Reducing supervision in industrial computer vision tasks

Doctoral thesis for obtaining

the academic degree

Doctor of Engineering Sciences (Dr.-Ing.)

submitted by

Matthias Hermann

at the

Universität Konstanz

Faculty of Sciences

Department of Computer and Information Science

Konstanz, 2023

Date of the oral examination: 1 March 2024
First reviewer: Prof. Dr. Bastian Goldlücke
Second reviewer: Prof. Dr. Matthias O. Franz
Chair: Prof. Dr. Tobias Sutter

Danksagung

Als ich im Oktober 2017 am Institut für Optische Systeme die Stelle im BMBF Projekt Multiflexinspect übernahm, wusste ich wenig über Oberflächeninspektion, Informationstheorie oder darüber, wie interessant und lehrreich die folgenden Jahre werden würden. In dieser Zeit begleiteten mich Mentoren, Kollegen, Studenten sowie geliebte Menschen aus meinem persönlichen Umfeld denen ich zu großem Dank verpflichtet bin.

Zuerst möchte ich meinem Hauptbetreuer und Mentor Prof. Matthias O. Franz danken. In der Zeit der Promotion war er immer mein zentraler Orientierungspunkt, Ratgeber und Entropiequelle. Eine solch enge Zusammenarbeit verstehe ich nicht als selbstverständlich und ich bin ihm für seine Führung in dieser Zeit sehr dankbar. Seine Expertise, seine Klarheit und seine Fähigkeit komplexe Themen anschaulich zu erklären, zeichnen ihn als Professor und Mentor aus. Für seine Rücksicht und seine bestärkenden Worte in schwierigen Phasen, professionell wie privat, bin ich ihm sehr dankbar. Ohne ihn wäre diese Dissertation nicht möglich gewesen. Weiter möchte ich Prof. Georg Umlauf für seine richtungsweisende Betreuung und die verantwortungsvolle Leitung des IOS Konstanz danken. Ohne sein Rückhalt und das Institut wäre diese Promotion ebenfalls nicht möglich gewesen. Auf sein Gespür für neue Kooperationen, neuartige Ansätze und komplexe algorithmische Zusammenhänge konnte ich mich immer verlassen. An der Universität Konstanz möchte ich mich besonders bei Prof. Bastian Goldlücke bedanken. Die Gespräche mit ihm und die daraus resultierende Hilfestellungen und Leitplanken führten mich maßgeblich durch die Promotion und die Prozesse der Universität.

Ein besonderer Dank gilt meinen Kollegen am IOS Konstanz. Hier danke ich besonders Michael Grunwald für die Einführung in die Strukturen am IOS und die enge Zusammenarbeit bei zahlreichen Forschungsprojekten. Die gemeinsamen Gespräche eröffneten mir stets neue Perspektiven und ohne seine positive Energie und Bestärkung hätte ich den Schritt in die Promotion nicht gewagt. Für neue Impulse und den regen Austausch möchte ich weiter Martin Schall, Oliver Dürr, Pascal Laube, Tobias Birkle, Dennis Grießer, und Daniel Dold danken. Außerdem möchte ich mich bei Baumer Inspection GmbH für die herausragende Kooperation bedanken. Insbesondere Stefan Bickert und Dr. Christoph Schmitt gewährleisteten hier stets einen funktionierenden Rahmen und ihr Industrieblick eröffnete häufig neue Möglichkeiten. Darüber hinaus möchte ich mich bei Dominik Volk und Robin Reiter für die unendlichen Abende und Wochenenden mit unbezahlbaren Gedanken und Ideen über die Jahre bedanken. Ihre Hingabe zur Technik und ihr Streben nach Verbesserung ist immer wieder ein elementarer Motivationsfaktor.

Außerdem möchte ich mich bei meiner Partnerin Rebecca für die endlose Geduld, ihre Liebe und Unterstützung in allen Lebenslagen bedanken. Die langen abendlichen und morgendlichen Diskussionen sind elementarer Bestandteil unseres Lebens und Schaffens. Ich bewundere sie für ihre Pragmatik und ihren gelassenen Blick von außen - Danke! Für ihren unbeugsamen Rückhalt möchte ich mich bei meinen Eltern Marika und Norbert bedanken. Beharrlich standen sie, auch bei starkem Gegenwind, hinter mir. Dasselbe gilt für meinen Bruder Fabian. Vielen Dank für eure Unterstützung, eure Konsequenz und euren Zuspruch. Meinen Schwiegereltern in spe Ulrike und Kurt möchte ich für die bedingungslose Unterstützung in allen Lebenssituationen danken. Vielen Dank, dass ihr mir und uns unter die Arme greift, wann immer es nötig ist.

Weitere wichtige Stützen über den gesamten Schaffensprozess waren Theresa und Henni. Danke, dass ihr ein Teil meines Lebens seid!

Zusammenfassung

Insbesondere bei Produkten, die einer ästhetischen Bewertung unterzogen werden, muss der industrielle Fertigungsprozess überwacht und visuelle Fehler erkannt werden. Zu diesem Zweck werden immer mehr Inspektionssysteme mit integrierter Computer Vision eingesetzt. Bei der optischen Inspektion mittels Kameras oder Entfernungssensoren ist in der Regel nur eine kleine Anzahl von Beispielen bekannt, bevor neue Beispiele inspiziert werden. Folglich steht kein großer Datensatz mit fehlerfreien und fehlerhaften Beispielen zur Verfügung, der zum Trainieren eines Klassifikators verwendet werden könnte. Aus diesem Grund müssen Methoden verwendet werden, die mit begrenzter oder schwacher Überwachung zurechtkommen. Für derartige Szenarien schlage ich neue dateneffiziente maschinelle Lernmethoden vor, die auf Ein-Klassen-Lernen basieren und damit nötige Überwachung beim Einsatz von Computer Vision in der Industrie reduzieren. Das entwickelte Modell zur Erkennung von Auffälligkeiten extrahiert automatisch Merkmale aus den Eingabebildern und wird nur auf verfügbaren nicht-defekten Referenzdaten trainiert. Auf den Merkmalsextraktor wird ein Ein-Klassen-Klassifikator auf der Grundlage der jüngsten Entwicklungen im Deep Learning aufgesetzt. Ich evaluiere den Defektdetektor sowohl in einem industriellen Inspektionsszenario als auch in modernsten Benchmarks aus dem Bereich des maschinellen Lernens. Im zweiten Teil dieser Arbeit wird das Modell durch die Verwendung einer kleinen Anzahl fehlerhafter Beispiele verbessert und somit eine weitere Quelle der Überwachung einbezogen. Die angestrebte reale Inspektionseinheit basiert auf einem Kamera-Array und einer Blitzlicht-Beleuchtung, die eine Inline-Erfassung von Mehrkanalbildern mit einer hohen Rate ermöglicht. Optional ist die Integration von Entfernungsdaten, wie Laser- oder Lidar-Signalen, mittels der entwickelten Methode zur Datenfusion ohne Referenzmuster möglich.

Abstract

Particularly for manufactured products subject to aesthetic evaluation, the industrial manufacturing process must be monitored, and visual defects detected. For this purpose, more and more computer vision-integrated inspection systems are being used. In optical inspection based on cameras or range scanners, only a few examples are typically known before novel examples are inspected. Consequently, no large data set of non-defective and defective examples could be used to train a classifier, and methods that work with limited or weak supervision must be applied. For such scenarios, I propose new data-efficient machine learning approaches based on one-class learning that reduce the need for supervision in industrial computer vision tasks. The developed novelty detection model automatically extracts features from the input images and is trained only on available non-defective reference data. On top of the feature extractor, a one-class classifier based on recent developments in deep learning is placed. I evaluate the novelty detector in an industrial inspection scenario and state-of-the-art benchmarks from the machine learning community. In the second part of this work, the model gets improved by using a small number of novel defective examples, and hence, another source of supervision gets incorporated. The targeted real-world inspection unit is based on a camera array and a flashing light illumination, allowing inline capturing of multichannel images at a high rate. Optionally, the integration of range data, such as laser or Lidar signals, is possible by using the developed targetless data fusion method.

Contents

List of Abbreviations

AEP	Asymptotic Equipartition Property
AUC	Area Under The (ROC) Curve
BSS	Blind Source Separation
CNN	Convolutional Neural Network
CE	Cross Entropy
ERR	Equal Error Rate
FA	Factor Analysis
FPR	False Positive Rate
GHA	Generalized Hebbian Algorithm
ICA	Independent Component Analysis
KNN	K-Nearest Neighbors
KDE	Kernel Density Estimation
L-BFGS	Broyden–Fletcher–Goldfarb–Shanno optimization
MI	Mutual Information
MINE	Mutual Information Neural Estimation
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NSCL	Null-space class learning
NN	Neural Network
OCC	One-Class Classification
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristics
RNN	Recurrent Neural Network
RGB	Red, Green, and Blue color model
RGBD	Red, Green, and Blue color model + Depth
SFA	Slow Feature Analysis
SVM	Support Vector Machine
SVD	Singular Value Decomposition
TPR	True Positive Rate
VAE	Variational Autoencoder
ViT	Vision Transformer
VQ	Vector Quantization

CHAPTER 1

Introduction

This dissertation addresses the visual detection of defects in the manufacturing industry in order to sort them out before they leave the production facility and reach the customers. Fig. 1.1 shows typical production defects, such as scratches or misplaced particles, that appear in real-world installations.

Using computer vision to automate manufactured product inspection strives to replace complex, subjective, or repetitive manual inspection processes. Generally, three different approaches to automated inspection are distinguished in the literature [97]: (1) specifying a defect model and searching for similar patterns [16], (2) looking for differences to a given reference template [17], and (3) learning to discriminate between defects and non-defects [144] Depending on available datasets, the existing inspection approaches can be translated into two machine learning paradigms: supervised and weakly supervised classification. When labeled samples of both defects and non-defect samples can be easily obtained, supervised classification approaches are preferred [157]. In practical applications, only limited, partially labeled, or strongly unbalanced data is generally available. Tasks characterized by such reduced label information belong to the weak supervision domain. Translating this into the context of industrial defect detection means that sometimes a small amount of defective data, often a large amount of non-defective data, but mostly no label information at all is available. Such situations massively complicate a purely supervised approach. Current approaches often use unsupervised or weakly supervised learning techniques such as out-

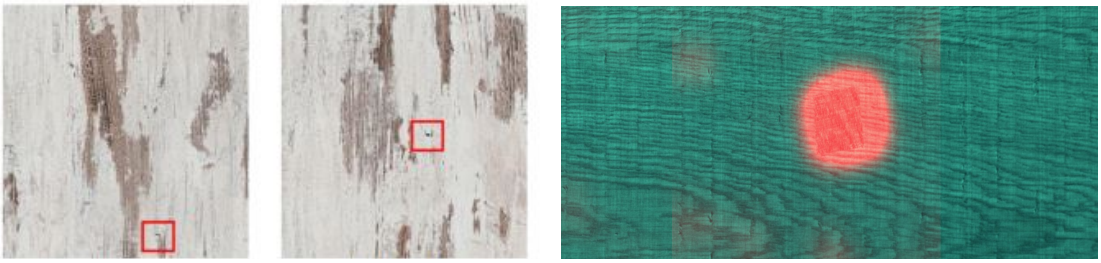


Figure 1.1: Detected surface defects, such as scratches and printing faults, on a wooden texture using the targeted industrial inspection system. Depending on the desired output, defect detections are marked by a bounding box or highlighted through a heat map.

lier and novelty detection to circumvent this issue [5, 144]. In outlier detection, the available data is polluted by an observed small fraction of unlabeled outliers, which need to be sorted out unsupervised and without further knowledge. In weakly supervised novelty detection scenarios, the training data is purely non-defective, and models seek to detect unknown unobserved defects. Such systems traditionally consist of two independent subsystems that are designed separately. The first is a domain-specific feature extraction stage, and the second is a general binary classifier applied to novel examples [136, 118, 44]. The difficulty in finding a suitable feature space for defect detection is generalizing to replicated objects with very little knowledge about possible defects.

The leitmotif of this dissertation is applying computer vision to industrial surface inspection under limited supervision and developing methods that reduce the need for labels. In the first part, this work focuses on novelty detection techniques, where the available training data is considered from a single defect-free class, and there is no knowledge about possible defects. The second part considers incremental one-class learning, where only a few examples from a previously unknown defect class are available for training. The last part covers a self-supervised data fusion approach where data from different optical sensor modalities is fused without knowing the registration parameters, and no registration target is available.

1.1 Challenges in reducing supervision

Optical inspection in industrial scenarios using computer vision consists of three tasks. After data capturing using combinations of cameras or range scanners, the task is designing a suitable feature space representation for the gathered data. The final challenge is creating a detection mechanism for successful distinction. The feature space is meant to extract information that supports the separation of non-defects from defects. Several techniques for feature design have been proposed in the inspection literature. These range from handcrafted features [125], texture analysis [190], orthogonal bases and wavelets [16, 5] to dictionaries [17], random projections [8], and pre-trained deep neural networks [183, 144]. Modern deep neural networks can extract features and learn an implicit decision boundary simultaneously when trained with huge amounts of labeled data [96]. The superiority of such deep neural networks in supervised classification tasks shows the need for deep higher-order features and labels. In analogy, current state-of-the-art methods for weakly supervised classification rely on deep autoencoders or deep generative models and use reconstruction error or log probability as a score function for separation [1]. However, such models often fail in practice and perform no better than k-nearest neighbors (KNN) or kernel density estimation (KDE) techniques when trained on the same data [124].

