface-defect detection." In: *Journal of Intelligent Manufacturing* 31.3 (2020), pp. 759–776.

[TAG19] Matteo Terreran, Morris Antonello, and Stefano Ghidoni. "Boat hunting with semantic segmentation for flexible and autonomous manufacturing." In: *2019 European Conference on Mobile Robots (ECMR)*. IEEE. 2019, pp. 1–8.

[Tea16] AzureML Team. "AzureML: Anatomy of a machine learning service." In: *Conference on Predictive APIs and Apps*. PMLR. 2016, pp. 1–13.

[Tho07] BRANTS Thorsten. "Large Language Models in Machine Translation." In: *Proc. of EMNLP-CoNLL, 2007* (2007), pp. 858–867.

[TM23] Hasan Tercan and Tobias Meisen. "Online Quality Prediction in Windshield Manufacturing Using Data-Efficient Machine Learning." In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD '23. Association for Computing Machinery, 2023, pp. 4914–4923.

[Ton+24] SM Tonmoy et al. "A comprehensive survey of hallucination mitigation techniques in large language models." In: *arXiv preprint arXiv:2401.01313* (2024).

[Tra+22] Phuong Hanh Tran et al. "Application of Machine Learning in Statistical Process Control Charts: A Survey and Perspective." In: *Control Charts and Machine Learning for Anomaly Detection in Manufacturing*. Cham: Springer International Publishing, 2022, pp. 7–42. ISBN: 978-3-030-83819-5. DOI: 10.1007/978-3-030-83819-5_2. URL: https://doi.org/10.1007/978-3-030-83819-5_2.

[UC21] Dennis Ulmer and Giovanni Cinà. "Know your limits: Uncertainty estimation with relu classifiers fails at reliable ood detection." In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 1766–1776.

[Umb05] S.E. Umbaugh. *Computer Imaging: Digital Image Analysis and Processing*. A CRC Press book. Taylor & Francis, 2005. ISBN: 9780849329197. URL: https://books.google.de/books?id=JNhRSAMFn6YC.

[Unn+19] Saritha Unnikrishnan et al. "Machine Learning for Automated Quality Evaluation in Pharmaceutical Manufacturing of Emulsions." In: *Journal of Pharmaceutical Innovation* (2019), pp. 1–12.

[VEA22] Daniel Vale, Ali El-Sharif, and Muhammed Ali. "Explainable artificial intelligence (XAI) post-hoc explainability methods: Risks and limitations in non-discrimination law." In: *AI and Ethics* 2.4 (2022), pp. 815–826.

[Ver+19] Sachin Vernekar et al. *Out-of-distribution Detection in Classifiers via Generation*. 2019. arXiv: 1910.04241 [cs.LG].

[VH20] Jesper E Van Engelen and Holger H Hoos. "A survey on semi-supervised learning." In: *Machine learning* 109.2 (2020), pp. 373–440.

[vHF22] Benjamin van Giffen, Dennis Herhausen, and Tobias Fahse. "Overcoming the pitfalls and perils of algorithms: A classification of machine learning biases and mitigation methods." In: *Journal of Business Research* 144 (2022), pp. 93–106. ISSN: 0148-2963. DOI: https://doi.org/10.1016/j.jbusres.2022.01.076. URL: https://www.sciencedirect.com/science/article/pii/S0148296322000881.

[Wan+04] Z. Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity." In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[WCL18] Shengping Wen, Zhihong Chen, and Chaoxian Li. "Vision-based surface inspection system for bearing rollers using convolutional neural networks." In: *Applied Sciences* 8.12 (2018), p. 2565.

[Wei+19] Dorina Weichert et al. "A review of machine learning for the optimization of production processes." In: *The International Journal of Advanced Manufacturing Technology* 104.5-8 (2019), pp. 1889–1902.

[WF16] Jian Wu and Peter Frazier. "The parallel knowledge gradient method for batch Bayesian optimization." In: *Advances in neural information processing systems* 29 (2016).