Convolutional neural networks (CNNs) are special neural networks inspired by biological visual receptive fields that process spatial input hierarchically [96]. In signal processing, unsupervised component analysis is a closely related approach to modeling visual receptive fields by estimating a new linear or non-linear basis for the given data without using any supervision or labels. This technique can be related to cluster analysis but generally finds a common feature basis for all examples. This line of research has two major results [75]. The first is that principal component analysis (PCA) applied to natural image patches finds a Fourier basis. The second is that edges form independent components (ICA) of natural image patches. Based on that, several approaches for designing an unsupervised deep neural network bottom-up have been proposed [155, 128, 100, 182]. The most similar concept to CNNs is the so-called tiled ICA, which consists of several tiles of independent components [126] covering patches of the input image space. Recently, this approach was revisited in explaining the superiority of modern CNN architectures [176]. Further, a very close relationship exists between PCA, ICA, and reconstruction loss in deep autoencoders. Indeed,

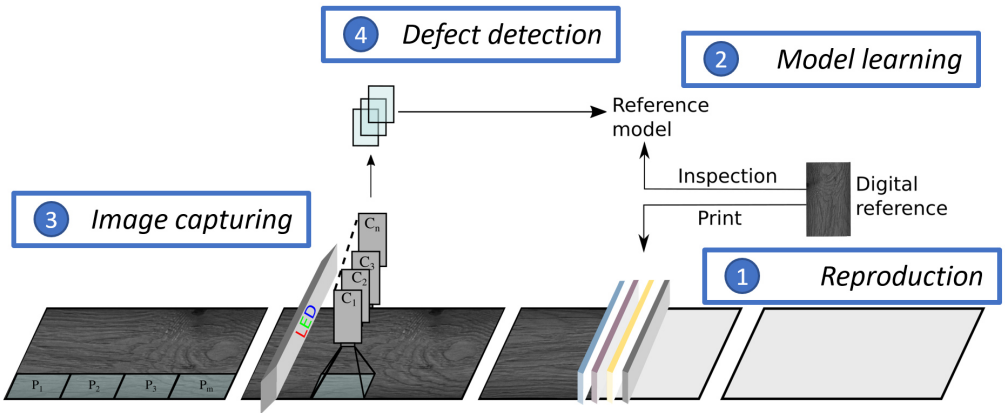


Figure 1.2: Schematic illustration of the targeted optical industrial inspection system. (1) The production system replicates digital references. (2) The task of the inspection system is to learn a model of that reference. (3) During operation, the system captures images of the replicated products. (4) Based on the previously learned model and the captured images, the system is able to detect defects.

they are equivalent in the linear case [99]. A slightly different approach for extracting higher-order features is based on slowly varying features. The so-called slow feature analysis (SFA) algorithm consists of two decoupled stages and can find linear [189] and non-linear features [83]. Although several approaches are available for modeling receptive fields, it is still unclear how to design an architecture for effective defect detection under weak supervision.

A recent direction towards weakly supervised classification is using deep hierarchical models by incorporating many non-linear layers and analyzing non-linear statistical independence [37]. Unfortunately, incorporating non-linearity makes any unsupervised model unidentifiable, making the learned features relatively meaningless [79]. However, as recently shown, this problem can be mitigated by incorporating a further source of supervision, such as using a few labeled examples or another observed auxiliary signal that can be used for self-supervised learning [77, 89]. Although there are many different aspects of learning in the weak supervision domain, there is no common understanding of how the proposed techniques can improve industrial defect detection.

1.2 Research contributions

This dissertation introduces machine learning methods for an industrial inspection system where only weakly labeled data and limited supervision are available. The captured data streams, as visualized in Fig. 1.2, arise from an optical system consisting of cameras and optional range sensors monitoring the manufacturing process. The considered data within the industrial inspection system is categorized into four partitions. Observed non-defects (*reference*), unobserved non-defects (*variants*), observed defects (*outliers*), and unobserved defects (*novelties*). For simplicity, observed data is often called training data, and all unobserved defects are considered anomalies. Label information is considered strong, whereas other available information is considered weak. Examples of such weak sources are [201]: temporal or spatial structure, implicitly given labels, i.e., only non-defective examples, few labeled defective examples, and domain knowledge about process con-

text. To solve a particular defect detection problem effectively, all available supervision must be exploited.

This work is connected to the research line of unsupervised learning [89], semi-supervised learning [200, 199], transfer learning [192], and component analysis in signal processing, which has a long history in optical inspection [97]. Based on this, the algorithmic goal is to find a data representation with suitable invariances against variants and detection capabilities towards defects. My research is analyzing learned representations with information theory in mind. In this field, the entropy of a representation distribution upper-bounds the size of its typical set [37]. Hence, high entropy increases the overall chance of intersecting with other unobserved distributions. This work is inspired by the hypothesis that the representation with the smallest typical set should be the best possible model for the measured weakly labeled data. When the processed data consists of real-world signals, independent component analysis (ICA) is known to produce such low entropy or sparse features, making it a promising starting point [75]. Following the identified research direction, this dissertation proposes novel machine learning approaches that reduce supervision in industrial computer vision tasks. The main contributions of this work are:

- A novel data-efficient novelty detection algorithm using typical set theory targeting optical inspection tasks.
- Estimating the required sparsity statistics using a novel algorithm for independent component analysis (ICA).
- Improving the detection capabilities incrementally with a small number of labeled defective examples using a novel regularization scheme for null-space learning.
- Extending the system to multi-modal sensor configurations using a novel data fusion approach based on mutual information.

1.3 Chapter overview

After this introductory Chapter 1, the organization of this thesis is based on the previously described research contributions and the corresponding publications that I published while working on industrial inspection.

Beginning with relevant fundamentals in Chapter 2, an introduction to industrial computer vision, deep learning, and relevant aspects of information theory is given. Chapter 3 introduces the developed novelty detection method μshift [67] that can efficiently detect novel defects using only a few non-defective examples. This section contains a hyperparameter selection method for the crucial optimal patch size based on the information-theoretic concepts of negentropy and typical set theory. Unfortunately, estimating negentropy is difficult for large and high-dimensional datasets that do not fit into memory. Therefore, Chapter 4 introduces Lie-Adam [65], a large-scale independent component analysis (ICA) method for estimating negentropy for large data matrices. In Chapter 5, a second one-class learning method is introduced. This algorithm targets situations where novel defect classes that were not previously known appear during the operation of an existing classifier. Such situations require updating existing models using a small number of examples. The proposed enhancement is based on null-space learning (NSCL) and introduces a novel regularization scheme that mitigates overfitting. Hence, the name of the R-NSCL [69]. As practical inspection systems often lack massive datasets, every available source of supervision simplifies the problem. Therefore, the incorporation of range data captured using laser systems or Lidar sensors is often

required. Chapter 6 covers the optional image and range data sensor fusion for the targeted inspection system. Here, the algorithm optimizes the extrinsic calibration matrix, which encodes the relative position of the camera with respect to the range sensor. The developed targetless fusion algorithm LMI [64] registers the range sensor by sampling patches from the common image plane and optimizing local mutual information between patches of both sensor modalities. The output of the algorithm is a fused RGB image with an additional depth channel (RGBD).

Because of the different relevant research areas, Chapters 3 to 6 include a separate section listing related works and a separate conclusion. An overall summary of this work is given in Chapter 7. With the exception of Chapters 1 and 7, the first-person plural personal pronoun is applied since the presented contributions were made in collaboration. The singular form is only applied when comparing the contributions of different authors or if the context reflects my own opinion.

1.4 Publications

During my time working on the topic of industrial inspection, I published various papers that reflect the results obtained throughout this period. This section lists all publications on which this thesis is based. I will outline the contributions made by each author. The publication list is ordered by supervision regime and publication date.

Unsupervised learning

- [65] **M. Hermann, G. Umlauf, and M. Franz. Large-scale independent component analysis by speeding up Lie group techniques. In *47th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022**.

  This work presents Lie-Adam, a novel independent component analysis (ICA) algorithm for large datasets that do not fit into memory. The inspiration for this algorithm came from me, as I noticed during negentropy experiments that Adam [91] outperformed standard L-BFGS optimization [109]. I analyzed the orthogonal projection problem of ICA and improved the geodesic flow [138] optimization algorithm. I implemented the algorithm and conducted all experiments. I wrote the paper in close cooperation with G. Umlauf. M. O. Franz supervised the work.

- [68] **M. Hermann, Georg Umlauf, and M. O. Franz. Fast and memory-efficient independent component analysis using Lie group techniques. In *10th International Conference on Curves and Surfaces*, 2022**.

  This work presents an extended version of Lie-Adam. As an extension, I compared different whitening strategies using the rotation freedom in the whitening space. I implemented the methods and conducted the experiments. I created the poster closely together with M. O. Franz. G. Umlauf supervised the work.

Weakly supervised learning

- [59] **Frederic Hake, Matthias Hermann, Hamza Alkhatib, Christian Hesse, Karsten Holste, Georg Umlauf, Gaël Kermarrec, and Ingo Neumann. Damage detection for port infrastructure by means of machine-learning-algorithms. In *Proceedings of FIG Working Week*, 2020**.

  This work applies novelty detection to the problem of detecting unknown damage in port infrastructure. The motivation came from Gaël Kermarrec, part of the Geodetic Institute of Leibnitz University, Hannover. F. Hake, H. Alkhatib, C. Hesse, and K. Holste developed

the data preprocessing pipeline and were responsible for data acquisition using a submarine with a pulse sensor. I implemented the novelty detection pipeline on top of the generated height fields. I wrote half of the paper in close cooperation with F. Hake and G. Kermarrec. G. Umlauf and Ingo Neumann supervised the work.

- [57] **Michael Grunwald, Matthias Hermann, Fabian Freiberg, and Matthias O Franz. Biologically-vision-inspired vs. CNN texture representations in novelty detection. In *SPIE Applications of Machine Learning*, 2021**.

  This work compares different texture representations from psychophysics and machine learning concerning the human perceptibility of defects. Additionally, it extends our Min-Max novelty detection algorithm [56] that projects normal data onto a minimal hypersphere using a neural network. The motivation for this work came from Michael Grunwald and Felix Wichmann from the University of Tübingen. I proposed an additional regularization term in the novelty detection model, which improved stability. I implemented the novelty detection pipeline using neural network features and conducted half of the experiments. I wrote half of the paper in close cooperation with M. Grunwald and F. Freiberg. M. O. Franz supervised the work.

- [66] **M. Hermann, G. Umlauf, B. Goldlücke, and M.O. Franz. Fast and efficient image novelty detection based on mean-shifts. *21th International Conference on Image Analysis and Processing (ICIAP)*, 2022**.

  This work presents μshift, a novel novelty detection algorithm based on mean shifts of extracted patch ensembles. Here, I identified a particular hyperparameter selection method that is based on the typical set size of the encoded patches. The inspiration for the method came from me based on the observation that features averaging improved novelty detection performance. I developed the method, conducted the experiments, and implemented the baseline methods. I wrote the paper in close cooperation with M. O. Franz. B. Goldlücke and M. O. Franz supervised the work.

- [67] **M. Hermann, G. Umlauf, B. Goldlücke, and M.O. Franz. Image novelty detection based on mean-shift and typical set size. *Sensors — Unusual Behavior Detection Based on Machine Learning*, 2022**.

  This work presents an extended version of the μshift [66] algorithm for novelty detection using deep neural network features. The main contribution is the new estimation technique for the necessary, sufficient statistics, such as averages and covariance, which came from my inspiration. Additionally, the work introduces novel patch-based feature representations based on deep convolutional neural networks. I implemented the method and conducted all of the experiments. I wrote the paper in close cooperation with M. O. Franz. B. Goldlücke supervised the work.

- [69] **M. Hermann, G. Umlauf, B. Goldlücke, and M. O. Franz. Incremental one-class learning using regularized null-space training for industrial defect detection. *Submitted to ICMV*, 2023**.

  This work presents R-NSCL, a novel regularization scheme for incrementally learning a single novel defect class in a few-shot scenario. The orthogonal-gradient update idea came from M. O. Franz, which I related to the null space neural network training framework (NSCL) [184]. I identified the overfitting issue of standard NSCL, developed the novel regularization scheme, implemented the baseline methods, and conducted all experiments. I wrote the

paper in close cooperation with M. O. Franz. B. Goldlücke and G. Umlauf supervised the work.

Self-supervised learning

- [64] **M. Hermann, Dennis Grießer, Bernhard Gundel, Daniel Dold, Georg Umlauf, and M. O. Franz. Targetless Lidar-camera registration using patch-wise mutual information. In *25th International Conference Information Fusion (FUSION)*, 2022**.

  This work presents LMI (Local Mutual Information), a novel algorithm for targetless self-supervised camera-Lidar calibration based on mutual information. I inspired that work based on a practical project requirement that I related to self-supervised learning. The method is based on mutual information neural estimation (MINE) [11]. I developed the algorithmic pipeline, implemented the baseline methods, and conducted all experiments. I wrote the paper in close cooperation with G. Umlauf. D. Grießer, B. Gundel, and D. Dold supported the data acquisition using the institute's sensor system. M. O. Franz supervised the work.

CHAPTER 2

Fundamentals

The following chapter introduces relevant preliminaries for the dissertation. The first part introduces computer vision and its practical data and supervision requirements. Afterward, an overview of deep learning is given. The second part introduces one-class classification and describes the modules needed for such a model. The last part briefly introduces information theory and relevant aspects of it for the following chapters.

2.1 Data and supervision requirements in computer vision

Every computer vision system is characterized by a task, the input data, the desired output data, and the algorithmic design. A particular design choice for relating the input data to the output is called a *model*. For modeling and solving computer vision problems, requirements for the particular task and preferably a diverse set of examples $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})\} \subseteq \mathcal{X} \times \mathcal{Y}$ of the input space $\mathcal{X}$ and output space $\mathcal{Y}$ are needed [172]. In the simplest form, there is only a description of the task and an explanation of the evaluation or testing scenario. Sometimes, a handful of examples illustrating the target scenario is given. Hence, development data must be entirely hallucinated, synthesized, or gathered during development. This traditional approach often leads to underfitting or overfitting. Both problems lead to imperfect solutions to the task. Typical reasons are underestimated noise, underestimated variation in the data, unknown external effects, or undefined but desired generalization properties [15]. Note that this issue is not solely a machine learning problem but also affects human developers who take handy shortcuts or design overcomplicated methods while learning about the given problem. To streamline the development of computer vision algorithms and improve evaluation, the community started to use datasets that cover particular aspects of the problem. These standardized datasets are split into a training set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test} = \{(\mathbf{x}, \mathbf{y})_1, ..., (\mathbf{x}, \mathbf{y})_M\}$ of M examples describing the expected input and desired output. The test data is usually unavailable during model development or training and simulates the real-world testing scenario. In computer vision, supervised and unsupervised learning are two dominating algorithm design concepts, which we will introduce shortly.

Supervised learning

In supervised learning, the available training data $\mathcal{D}_{train} = \{(\mathbf{x}, \mathbf{y})_1, ..., (\mathbf{x}, \mathbf{y})_N\}$ contains additional N examples from the task. This means supervision of how the test scenario will look is available during development. Supervision mostly comes in the form of labels $\mathbf{y}$ and can take

(a) Pixel-wise annotations (COCO [108]) (b) Categories (CIFAR-10 [95])

Figure 2.1: Labels in computer vision can take various forms, such as pixel-wise annotations or categories.

different forms. Fig. 2.1 shows two common types of labels: pixel-wise annotations and categories. This means that data scientists have marked each data point in the training set with the correct label (e.g., cat or dog) so that the algorithm can learn how to predict outcomes for unseen data and accurately identify objects in new image data.

Typical computer vision tasks of supervised learning algorithms include object detection [108], visual recognition [50], and classification [95]. In object detection, supervised learning algorithms are used to learn how to identify and localize objects in images. In image recognition, these algorithms are used to learn how to identify a particular object category (e.g., persons) from a set of images. Lastly, in image classification, the task is to learn how to assign a class label (e.g., cat or dog) to an image. The process of supervised learning can be outlined as follows:

1. **Dataset preparation**: Acquiring appropriately labeled data, indicating the correct answer, i.e., ground truth, for each data point. For instance, when developing a supervised machine learning model to identify specific machine parts, you must collect an image dataset with accurately labeled images containing those parts.

2. **Model training**: The model or developer learns to establish connections between specific features and corresponding labels and encodes the relationship into parameters, e.g., in deep learning (cf. Sec. 2.2), or any other type of algorithm. For example, in a supervised learning setup for image classification, the model might learn that images with many green pixels are likely representations of trees and hence labeled as *tree*.

3. **Model evaluation**: Following the training phase, assessing the model's performance on new, unseen test data $\mathcal{D}_{test}$ is essential to evaluate its effectiveness.

Unsupervised learning

In supervised learning, the available training data $\mathcal{D}_{train} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ contains additional N examples from the input, but misses labels. Therefore, the possible tasks are different, and the challenge is to find patterns in the data, such as groups, clusters, or anomalies. Therefore, typical unsupervised learning algorithms can be used for tasks such as image segmentation [196], dimensionality reduction [80], density estimation [43], and clustering [172]. In image segmentation, unsupervised learning algorithms cluster pixels and autonomously form meaningful objects

within an image. Dimensionality reduction plays a role in lossy data compression by retaining a significant portion of the vital information. Density estimation builds a statistical model of the data that can be used for detecting outliers or entropy coding. And image clustering is about finding groups in the data by coherently organizing the data.

The process of unsupervised learning can be outlined as follows:

1. **Dataset preparation**: Collecting data without labeling. For instance, to create an unsupervised model for categorizing animal images, a dataset containing images must not be organized into folders with dogs, cats, birds, etc.

2. **Model training**: The model learns to group elements or identify patterns within the data. Available domain knowledge can be integrated into the model manually but is not encoded into labels.

3. **Model evaluation**: Following the training phase, the performance assessment is identical to supervised learning. Thus, evaluating the model's performance on new, unseen test data $\mathcal{D}_{test}$.

It's important to note that the results produced by unsupervised learning algorithms can be significantly improved if the data is appropriately labeled. For practitioners, it is vital to incorporate every available source of supervision and only rely on unsupervised methods when necessary.

Data limitations and model generalization

The most notable difference between supervised and unsupervised learning algorithms for computer vision is how they handle data [53]. Supervised learning requires a labeled training dataset, whereas unsupervised learning works without labeled data. Unsupervised learning, while having a clear objective, lacks a predefined output it aims for. Its focus is on understanding the underlying structure of the data. In contrast, implementing supervised learning is more straightforward because the model directly learns how to map input data to desired outputs. However, this comes with the drawback that supervised learning demands a significant amount of initial human effort to label the data correctly, which is expensive in practice and often impossible in industrial contexts. For modern computer vision models based on deep learning (cf. 2.2), domain experts must label thousands to millions of data points. In practice, semi-supervised techniques are often applied where one reutilizes existing labels and hence mixes unsupervised and supervised learning [199]. However, the accuracy of such labeling dramatically impacts the performance of machine learning models. While this can lead to highly effective models, it comes with biases and algorithms that work best under specific conditions with data similar to the existing labeled training data.

In industry, two significant problems limit the deployment of computer vision: First, there is a significant need for domain-specific data that is not widely used in the research field. Second, there is a lack of labels because of annotation costs, missing domain knowledge, or regulatory and system restrictions. Both problems heavily impact learning and limit the achievable generalization capabilities of the models in the real world and, consequently, their practical applicability. In computer vision, there are three dominating approaches to circumvent the problem of limited data:

1. **Synthesizing training data**: The automatic or manual generation of training data has a long history, e.g., in handwriting recognition [180]. The main task is to define a generative model $p(\cdot)$ for either input data $p(\mathbf{x})$, output data $p(\mathbf{y})$ or both $p(\mathbf{x}, \mathbf{y})$. Afterward, new

data is sampled from the model, e.g., $\mathbf{x}^* \sim p(\mathbf{x})$. Despite the recent success of such models, training models on domain-specific data is still hard as massive data is needed [38].

2. **Transfer learning**: Another approach is learning on another source domain where sufficient data is available. The learned model is then transferred to the target domain [192] and potentially finetuned with labeled examples that are available [73].

3. **Semi-supervised learning**: A related concept to transfer learning is semi-supervised learning [199]. Here, labeled and unlabeled data sets are available. Instead of transferring and finetuning subsequently, the model is trained using examples from both sets simultaneously.

4. **Self-supervised learning**: When there is no source domain to exploit, the third approach is deriving surrogate labels $\mathbf{u}$ from the data itself and using these instead of actual labels $\mathbf{y}$. A typical approach is augmenting the data and trying to predict the type of augmentation or the original input [51]. There are significant advances in that direction, but recent models still require massive amounts of input data to learn suitable models for downstream tasks [129].

With the advent of large foundation models for computer vision, such as VGG-19 [168], or ViT [46], that are trained on millions of labeled data examples, transfer learning has become the de facto standard throughout computer vision [73, 183, 46, 61, 58]. This was enabled by the public availability of large labeled datasets, such as ImageNet [42, 154], or COCO [108].

2.2 Deep learning

Deep learning is a subset of machine learning that focuses on training and utilizing artificial neural networks, known as deep neural networks, to solve complex problems. It is a rapidly evolving field that has garnered significant attention and revolutionized various domains, including computer vision [46], natural language processing [181], and signal generation [178]. Deep learning has achieved remarkable breakthroughs and state-of-the-art results in numerous domains. Applications range from image and object recognition [96], natural language understanding [181], machine translation [55], autonomous robots [22], drug discovery [85], and even playing complex games like chess or Go [166]. The power of deep learning lies in its ability to automatically learn hierarchical representations, eliminating the need for manually crafted features and enabling end-to-end learning from raw data.

Deep learning mimics the structure and functioning of the human brain's neural networks [53]. These networks comprise multiple layers of interconnected artificial neurons that process and transform input data, progressively extracting higher-level features and representations [165]. The term *deep* refers to the depth of these neural networks, typically consisting of numerous hidden layers between the input and output layer [53]. Fig. 2.2 shows a typical neural network with two hidden layers. Mathematically a neural network maps input data to outputs

$$\hat{\mathbf{y}} = \mathbf{W}\phi_{\tilde{\theta}}(\mathbf{x}), \tag{2.1}$$

where the predictions $\hat{\mathbf{y}}$ are computed by linearly mapping the F-dimensional neural network feature vector $\phi(\mathbf{x})$ to the D-dimensional output space using the matrix $\mathbf{W} \in \mathbb{R}^{D \times F}$. The network's trainable weights, i.e., parameters, are denoted by the vector $\tilde{\theta}$. Together with the trainable output mapping $\mathbf{W}$, they form the parameter space $\theta = \{\tilde{\theta}, \mathbf{W}\}$ for optimization. One of the critical advantages of deep learning is its ability to learn patterns and relationships from vast amounts of

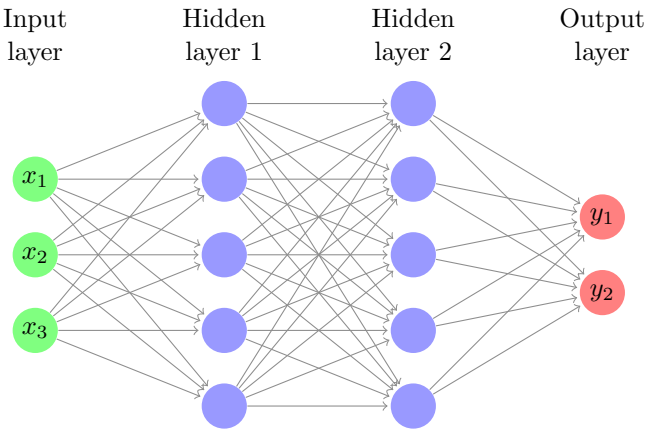


Figure 2.2: A simple neural network with three input nodes, two hidden layers, and two output nodes.

data automatically. This characteristic makes it particularly effective in handling unstructured and high-dimensional data, such as images, text, and audio. Deep learning models excel at capturing complex dependencies, enabling them to make accurate predictions and classifications in various tasks [46, 178, 96]. To train deep neural networks, a large labeled dataset is required. The training process involves iteratively adjusting the network's weights and biases using optimization algorithms to minimize the difference between predicted outputs $\hat{\mathbf{y}}$ and ground truth labels $\mathbf{y}$ [153]. The difference is expressed by a loss function $\mathcal{L}$, such as mean-squared-error (MSE)

$$\mathcal{L}_{MSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{D}\sum_{k=1}^{D}(\hat{\mathbf{y}}_k - \mathbf{y}_k)^2 \tag{2.2}$$

or cross-entropy (CE) loss

$$\mathcal{L}_{CE}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=k}^{D}\mathbf{y}_k \log \hat{\mathbf{y}}_\mathbf{k}, \tag{2.3}$$

evaluated on the training examples [53]. Additionally, deep learning algorithms often incorporate techniques like regularization, dropout [170], and batch normalization [156] to enhance generalization and prevent overfitting.

Neural network training

Backpropagation [153] is a fundamental technique that enables deep neural networks to learn complex patterns and representations from large amounts of data. It is a method for efficiently computing the gradients of the model's weights with respect to the loss function $\nabla_\theta \mathcal{L}$, which allows for the optimization of the model parameters through gradient descent [18]. The backpropagation algorithm operates by propagating the errors or gradients backward through the network, starting from the output layer and moving toward the input layer by applying the chain rule of calculus. Therefore, it calculates the contribution of each weight $\theta^{(j)}$ in the network to the overall error. By iteratively adjusting the weights in the direction opposite to the gradient with respect to the loss, i.e.,

$$\theta_{i+1} = \theta_i - \eta \nabla_{\theta_i} \mathcal{L}, \tag{2.4}$$

the network can gradually minimize the loss function and improve its performance. Here η is the chosen step size for the descent. The backpropagation algorithm consists of two main phases: the forward pass and the backward pass.

- **Forward pass**: During the forward pass, the input data is fed into the neural network, and the activations of each neuron are computed layer by layer, starting from the input layer and progressing through the hidden layers until reaching the output layer. The activations are computed using the current weights of the network.

- **Backward pass**: In the backward pass, the gradients of the loss function with respect to the outputs of the neurons in the output layer are calculated. These gradients represent the contribution of each output neuron to the overall error. Then, the gradients are propagated backward through the network, layer by layer, using the chain rule. At each layer, the gradients are multiplied by the local gradients of the activation function and are passed on to the previous layer. This process continues until the gradients reach the input layer.