[WJ17] Zi Wang and Stefanie Jegelka. "Max-value entropy search for efficient Bayesian optimization." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3627–3635.

[WRJ90] James P Womack, Daniel Roos, and Daniel T Jones. *The machine that changed the world*. New York, NY: Scribner, Oct. 1990.

[WSL06] Gabriele Winter, Karlheinz Schaub, and Kurt Landau. "Stress screening procedure for the automotive industry: Development and application of screening procedures in assembly and quality control." In: *Occupational Ergonomics* 6.2 (2006), pp. 107–120.

[Wu+21] Shan Wu et al. "This is the way: Sensors auto-calibration approach based on deep learning for self-driving cars." In: *IEEE Sensors Journal* 21.24 (2021), pp. 27779–27788.

[Wue+16] Thorsten Wuest et al. "Machine learning in manufacturing: advantages, challenges, and applications." In: *Production & Manufacturing Research* 4.1 (2016), pp. 23–45.

[XM12] Huan Xu and Shie Mannor. "Robustness and generalization." In: *Machine learning* 86 (2012), pp. 391–423.

[Xu+19] Feiyu Xu et al. "Explainable AI: A brief survey on history, research areas, approaches and challenges." In: *Natural Language Processing and Chinese Computing: 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II 8*. Springer. 2019, pp. 563–574.

[Yan+18] Yaoqing Yang et al. "FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[Yan+24] Jingkang Yang et al. "Generalized out-of-distribution detection: A survey." In: *International Journal of Computer Vision* (2024), pp. 1–28.

[Yap+14] Bee Wah Yap et al. "An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets." In: *Proceedings of the first international conference on advanced data and information engineering (DaEng-2013)*. Springer. 2014, pp. 13–22.

[Yi+22] John Seon Keun Yi et al. "Pt4al: Using self-supervised pretext tasks for active learning." In: *European Conference on Computer Vision*. Springer. 2022, pp. 596–612.

[Yin19] Xue Ying. "An overview of overfitting and its solutions." In: *Journal of physics: Conference series*. Vol. 1168. IOP Publishing. 2019, p. 022022.

[YK19] Donggeun Yoo and In-So Kweon. "Learning Loss for Active Learning." In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 93–102.

[Zei12] Matthew D Zeiler. "ADADELTA: an adaptive learning rate method." In: *arXiv preprint arXiv:1212.5701* (2012).

[Zha+21a] Dingwen Zhang et al. "Weakly supervised object localization and detection: A survey." In: *IEEE transactions on pattern analysis and machine intelligence* 44.9 (2021), pp. 5866–5885.

[Zha+21b] Ganning Zhao et al. "CalibDNN: multimodal sensor calibration for perception using deep neural networks." In: *Signal Processing, Sensor/Information Fusion, and Target Recognition XXX*. Vol. 11756. SPIE. 2021, pp. 324–335.

[Zha+22] Yi-Fan Zhang et al. "Focal and efficient IOU loss for accurate bounding box regression." In: *Neurocomputing* 506 (2022), pp. 146–157.

[Zha+23] Aston Zhang et al. *Dive into Deep Learning*. `https://D2L.ai`. Cambridge University Press, 2023.

[ZP17] Chong Zhou and Randy C Paffenroth. "Anomaly detection with robust deep autoencoders." In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 665–674.

Own publications

[Mas23a] Philipp Mascha. "A Comparison of Model Confidence Metrics on Visual Manufacturing Quality Data." In: *Computer Vision and Machine Intelligence*. Ed. by Massimo Tistarelli et al. Singapore: Springer Nature Singapore, 2023, pp. 165–177. ISBN: 978-981-19-7867-8.

[Mas23b] Philipp Mascha. "VMLC: Statistical Process Control for Image Classification in Manufacturing." In: *Proceedings of Machine Learning Research Workshop and Conference Proceedings*. Vol. 222. 2023.