Once the gradients have been calculated for each weight in the network, they can be used to update the weights using any gradient-based optimization algorithm, such as stochastic gradient descent [18], ADAM [91], or L-BFGS (Broyden–Fletcher–Goldfarb–Shanno) optimization [109].

As research in deep learning continues to advance, novel architectures, i.e., definitions of $\phi(\mathbf{x})$, such as convolutional neural networks (CNNs) [101], recurrent neural networks (RNNs) [116], and transformers [181], have emerged to tackle specific problem domains and improve performance. These advancements, coupled with the availability of massive computational resources, have enabled deep learning to drive innovations in various industries.

2.3 One-class learning

In many applications, determining whether an observation belongs to the same category, i.e., an inlier, or should be considered different, i.e., an outlier, is crucial. This capability is often used to clean real datasets, detect novelties, or learn a novel concept incrementally. The term one-class refers to the fact that the available training data belongs to a single category, and no fine-grained labels categorizing each observed example exist. Three main problems appear in one-class learning [135]:

- **Outlier detection**: In this scenario, the considered data contains outliers, which are observations that significantly deviate from the majority. Estimators for outlier detection aim to model the regions where the data is most densely concentrated and classify distant observed examples as outliers.

- **Novelty detection**: Here, training data that is free from outliers is available, and the task is to derive a decision boundary for identifying whether a new observation is an outlier or an inlier. In this context, an outlier is also considered a novelty as it is unobserved during training.

- **Incremental learning**: This scenario describes the extension of an existing classifier with a single novel class. Here, the training data consists of novel observed examples from a previously unknown class, and a decision boundary for them needs to be derived. Compared to novelty detection, the decision boundary must be compatible with the existing decision regions and should not interfere with previously learned concepts.

Both outlier detection and novelty detection are utilized for anomaly detection, where the focus is on identifying abnormal or unusual observations. Outlier detection is known as unsupervised anomaly detection, while novelty detection is called weakly supervised anomaly detection. In the

case of outlier detection, the outliers or anomalies are assumed to be located in low-density regions, and they typically do not form dense clusters. In novelty detection, on the other side, novelties or anomalies can form dense clusters as long as they are situated in low-density regions relative to the training data, which is considered normal in this context [133]. The task differs in incremental learning as novel examples are observed during training. Hence, existing learned decision regions must be adjusted with respect to the novel concept without changing the previous behavior.

A one-class learning system typically comprises three components that are designed separately [135].

- **Domain knowledge**: Learning with positive data, positive and unlabeled data, or learning with positive data in the presence of little knowledge about negative data.
- **Feature extraction**: Using handcrafted features, statistical data-driven features, or deep-learning-based features.
- **Classification algorithm**: Leveraging statistical classification methods, representation-based classification methods, general hyperplanes, or deep-learning-based methods to define decision boundaries.

2.3.1 Feature extraction

For every model, the collected data examples need to be represented in some set that consists of individual observations $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$. The problem of data representation and deriving features that contribute to classification has been extensively studied in the context of multi-class classification [53, 101, 165]. Generally, to classify objects accurately, it is essential to select features from the observations that enable a classifier to distinguish regions where examples from each class consistently appear. Similarly, in one-class classification (OCC), selecting features that separate positive class data from the rest of the objects is crucial. However, learning or selecting such features becomes more challenging in OCC, as there is no access to non-positive data during training. Nevertheless, two key properties are desirable for an effective feature in one-class classification [134]:

- **Compactness**: A desirable feature should have a similar representation for different images of the same class. Thus, features extracted from a set of images from a given class should be compactly clustered in the feature space and have low entropy.
- **Descriptiveness**: The feature should produce distinct representations for images of different classes and, therefore, carry as much information as possible about the input. Ideally, each class should have a feature representation that is distinct from other classes.

In the early stages of computer vision, features were manually engineered to provide descriptive representations of images [39]. These features are commonly referred to as handcrafted features. Subsequently, researchers turned to data-driven approaches that optimize feature representations [57]. The usage of these data-driven methods for feature extraction does not differ significantly from the usage of handcrafted features and is generally denoted by an abstract feature extraction function $\phi(\cdot)$, transforming input data examples $\mathbf{x}$ into their feature representation $\phi(\mathbf{x})$. The feature space Φ consists of all such mappings from the input space $\mathcal{X}$,

$$\Phi = \{\phi(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}. \tag{2.5}$$

Data-driven approaches to optimizing feature representations have become a separate research field called representation learning [53]. Recent works on representation learning utilize deep learning techniques with sophisticated loss functions for optimization. Generally, these techniques can be divided into four groups characterized by the availability of label information.

Semi-supervised

The semi-supervised regime is characterized by a given training set $\mathcal{D}_{train} = \{(\mathbf{x}, \mathbf{y})_1, ..., (\mathbf{x}, \mathbf{y})_N\}$ that consists of tuples of N data examples $\mathbf{x}$ with corresponding labels $\mathbf{y}$ from a similar related domain. Based on that, a standard deep neural network is trained by standard backpropagation. During training, the first layers learn a compressed descriptive data representation, while the last layers learn a classifier [165]. After training, the intermediate activations of the first layers can be used as data representations for one-class classification. Transferring features to another task is also called transfer learning [183] and is widely used in deep learning [73].

Weakly supervised

Weakly-supervised learning is characterized by a given training set $\mathcal{D}_{train} = \{(\mathbf{x}, \hat{\mathbf{y}})_1, ..., (\mathbf{x}, \hat{\mathbf{y}})_N\}$ that consists of tuples of data examples $\mathbf{x}$ and labels $\hat{\mathbf{y}}$. In contrast to the supervised regime, $\hat{\mathbf{y}}$ is not fully observed, not aligned, or loosely derived from context. Examples of weak labeling are the knowledge that all data examples arise from a known distribution [201] or observed related auxiliary signals, such as audio and video. Weakly supervised algorithms learn a feature representation by optimizing a proxy measure such as joint log probability [89] or multi-modal contrast [58]. Here, the popular CLIP [58] model produces surprisingly universal features that can be used for many downstream tasks.

Self-supervised

The self-supervised regime is characterized by a given training set $\mathcal{D}_{train} = \{(\mathbf{x}, \mathbf{u})_1, ..., (\mathbf{x}, \mathbf{u})_N\}$ that consists of tuples of data examples $\mathbf{x}$ and synthetic labels $\mathbf{u}$. These synthetic labels are derived from data directly and can be distorted or augmented data examples [106, 51]. Deriving synthetic labels allows for incorporating domain knowledge such as predefined orientation or available teachers. Recent algorithms in that area, such as DINO [129] or masked autoencoders (MAE) [61], produce features comparable to features from supervised training.

Unsupervised

There is no label information in the unsupervised regime, and the given training set $\mathcal{D}_{train} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ consists only of data examples. Traditionally, this is the regime of autoencoder (AE) techniques, such as principal component analysis (PCA) and deep autoencoders (DAEs) [88]. Recent developments showed that unsupervised feature learning results in arbitrary feature representations and is mostly unsuitable for learning high-level abstractions needed for downstream tasks, such as classification or object detection [81].

2.3.2 Decision boundaries

The final stage of any one-class classification algorithm is formulating the classification boundary. Several approaches for modeling such a boundary have been proposed over the years [135]. In the following, three popular choices are introduced shortly. The resulting hyperplanes are visualized in Fig. 2.3.

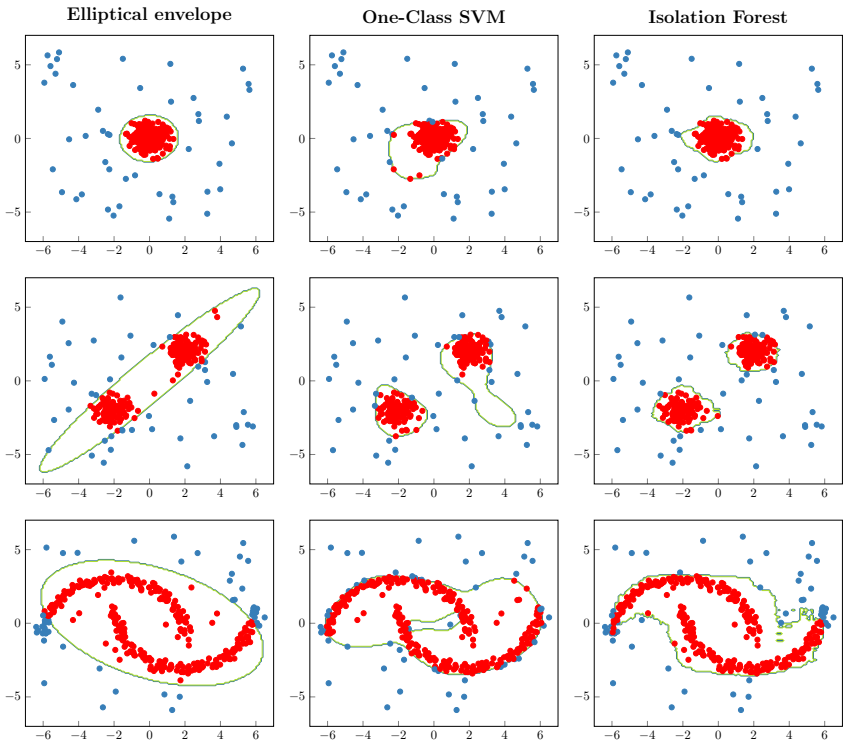


Figure 2.3: Visualizing the shapes of the decision boundary for various one-class classification algorithms.

Elliptical envelope

The elliptical envelope algorithm [149] is a statistical algorithm used for outlier detection. It assumes that the majority of the data points are generated from a multivariate Gaussian distribution. The algorithm estimates the parameters of this Gaussian distribution to create an ellipse that encompasses the normal data points. The elliptical envelope algorithm identifies outliers as data points that lie outside the boundaries of the estimated elliptical region. To build the model, the algorithm estimates the mean vector and covariance matrix of the data points.

One-class SVM

The one-class support vector machine (OC-SVM) [160] algorithm aims to find a hyperplane that separates the majority of data points from the origin while including as few points as possible on the other side of the hyperplane. This hyperplane acts as a decision boundary to distinguish between normal and abnormal instances. OC-SVM achieves this by maximizing the margin around the origin while allowing a predefined fraction of training instances to be considered outliers.

Isolation forrest

The isolation forest algorithm [110] is an ensemble-based outlier detection algorithm that uses isolation trees to identify anomalies in a dataset. It is particularly effective in detecting outliers in high-dimensional data. The algorithm works by randomly selecting a feature and a split value within the range of that feature for each isolation tree. It recursively partitions the data based on these splits until each data point is isolated in its leaf node. Since anomalies are expected to be less frequent and more easily isolated, they typically require fewer splits than normal instances. The

algorithm forms an isolation forest ensemble by constructing multiple isolation trees in parallel. The overall anomaly score for each data point is then calculated based on the average path length required to isolate that point across all the trees.

Simple and effective pipelines

Recent developments often combine learned feature representations with a classical approach [59]. The standard pipeline for such a transfer learning approach is pre-training a neural network on a large dataset, such as ImageNet [42]. After training, this fixed feature representation is used, and a standard one-class classifier, such as OC-SVM, is used as the classification algorithm. Across domains, it has been shown that changing the feature space Φ or pre-training protocol has a much greater impact on the resulting classification model than tuning the classifier itself [73].

2.3.3 Evaluation metrics

One-class classifiers are assessed using a test dataset containing both positive and negative data, making their testing procedure similar to that of binary classifiers or detectors. Previous studies in the field commonly utilize the Receiver Operating Characteristics (ROC) curve to measure the performance of one-class classification [52, 135, 1]. The ROC curve depicts the relationship between the false positive rate (FPR) and the true positive rate (TPR) a classifier achieves at every threshold. The TPR is defined as the ratio of correctly classified positive examples to the total number of positive examples, while the FPR is the ratio of misclassified negative examples to the total number of negative examples:

$$TPR = \frac{\#\,\text{correct positives}}{\#\,\text{positives}}, \tag{2.6}$$

$$FPR = \frac{\#\,\text{misclassified negatives}}{\#\,\text{negatives}}. \tag{2.7}$$

Several metrics can be derived for evaluation based on the ROC curve. One commonly used metric is the AUC-ROC curve, which represents the area under the ROC curve [161]. An ideal classifier would have an AUC-ROC value of 1.0, while random guessing would result in an AUC-ROC of 0.5. The AUC-ROC does not rely on specific operating points for evaluation, thus providing a measure of effectiveness independent of a particular threshold. Another metric derived from the ROC curve is the FPR at a selected TPR value (FPR@TPR), also known as the equal error rate (EER) [161]. The EER is the FPR value corresponding to the point on the ROC curve where both the FPR and TPR are equal. This metric provides a specific operating point for evaluating the classifier's performance.

While these metrics provide insights into the classifier's effectiveness, independent of operating points and considering both positive and negative examples, accuracy is often used in practice:

$$ACC = \frac{\#\,\text{correct classified}}{\#\,\text{positives} + \#\,\text{negatives}}. \tag{2.8}$$

The widespread adoption of accuracy as a performance metric can be attributed to its simplicity in the calculation, ease of interpretation, and ability to provide a single value summarizing the

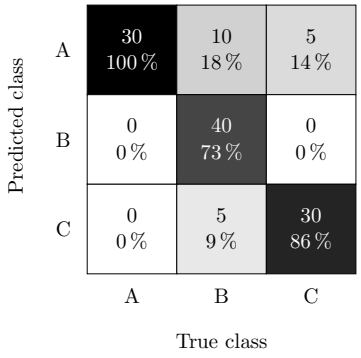


Figure 2.4: Confusion matrix with three classes and 120 data examples. The columns of the matrix represent the instances of the actual true class while each row represents the predicted class.

model's capability. As a result, it is common to see accuracy also used in imbalanced classification problems, where the distribution of examples across different classes in the training dataset is unequal [19]. This is a common pitfall: When the class distribution is slightly skewed, accuracy can still be a helpful metric. However, when the skew becomes more pronounced, accuracy loses its reliability as a measure of model performance. This lack of reliability stems from the standard practices and intuitions of machine learning practitioners regarding classification accuracy [47]. Typically, classification is carried out on small datasets with equal or nearly equal class distributions. Consequently, many believe that a high accuracy score indicates good performance, and values above 90 % are considered excellent. However, achieving 90 % or even 99 % accuracy in an imbalanced classification problem can be deceptively easy. As a result, relying on intuitions developed from balanced class distributions leads practitioners astray, causing them to mistakenly believe that a model performs well or even exceptionally when in reality, it does not [47]. Therefore, it is essential to check the used evaluation metrics and always reinsure, for example, by looking at the confusion matrix [140]. Fig. 2.4 shows an example of such a confusion matrix for three classes and 120 data points.

While the ROC Curve and AUC-ROC are commonly used and effective evaluation methods, they can yield overly optimistic results when dealing with severe class imbalances, particularly when the minority class has a small number of examples. As an alternative, the precision-recall curve can be employed similarly when explicitly emphasizing the classifier's performance regarding the minority class is desired [36].

2.4 A bit of information theory

Information theory is a branch of mathematics and computer science that deals with the quantification, transmission, and processing of information [37]. It provides fundamental tools and concepts for understanding and analyzing various aspects of information, including its measurement, encoding, transmission, and processing. The fundamental properties of information and efficient communication systems, data compression techniques, and statistical models can be characterized by concepts like entropy, the typical set, and mutual information [37].

2.4.1 Entropy

Entropy is a key concept in information theory [37]. It measures the average amount of uncertainty or randomness in a data set. In other words, it quantifies the amount of information contained

in a random variable. Entropy is typically denoted by the symbol $\mathcal{H}$ and is calculated using the probabilities associated with the different possible outcomes of the variable. Higher entropy indicates greater uncertainty and lower predictability, while lower entropy corresponds to more predictable data. The general formula is given by

$$\mathcal{H}(X) = -\sum_{i=1}^{n} P(X_i) \log P(X_i), \tag{2.9}$$

where $P(X_i)$ is the probability of the i-th outcome of X and n is the total number of possible outcomes of X. There is a useful connection between entropy and the expected negative log probability that is given by

$$\mathcal{H}(X) = \mathbb{E}[-\log P(X)]. \tag{2.10}$$

This formulation is beneficial in maximum-likelihood optimization, where the probability distribution P_θ is parametrized by parameters θ. In this framework, maximizing the expected log probability with respect to a dataset and the parameters optimizes an upper bound to the entropy of the dataset. This is heavily used in generative modeling where the task is to learn the distribution of data [43].

A related concept is the term relative entropy which is also called Kullback–Leibler divergence [37]. Relative entropy measures the difference between two probability distributions. Generally, the formula is given by

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x} P(x) \log \left(\frac{P(x)}{Q(x)} \right), \tag{2.11}$$

where x are possible outcomes of X. However, a more entropy-related formulation is given in terms of the difference between cross-entropy and entropy

$$D_{\mathrm{KL}}(P\|Q) = \mathcal{H}(P, Q) - \mathcal{H}(P). \tag{2.12}$$

Here, the term cross-entropy, given by

$$\mathcal{H}(P, Q) = -\sum_{i=1}^{n} P(X_i) \log Q(X_i), \tag{2.13}$$

specifies the average number of bits needed to identify an event drawn from the set if the used coding scheme is optimized for another probability distribution Q instead of the correct distribution P. In deep learning, cross-entropy is the standard loss for classification problems [95].

2.4.2 Typical set

The typical set is a fundamental concept in information theory that allows us to identify the most probable outcomes of a random variable [37]. It represents a subset of the possible outcomes that have a high probability, capturing the typical behavior of the variable. For example, consider a binary Bernoulli variable with $p(0) = 0.1$ and $p(1) = 0.9$ and two possible observed sequences: (1) $(1, 1, 1, 1, 1)$ and (2) $(1, 1, 1, 1, 0)$. Because $p(0) < p(1)$, the first sequence is most likely, but only the second is typical and, therefore, part of the typical set. Considering the typical set, we can efficiently encode and transmit data by focusing on the most probable outcomes. The typical set is closely related to the entropy of the variable, and the formula for the set is given by

$$\mathcal{A}(X^n, \varepsilon) = \left\{ x^n \in X^n : 2^{-n(\mathcal{H}(X)+\varepsilon)} \leq P(X^n = x^n) \leq 2^{-n(\mathcal{H}(X)-\varepsilon)} \right\} \tag{2.14}$$

This formula defines the typical set for a sequence of random variables X^n with n elements. The set contains all sequences x^n from the sample space $\mathcal{X}^n$ for which the probability $P(X^n = x^n)$ falls within a certain range based on the entropy $\mathcal{H}(X)$ and a small tolerance ε. Therefore, the set has a total probability close to one, which is a consequence of the asymptotic equipartition property (AEP) [37]. There is a second central relationship between the size of the typical set $|\mathcal{A}(X)|$ and the entropy of the feature distribution [37], which is given by

$$\log |\mathcal{A}(X)| \leq f(\mathcal{H}(X)), \tag{2.15}$$

where $f(\cdot)$ is a monotonically increasing function which satisfies certain constraints. This is particularly useful as it directly links the entropy of a random variable X to the size its distribution occupies in space and hence its compactness (cf. Sec. 2.3.1).

2.4.3 Mutual information

Mutual information a is a non-linear correlation measure for the statistical dependence between two random variables [37]. It quantifies how much knowing the value of one variable reduces uncertainty about the other variable. Mutual information is denoted by the symbol $\mathcal{I}(X, Y)$, where X and Y represent the two random variables. It is calculated using the joint probability distribution of the variables and the individual probability distributions. Higher mutual information indicates a stronger relationship between the variables, with more shared information, whereas lower mutual information indicates differences or information loss. For example, lossy compression reduces mutual information between input and code, reducing information content and descriptiveness (cf. Sec. 2.3.1). The formula for mutual information is given by

$$\mathcal{I}(X, Y) = \sum_{y \in Y} \sum_{x \in X} P(x, y) \log \left(\frac{P(x, y)}{P(x)P(y)} \right). \tag{2.16}$$

The same connection can also be expressed through relative entropy

$$\mathcal{I}(X, Y) = D_{\mathrm{KL}}(P(X, Y) \| P(X)P(Y)), \tag{2.17}$$

which measures the difference between the joint and marginal distributions by definition. Generally, mutual information is introduced with only two variables. However, there exist several extensions to multiple variables. One extension that is relevant for this thesis is total correlation [187] that is given by

$$\mathcal{I}(X_1, ..., X_n) = \sum_{i=1}^{n} \mathcal{H}(X_i) - \mathcal{H}(X_1, ..., X_n). \tag{2.18}$$

It measures shared information between variables as the difference between the sum of individual and joint entropy.

CHAPTER 3

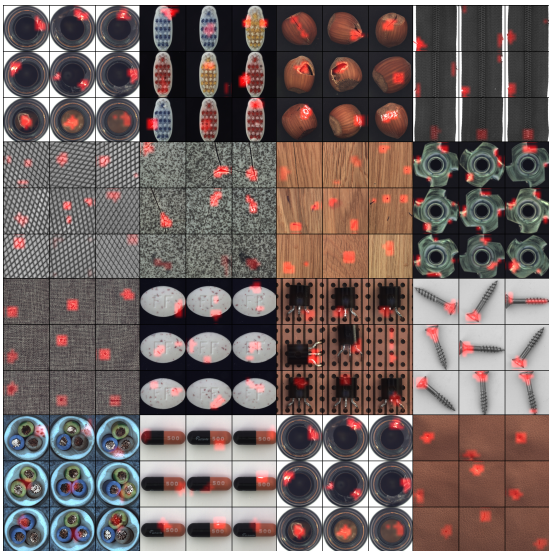
Novelty detection based on mean-shifts

The first chapter approaches the targeted inspection problem using a novelty detection approach. Here, the weakly labeled training data only consists of a single class. In the concrete case of industrial inspection, this training class consists of defect-free examples. The following chapter is based on the μshift publications [66, 67] and describes the necessary steps for model training and model testing.

3.1 Introduction

The human visual system's ability to detect unusual patterns in images is an important capability. Humans can differentiate between expected variance in the data and outliers after having only seen examples of normal instances. This chapter addresses the computer vision approach to this problem, usually known as image novelty detection. Novelty detection is related to outlier detection in the sense that both methods try to detect anomalies. However, while the latter is totally unsupervised, novelty detection has access to a training dataset consisting of clean normal reference data and hence is an instance of weakly supervised learning. The output of such an algorithm is a scoring function (anomaly score) that can be used to grade test data from inlier (normal) to outlier (novel) (e.g., [152]). Since the anomaly score is computed for a single input example, it can also be used for binary classification tasks. The major difficulty of such a model is that the decision boundary is not robust against overlapping between inlier and outlier distributions. This motivates the main idea of our ensemble approach to novelty detection: representing both training and test images as ensembles of image patches [66]. Instead of scoring a single test example with respect to the normal distribution, the ensemble approach first transforms the test example into an ensemble of patches and checks the test and training ensemble against each other, which improves the robustness of the decision process. There is a wide range of methods for testing if two samples originate from the same distribution. Here, we use the Hotelling T^2 test [72] for assessing the mean shift between two populations. As this statistic is simple to compute, fast, and effective, it is particularly suitable for large datasets.

Novelty detection by means of mean shifts is only possible when the inlier and outlier distributions are sufficiently spatially separate from each other. We found in our experiments that the degree of separation strongly depends on the computational details of how the image patches are extracted from the input images. Hereby, the critical hyperparameters are the size and the number of extracted patches and the selected feature map that is used for transforming the patches. We also



(a)

(b)

Figure 3.1: Detection of (a) locally concentrated novelty and localization on the MVTec dataset [14] and (b) global anomalies on the CIFAR-10 dataset. All shown examples are from the outlier test set and the red color highlights the location of the novelty in the images. The overlayed red score map is computed using the μshift anomaly score (cf. Eq. 3.18) without applying the spatial max-operator, such that the model output is a 2D grid of anomaly scores. These scores are mapped to a red heat map and resized to match the input resolution using bilinear interpolation. Hence, red areas correspond to potentially anomalous regions.

found that these parameters depend on the specific problem at hand and thus have to be set based on available domain knowledge. There are two cases to consider (cf. Fig. 3.1):

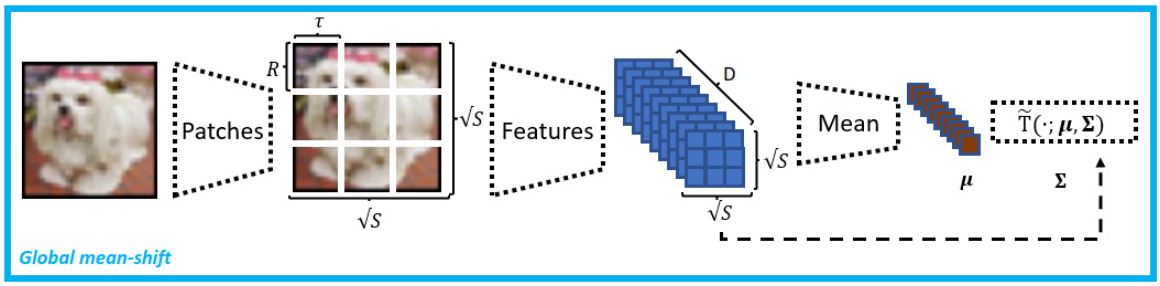
- **Global novelty** is spread across the entire image, e.g., when separating dog images from cat images.

- **Local novelty** appears only in some parts of the image, whereas the other parts of the image are totally normal, e.g., detecting tiny manufacturing defects in industrial visual inspection systems.

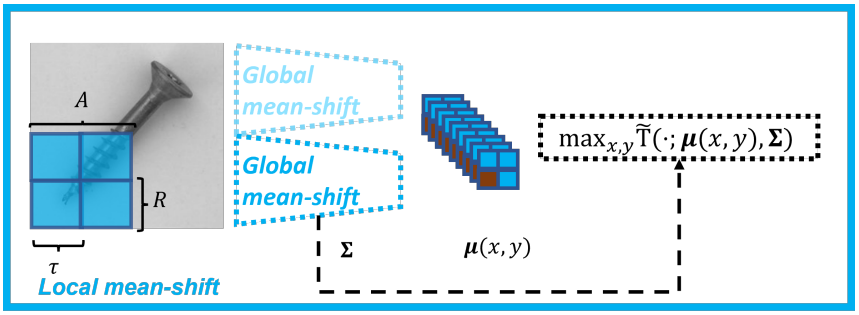
We identified the following practical principles for successful image novelty detection using mean shifts: (1) First, as anomalies mostly consist of patterns not available in the normal class, a *rich* feature space, such as a pre-trained neural network, needs to be used. (2) No dimension reduction based on the inlier data should be applied, as the inlier data occupies only a small portion of the feature space, and projecting onto its subspaces causes the anomalies to overlap with the normal data. And (3), the spatial size of the expected anomalies needs to be correctly expressed in terms of the hyperparameters, i.e., patch size and local mean-shift region, as a small local novelty cannot influence the mean shift sufficiently in a too-large averaging areas, mainly, because the distributions overlap only in insufficiently small regions.

Contributions. In this chapter, we propose a non-expensive algorithm [1] based on the Hotelling T^2 test for image novelty detection that is stacked on top of a standard pre-trained neural network, such as EfficientNet [173] or Vision Transformer (ViT) [46]. Using an upstream pre-trained neural network induces a *rich* feature space with a diverse set of pre-learned patterns and accommodates the previously mentioned principle (1). We follow principle (2) and use a full-rank covariance matrix for modeling the neural network features instead of relying on a compressed low-rank approximation which improves performance significantly. Further, to fulfill principle (3), we generalize the ensemble approach to novelty localization and add a hyperparameter that controls the expected spatial size of the anomalies, which has a strong impact on overall performance in practical applications. We show in extensive experiments that our approach not only achieves comparable results to existing state-of-the-art approaches but is also applicable to a large-scale industrial inspection scenario. Further, due to its simple architecture, the model has faster prediction times compared to existing approaches. Lastly, because we only need to estimate the mean of the training dataset, our method is very data-efficient and reaches 90 % AUC with only 10 non-defective examples of the MVTec dataset [14]. Fig. 3.1 shows examples from the evaluated datasets.

[1] `https://github.com/matherm/deep-mean-shift`

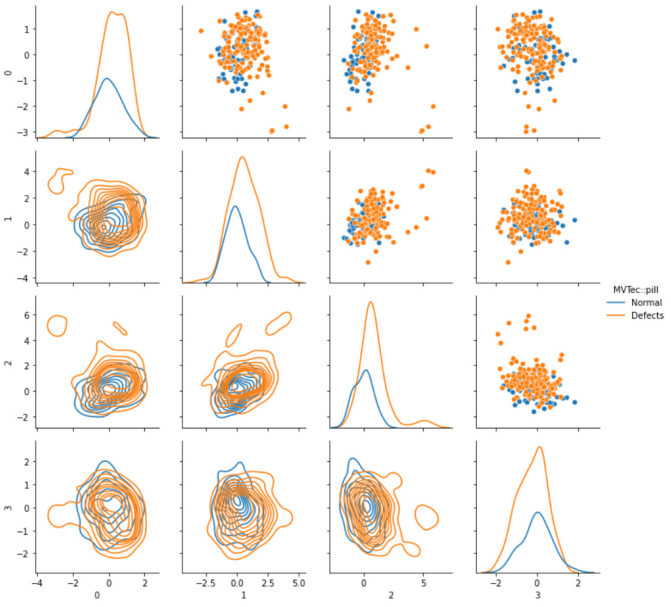


(a)

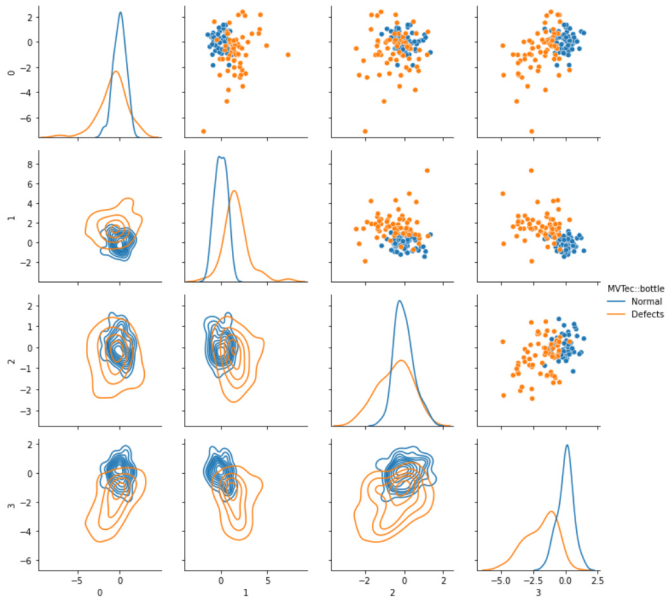


(b)

Figure 3.2: Schematic illustration of our mean-shift method. (a) First, the feature map ϕ is computed per extracted image patch, then the mean statistics are computed over all training patches. Together with the empirical covariance matrix of the normal data, the Hotelling T^2 test is used as an anomaly score. (b) The local mean-shift variant applies the global mean-shift method to a local region A of the image and hence yields a field of mean vectors $\boldsymbol{\mu}(x, y)$. The final score is computed by taking the maximum over the field of local scores. The covariance matrix is shared across all local regions.



(a)



(b)

Figure 3.3: Visualizing the mean-shift between normal and defective examples of the MVTec dataset for (a) the *Pill* and (b) the *Bottle* classes. For visualization, four random features of the EfficientNet-B4 features were chosen. As hyperparameters, we selected $\theta = \{L = 5, R = 48, \tau = 16, A = 96\}$.

3.2 Related work

The use of limited supervision for image classification has been studied extensively [175, 186]. Some approaches (e.g., [131]) consider the unbalanced setting where a small number of anomalous examples is given, but many examples are given from the normal class. However, these approaches use additional supervision that is not used in our method. The presented method relates more closely to anomaly detection approaches that use limited to weak supervision [152]. During training, we only use examples from the normal class and therefore consider our method an instance of novelty detection, a semi-supervised version of anomaly detection, sometimes also referred to as one-class classification [136]. There are different approaches to the problem in general, and we, therefore, group the related methods into the categories of reconstruction-, classification-, distribution-based, and self-supervised methods (cf. Sec. 2.3.1).

Reconstruction-based methods. These methods derive a data-driven encoder and decoder from the reference data and expect the anomalous data to have a higher reconstruction error compared to normal data. However, such models are mostly based on unconstrained compression and, therefore, often oversee novel patterns, resulting in poor performance in practice [146].

Classification-based methods. These methods attempt to model a discriminating hyperplane between data regions of normal data and those of anomalous data [151] without necessarily using compression. Such methods often perform well in practice. However, their main limitation arises from the fact that the hyperplane can only be estimated accurately in regions occupied by the training examples [54]. The recently proposed Mahalanobis method [146] tries to heal the problem by negating the estimation process by using the null space of a pre-trained neural network feature space instead.

Distribution-based methods. These methods are another branch of novelty detection that model the distribution of the normal data. Such methods are often built around autoencoders [7] or normalizing flows [150]. However, it has been argued and empirically found that distribution-based methods that fit a flexible parametric distribution with the maximum likelihood objective may not be well-suited for detecting out-of-distribution data [197].

Self-supervised methods. These methods try to improve distribution-based methods by replacing the data likelihood with a proxy classification objective, such that classifying normal data based on that objective allows for a good separation of normal and anomalous data. These techniques are related to non-linear independent component analysis (ICA) using an auxiliary variable, such as a time segment, a generalized non-stationary variable, or synthetic labels [76, 81]. A successful application of this theory to images is to predict image rotations [52, 51]. The proxy objective is given by first rotating the image by an arbitrary angle and then trying to predict that angle using a deep convolutional neural network. However, this strategy only works well for aligned objects with a *natural* orientation, where the rotation dependence is strong enough to learn a good rotation predictor.

In the literature, there are mostly specialized algorithms for either global or local novelty detection, and hence there are different methods superior within each scenario. For global novelty, particular rotation prediction [52], Deep SVD [151], and Deep Robust One Class Classifier (DROCC [54]) excel. The rotation prediction method is a self-supervised scheme that solves a proxy classification problem for feature learning and uses a softmax-based anomaly score. While the Deep SVD is a deep learning-based version of the singular value decomposition (SVD), the DROCC method uses

a nearest neighbor approach on pre-trained neural network features. For local defect detection, a recent method named PatchCore [148] achieves almost total recall on the MVTec challenge [14] using a greedy algorithm for dataset reduction based on coreset theory [163]. It is also based on an underlying nearest neighbor search in the feature space of a pre-trained WideResnet-50 [194] but uses a modified distance as an anomaly score. CutPaste [106] is a self-supervised method specially designed for local novelties. It is similar to rotation prediction [52] as it also solves a self-supervised surrogate classification problem. However, instead of predicting the rotation of the input example, it cuts out small patches and pastes them to another image location to create the contrastive dataset. The anomaly score is the class probability of being an altered image.

The method presented in this chapter is most related to recent works that model the internal distribution of images [150]. However, unlike these approaches, our model benefits from the estimation of only low-order cumulants of image patches, i.e., mean and covariance, instead of a parametric model of the full density, which is a simpler task in general. By choosing non-linear basis functions for representing the patches, here a pre-trained deep neural network, our method adds a relatively small computational overhead compared to our basic method based on raw pixel mean-shifts [66], but improves the detection and localization performance effectively (cf. Fig. 3.9).

There are two similar approaches named PatchSVDD [191], and PaDiM [41] that we want to relate shortly. PatchSVDD optimizes a deep spherical embedding for extracted image patches. As this is based solely on the reference data, it implicitly reduces dimensionality, and novelties with orthogonal patterns are projected onto the *null space* of the normal distribution. This decreases the performance of the method compared to our approach, which does not involve any dimension reduction. PaDiM is similar to ours and also computes full-rank Mahalanobis distances. However, it does not benefit from extracted patch ensembles that turned out to be performance critical in our tests. Therefore it is a special case to ours where the size of the extracted patch ensemble is one, and the covariance matrix is not shared across locations.

3.3 Deep μshift

The central part of our deep μshift algorithm is the μshift($\mathbf{x}$) anomaly score [66] that is based on the Hotelling T^2 test [72]. This test is formulated on the basis of two samples of two distributions and measures the mean shift between them. Therefore, it is a multivariate extension of the well-known Student's t-test. Here, we use ensembles of image patches as samples and measure their mean shift in some specified feature space Φ. In this section, we first describe the data representation and the mean-shift detection in its classical form.

On a high level, the first step is the extraction of patches $\{\mathbf{I}_0(s), \ldots, \mathbf{I}_N(s)\}$ from the normal training images $\{\mathbf{I}_0, \ldots, \mathbf{I}_N\}$. These patches are transformed by a feature map ϕ, typically parametrized by a pre-trained neural network. For indexing the ensemble where necessary, we introduce the indexing variable s. Using this notation, a single extracted patch of the i-th example in feature space is denoted by $\mathbf{x}_i$ and the corresponding patch ensemble by $\mathbf{x}_i(s)$. Based on all available transformed training patches $\mathbf{X}$, the required statistics, i.e., mean vector and covariance matrix, are computed. For a given test image $\mathbf{I}^*$, the same preparatory steps are applied, and we extract an ensemble of patches $\mathbf{I}^*(s)$ and compute the features $\mathbf{x}^*(s)$. We then evaluate the mean-shift of the test example by comparing the test ensemble mean $\boldsymbol{\mu}(\mathbf{x}^*(s))$ with the mean of the entire training dataset $\boldsymbol{\mu}(\mathbf{X})$.

3.3.1 Data representation

The input examples $\mathbf{I}_i \in [0,1]^{3\times H\times W}$, with $i = 1, ..., N$, are square-sized RGB images, i.e., $H = W$. The distinctive property of our algorithm is to generate patch ensembles instead of processing the full image. For patch extraction, we tested several sampling strategies without noticing performance-critical differences. Therefore, we extract all valid patches of size R inside the image. The term *valid* is used in accordance with the neural network literature and means that all extracted patches must be entirely contained within the image borders. This is equivalent to cropping patches by a sliding window without applying image padding or crossing the border. As the input images are potentially large, the horizontal and vertical stride τ of the sliding window allows limiting the total number of cropped image patches S. We fix this parameter to $\tau = 2$ for small images and $\tau = 16$ for larger ones. Hence, the maximum number S of distinct image patches per input image depends only on the size of the image and the patch size R, i.e., the larger the image relative to the patch size, the more patches can be extracted. We do not apply any pre-processing and compute a feature representation ϕ for the extracted patches $\mathbf{I}_i(s)$ using a pre-trained neural network, given by

$$\mathbf{x}_i(s) = \phi(\mathbf{I}_i(s)) \in \mathbb{R}^D, \tag{3.1}$$

where D is the number of features after flattening the computed feature map. Flattening is needed since some feature maps ϕ, e.g., Convolutional Neural Networks (CNN) [67] or raw pixels [66], retain the spatial dimensions of the input patches. We organize the flattened feature vectors of all available extracted normal training patches in a long concatenated design matrix $\mathbf{X} \in \mathbb{R}^{NS\times D}$.

3.3.2 Mean-shift detection

In our proposed method, we perform mean-shift detection with the Hotelling T^2 test [72]. Since this test is a generalization of the Student's t-test, it estimates the significance of mean shifts between two populations. In this section, we introduce the Hotelling T^2 test with its required statistics in the original form. In the second part of the chapter, in Sec. 3.4, we derive a generalized version that can transition smoothly between global and local population mean shifts. We discuss relevant hyperparameters that are needed for model selection in Sec. 3.3.4 and 3.4.1.

For detecting anomalies, we first compute the feature-wise mean

$$\boldsymbol{\mu} = \frac{1}{NS}\sum_{i}^{N}\sum_{s}^{S}\mathbf{x}_i(s) \tag{3.2}$$

over all extracted and flattened feature maps of the training dataset $\mathbf{X}$. Note that μ has the same dimension as $\mathbf{x}_i$. We then compare this reference mean with the mean

$$\boldsymbol{\mu}^* = \frac{1}{S}\sum_{s}^{S}\mathbf{x}^*(s) \tag{3.3}$$

of the transformed patches $\mathbf{x}^*(s)$ extracted from a single test example $\mathbf{I}^*$. Given the two estimated mean vectors $\boldsymbol{\mu}^*$ and $\boldsymbol{\mu}$, the unnormalized Hotelling T^2 test statistic for a dependent test sample is computed by

$$\tilde{T}^2 = (\boldsymbol{\mu}^* - \boldsymbol{\mu})^T\hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{\mu}^* - \boldsymbol{\mu}), \tag{3.4}$$

where

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{NS-1}(\mathbf{X} - \boldsymbol{\mu})^T(\mathbf{X} - \boldsymbol{\mu}) \tag{3.5}$$

is the empirical covariance matrix of the training dataset $\mathbf{X}$. There is an intuitive geometric interpretation of the $\tilde{T}^2$ statistic available. That way, it can be interpreted as the squared Mahalanobis

distance [113] between the two estimated mean vectors. For completeness, we want to highlight that we discarded the constant normalization factor $\frac{NS^2}{NS+S}$ that appears in the original formula and hence denote our unnormalized version of the statistic by $\tilde{T}^2$ instead.

In principle, there are several options for defining the mean $\boldsymbol{\mu}$ of the reference data, e.g., by clustering or partitioning. Here, we choose the simplest option and compute the feature-wise mean over all patches of the training examples. This gives a single $\boldsymbol{\mu}$-vector for the entire dataset as denoted in Eq. 3.2. A naive global anomaly score is simply defined as the unnormalized T^2 test statistics over the entire image

$$\tilde{s}(\mathbf{x}) = \tilde{T}^2(\boldsymbol{\mu}^*(\mathbf{x}); \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.6}$$

The entire pipeline of this global method is illustrated in Fig. 3.2 on the left-hand side.

3.3.3 Covariance shrinkage

The empirical covariance matrix in Eq. 3.5 cannot be robustly estimated for high dimensional data as most of the eigenvalues are close to zero, and hence the estimates are very unstable. This is especially an issue for small datasets, where the number of patches is equal to or smaller than the covariance matrix dimension D, but is potentially also a problem for highly redundant datasets that occupy only a small subspace. In order to mitigate, we use the Ledoit-Wolf shrinkage estimator [102]. This estimator is given by a convex combination between a scaled identity matrix and the empirical covariance matrix

$$\boldsymbol{\Sigma} = (1-\alpha)\hat{\boldsymbol{\Sigma}} + \alpha\frac{\operatorname{trace}(\hat{\boldsymbol{\Sigma}})}{D}\mathbf{Id}, \tag{3.7}$$

where the so-called shrinkage factor $\alpha \in [0, 1]$ is given analytically by minimizing the quadratic loss between the true and estimated covariance matrix. The exact formula for α is a bit cumbersome, and we, therefore, refer the reader to Eq. 5 in the original paper for it [102]. Loosely speaking, the shrinkage factor α is an analytic function of the empirical covariance matrix and the number of data points. A useful property of the estimator is that the shrinkage factor is near one for small numbers of data points and reduces to zero with increasing data set size. Therefore, the shrunk covariance matrix converges to the true covariance matrix in the limit of an infinite number of data points. Our experiments show that the chosen estimator is crucial and responsible for almost 5 % of the overall performance. Fig. 3.4 shows the impact of the shrinkage factor on novelty detection performance for different values of α and different dataset sizes. For the experiment, we used 64×64 patches and the EfficientNet-B4 feature space, which has dimensionality $D = 6800$ after the flattening operation (cf. Eq. 3.1). Therefore, the estimation of a covariance matrix with shape 6800×6800 is required.

As already noted, outliers are projected onto the subspace spanned by the normal data, which makes a robust estimation of the full covariance matrix necessary without the possibility of using low-rank approximations [146]. This is especially important when the expected anomalies are small and characterized by patterns that are not present in the training set.

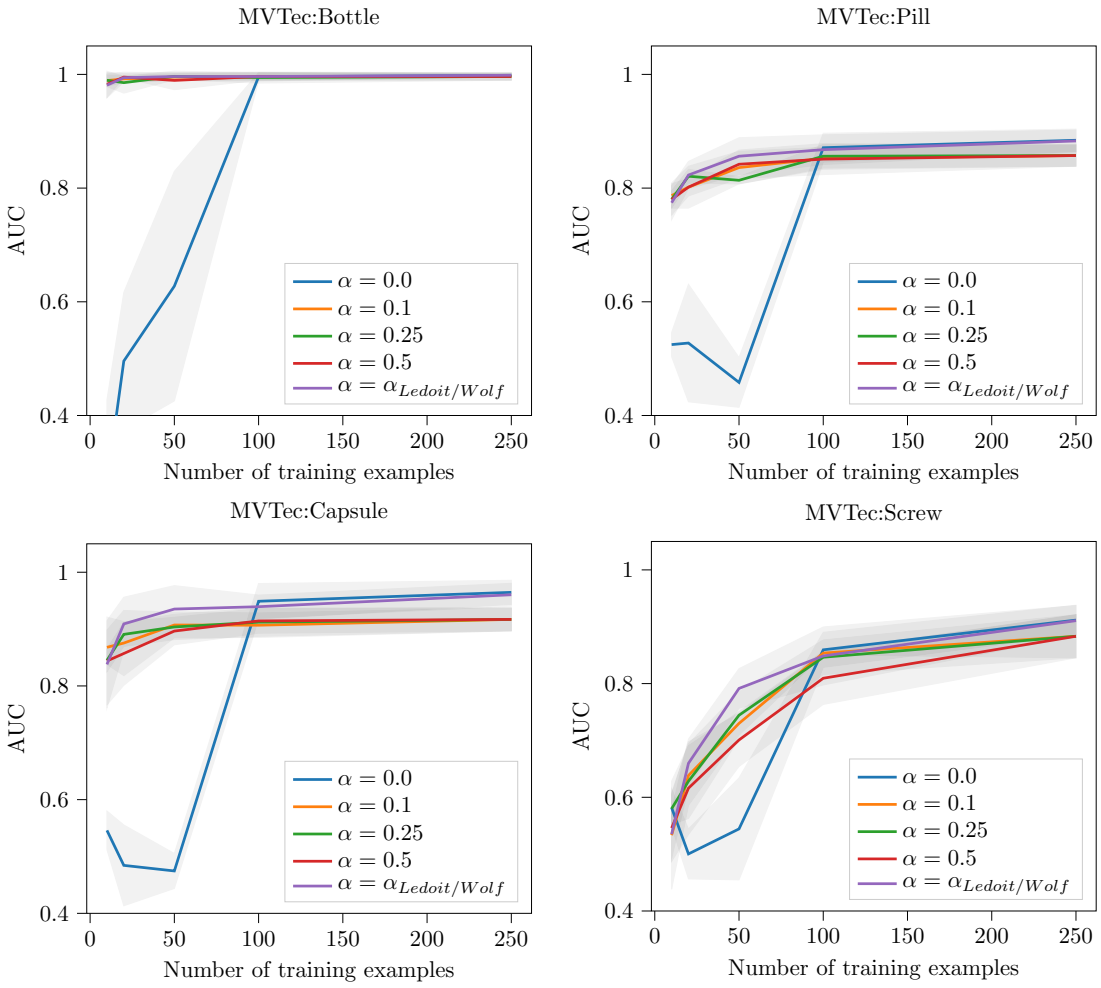


Figure 3.4: Impact of the the covariance shrinkage factor α on novelty detection performance. We selected four representative classes from the MVTec dataset [14] for evaluation. Due to the high dimensionality of the covariance matrix ($D = 6800$), the effect of the shrinkage factor is largest when the number of training examples is small. The Ledoit-Wolf shrinkage $\alpha_{Ledoit/Wolf}$ varies across experiments between $[0.01, 0.1]$ and clearly improves the average performance.

3.3.4 Hyperparameter selection

There are two main hyperparameters that need to be set for the global method. First is selecting the feature space Φ, and second is choosing the patch size R. In the following, we first discuss the feature spaces and how the chosen neural network architectures differ and impact the model.

Feature space Φ

Our model does not learn features and relies on a fixed *rich* feature representation ϕ for the input image patches $\mathbf{I}_i(s)$. To this end, we analyzed different deep convolutional neural networks (CNNs), namely EfficientNet-B4 Φ_L^{eff} [173], Wide-Resnet-50 Φ_L^{res} [194], and VGG-19 Φ_L^{vgg} [168]. L indicates the feature block (layer) of the deep neural network. The commonly used block convention wraps several adjacent layers of a deep neural network into blocks, such that the architectures become handier and easier to compare (e.g., [168]). We follow this convention, and the hyperparameter L indexes entire blocks of the architecture. The superscript indicates the used network architecture, e.g., Φ_3^{vgg} for the third feature block of VGG-19. Additionally, we analyzed the recently presented Vision Transformer Model (ViT) Φ_L^{vit} [46]. The required pre-training of the networks is always done by using the well-known ImageNet dataset, where all architectures reach a test set accuracy of around 85 %. Note, we concatenated two adjacent layers $[L, L+1]$ of EfficientNet-B4 as the layers are relatively low-dimensional, which improves the performance slightly (see, e.g., [148]). Due to the pooling layers, the spatial resolution of the feature map decreases with the depth of the network, i.e., deeper layers have lower spatial resolution. To enable channel-wise concatenation of different-sized feature maps in the first place, we match the spatial resolution of the feature maps by spatially resizing the smaller downstream feature map to the size of its larger predecessor feature map using bilinear interpolation.

Patch size R

The most crucial hyperparameter is the chosen image patch size R. Generally, the deeper the convolutional neural network, the smaller the spatial resolution of the resulting feature map, which is mainly caused by 2D-pooling operations [111]. While the receptive field grows, more global information is carried by the pixels of the feature maps. Importantly, for successful feature extraction, one needs to choose a layer that retains *enough* spatial resolution for the problem at hand and an appropriate receptive field to capture the anomalies. Hence, to gain sufficient separate inlier and outlier distributions, the layer selection L and patch size R depend on the size of the input images and the expected size of the anomalies. Note that the Vision Transformer (ViT) is different as the receptive field size is implicitly learned by the model and, in principle, equal to the size of the entire input and hence independent of the layer L. We did not notice a performance-critical impact of the stride parameter and kept it fixed to $\tau = 2$ for smaller inputs and $\tau = 16$ for larger ones.

Optimal patch size R using typical set theory

Depending on the patch size R, the appearance of the classes changes drastically, particularly in terms of dissimilarity between neighboring image patches. Fig. 3.5 shows the visual impact of the patch size on the ensemble statistics. This observation motivates the use of an entropy-related measure of dissimilarity or disorder for hyperparameter selection.

Feature space interpretation of T^2 For this method, a feature space interpretation of the T^2-statistic is first needed. The Mahalanobis distance in Eq. 3.4 can be computed by first whitening the data with a whitening transformation $\mathbf{A}$ and then computing the standard L_2 distance of the

whitened mean vectors. This allows us to reveal relevant hyperparameters, such as the noise floor and the rotation freedom. Due to the linearity of $\mathbf{A}$, this is equivalent to applying $\mathbf{A}$ to the mean difference vector in the input feature space:

$$\tilde{T}^2 = \|\mathbf{A}(\boldsymbol{\mu}^* - \boldsymbol{\mu})\|_{L_2} \tag{3.8}$$

The whitening transformation $\mathbf{A}$ can be decomposed into an orthogonal matrix $\mathbf{W}$ containing the Eigenvectors of the covariance matrix $\boldsymbol{\Sigma}$ as columns, a diagonal scaling matrix $\mathbf{S}^{-1/2}$, and an arbitrary rotation matrix $\mathbf{R}$, such that

$$\mathbf{A} = \mathbf{R}\mathbf{S}^{-1/2}\mathbf{W}, \tag{3.9}$$

with $\boldsymbol{\Sigma} = \mathbf{W}^\mathbf{T}\mathbf{S}\mathbf{W}$, $\mathbf{W}\mathbf{W}^\mathbf{T} = \mathbf{I}$, and $\mathbf{R} \in \mathrm{SO}(M)$. The matrix $\mathbf{S}$ consists of the variances $s_i, \cdots, s_M$ along the components in $\mathbf{W}$.

In this transformation, we also reduce the data dimension M to k by removing the noise floor. This is done by truncating the matrices $\mathbf{W}$ and $\mathbf{S}$ by removing the dimensions with the smallest variance. Across experiments, we found it helpful to control the number of informative variables k by a rule instead of fixing the number of retained features. We retain all components up to a fixed threshold of explained variance [80], in our case 90 %. We denote this number by $k = k_{90}$. For visualization, it is useful to decompose the $\tilde{T}^2$ statistic into the feature vector

$$\phi(\boldsymbol{\mu}^*) = \mathbf{A}\boldsymbol{\mu}^* \tag{3.10}$$

and the derived modified anomaly score

$$\mu\tilde{\text{shift}}(\mathbf{x}) = \|\phi(\boldsymbol{\mu}^*) - \phi(\boldsymbol{\mu})\|_{L_2}\,. \tag{3.11}$$

Fig. 3.6 shows the mean and variance statistics of ϕ for the deer class of CIFAR-10.

Typical set minimization It is known that whitened data stay whitened under rotation, so we can apply an arbitrary rotation matrix $\mathbf{R}$ without changing the $\tilde{T}^2$ statistic. Since we do not observe the outliers, we can only manipulate the statistics of the transformed reference data. In the introduction, we argued that a good strategy for model selection is to keep the size $|\mathcal{A}(\cdot)|$ of the typical set as small as possible, as this limits the blind spot of the mean shift detection mechanism. A central relationship between the size of the typical set and the entropy of the feature distribution [37, 124] is

$$\log|\mathcal{A}(\phi)| \leq f(\mathcal{H}(\phi)), \tag{3.12}$$

where $f(\cdot)$ is a monotonically increasing function which satisfies certain constraints. This means that in order to keep the *blind spot* small, we need to minimize the entropy of ϕ. Directly minimizing the entropy of stochastic variables is heavily studied in the field of sparse coding and independent component analysis [35]. A central measure in that field is the so-called negentropy, which is the negative of entropy. Negentropy has an appealing feature that arises from the maximum entropy principle, i.e., given a fixed variance, the maximum entropy distribution is a Gaussian [37]. This relation can be utilized by the construction of a negentropy approximation [35] that uses the Gaussian distribution as a contrast

$$\mathcal{J}[\phi] \propto \sum_i^k (g(\phi_i) - g(\gamma))^2, \tag{3.13}$$

where $g = \log\cosh(\cdot)$, $\gamma \sim \mathcal{N}(0,1)$, and ϕ is centered. As a consequence, the model selection rule simplifies to a linear search over the patch size P and a non-convex optimization of the rotation matrix $\mathbf{R}$,

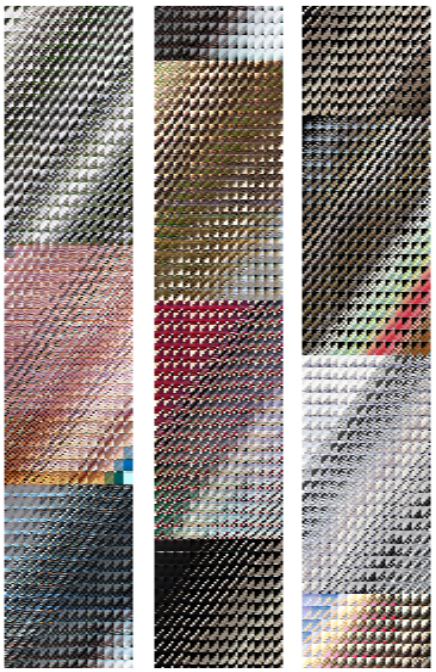
$$\arg\max_{R,\mathbf{R}} \mathcal{J}[\phi], \tag{3.14}$$

	μ**shift**$^{\text{eff}}$	μ**shift**$^{\text{vit}}$	μ**shift**$^{\text{vgg}}$	**RotNet** [52]	**DROCC** [54]	**Mah.Ad** [146]	**Deep-SVM** [160]	**DSVD** [151]
plane	0.776±0.004	**0.948±0.005**	0.853±0.002	0.739±0.006	0.817±0.022	0.745±0.006	0.718±0.020	0.617±0.410
car	0.858±0.008	**0.979±0.003**	0.896±0.010	0.905±0.013	0.767±0.099	0.748±0.008	0.712±0.003	0.659±0.210
bird	0.650±0.002	**0.942±0.005**	0.725±0.005	0.773±0.006	0.667±0.096	0.630±0.008	0.606±0.008	0.508±0.080
cat	0.613±0.011	**0.923±0.008**	0.673±0.015	0.741±0.013	0.671±0.151	0.657±0.008	0.643±0.019	0.591±0.140
deer	0.819±0.009	**0.955±0.005**	0.856±0.002	0.792±0.015	0.736±0.200	0.737±0.002	0.788±0.008	0.609±0.110
dog	0.706±0.013	**0.970±0.006**	0.758±0.002	0.848±0.013	0.744±0.195	0.706±0.004	0.661±0.006	0.657±0.250
frog	0.886±0.006	**0.970±0.003**	0.884±0.001	0.793±0.013	0.744±0.092	0.766±0.004	0.786±0.014	0.677±0.260
horse	0.843±0.016	**0.972±0.002**	0.858±0.010	0.915±0.006	0.714±0.022	0.757±0.012	0.704±0.013	0.673±0.090
ship	0.806±0.012	**0.976±0.005**	0.902±0.010	0.906±0.008	0.800±0.169	0.744±0.000	0.785±0.003	0.759±0.120
truck	0.855±0.007	**0.975±0.004**	0.923±0.013	0.885±0.010	0.762±0.067	0.779±0.006	0.796±0.014	0.731±0.120
Avg.	0.781±0.002	**0.961±002**	0.833±0.000	0.830±0.004	0.742±0.011	0.727±0.003	0.720±0.009	0.648±0.180

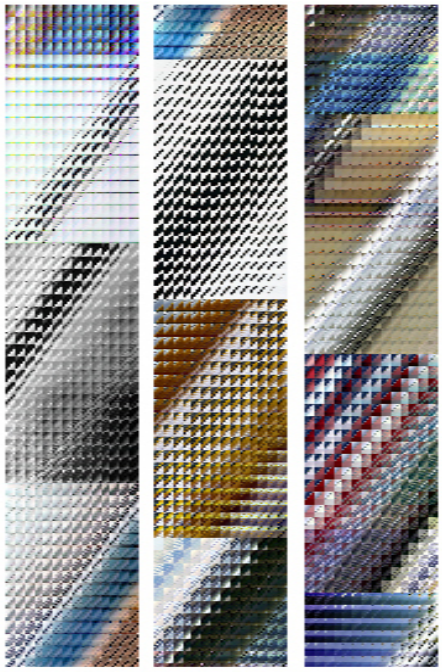
Table 3.1: Evaluating detection capabilities of globally distributed novelty on the CIFAR-10 dataset using five-fold cross-validation. Changing the CNN feature space impacts the detection performance significantly. We argue that this indicates a strong correspondence between novelty size and the size of the receptive field of the CNN architecture.

with $R \in [14, \sqrt{D/c} - \tau]$ and $\mathbf{R} \in \mathrm{SO}(k)$. We chose 14 as the minimum patch size to avoid the pathological case of selecting too small patches containing zero image content, such as black spots in MNIST. Again, the rotation matrix $\mathbf{R}$ needs only to be optimized as this model freedom highly impacts the negentropy measure but does not change the Mahalanobis distance (Eq. 3.4). While a grid search finds R, optimizing the rotation matrix $\mathbf{R}$ is a non-convex problem. In particular, the solution is constrained to be an orthogonal matrix.

The solution to this problem is known as independent component analysis (ICA), which is a well-known technique in signal processing. Note that our problem is slightly different from standard ICA applications, as we are optimizing with respect to the average across multiple patches instead of a single patch. However, this is just a simple preprocessing step, and the standard ICA algorithms, such as FastICA [78], can be applied afterward. For negentropy estimation problems, where the design matrix $\mathbf{X}$ does not fit into memory, we propose a specially tailored algorithm in Chapter 4

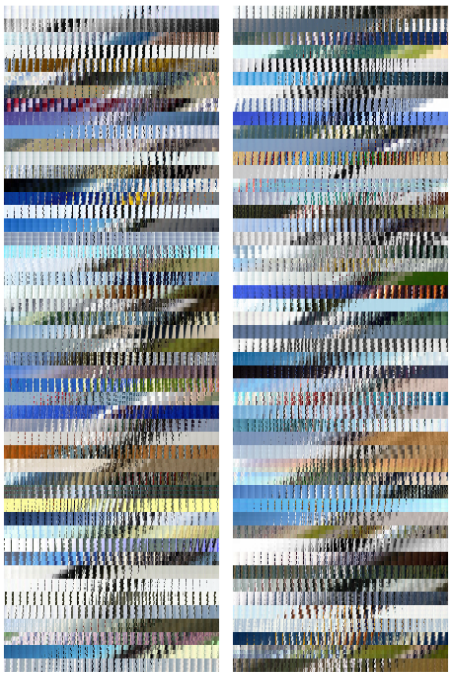


(a) Cats $(14, 14)$, $\mathcal{J}[\phi] = 0.019$.



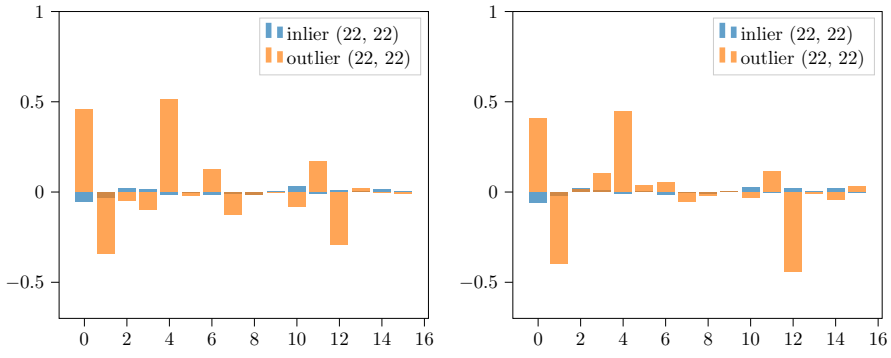
(b) Planes $(14, 14)$, $\mathcal{J}[\phi] = 0.030$.

(c) Cats $(30, 30)$, $\mathcal{J}[\phi] = 0.017$.

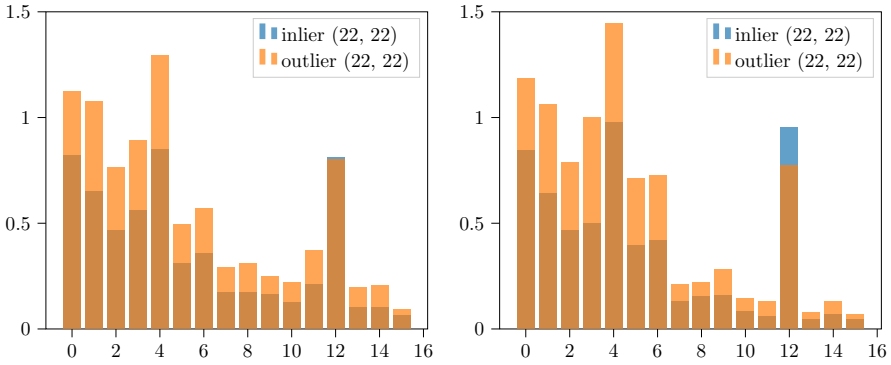


(d) Planes $(30, 30)$, $\mathcal{J}[\phi] = 0.031$.

Figure 3.5: Patch ensembles from two classes of CIFAR-10, cats, and planes, with two different patch sizes. The plane class has a more homogeneous appearance (e.g., blue sky), while the cat class is more chaotic (e.g., cat pose), yielding a smaller negentropy $\mathcal{J}$.



(a) $\mathbb{E}[\phi]$



(b) $\mathrm{Var}[\phi]$

Figure 3.6: Mean-shifts and variances of the first 16 principal components of ϕ for the deer class of CIFAR-10 with patch size $P = 22$. The left half of (a) and (b) shows the statistics with unoptimized rotation matrix $\mathbf{R}$, the right half after optimization.

3.3.5 Evaluating global novelty detection

We compare our method with methods that particularly excel in global novelty detection. As a baseline, we use the well-known OC-SVM [160] with an RBF kernel and flattened CNN feature maps. We reproduced all experiments by either using implementations provided by the authors or re-implementing the models by using available information and hyper-parameters. For testing global novelty detection, we use the CIFAR-10 dataset [95], which are 32×32 RGB images, and test the methods in a one-vs-all procedure. This means we use the 5000 available training examples of a single class as the normal class and classify the entire test dataset, consisting of 10 classes with 1000 examples each, afterward. We use area-under-the-ROC-curve (AUC) as a performance measure (cf. e.g., [52, 146]). The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various thresholds and hence measures the overall discrimination performance of a binary classifier. In terms of hyperparameters for μshift, we selected $\theta = \{L = 5, R = 32, \tau = 2\}$. This is the maximum patch size possible and a special case of the method. However, it is also optimal for the chosen scenario: Reducing the patch size decreases the AUC significantly for most of the network architectures (cf. Fig. 3.9). The same applies to the parameter L, which we verified in Fig. 3.7. Unfortunately, for deep features, experiments showed that hyperparameters could not be set by the hyperparameter selection method based on typical set minimization (cf. Sec. 3.3.4). We believe that the problem for deep features is the high-dimension of the feature space, which is mostly much higher than 1000 dimensions.

Tab. 3.1 shows the results averaged across five folds of cross-validation using varying training and test splits. It is interesting to note how the different CNN architectures strongly influence the performance and how the Vision Transformer (ViT), with its large receptive field, can separate the inliers from the outliers almost entirely. Particularly using EfficientNet-B4 is problematic in the special case of global novelties as it possesses the smallest receptive field among the tested architectures and is not able to capture the entire image context into a single feature variable.

We presented the original mean-shift method [66] based on raw pixel values and evaluated the performance without using pre-trained features or transfer learning. Tab. 3.2 and Tab. 3.3 show baseline results from other standard models, such as standard one-class support vector machine (OC-SVM) [160], kernel density estimation (KDE), variational autoencoder (VAE) [92], and latent space autoregression (LSA) [1]. We used raw pixels for all methods instead of pre-trained neural network features for a fair comparison. For μshift on raw pixels, hyperparameters were set by the hyperparameter selection method based on typical set minimization (cf. Sec. 3.3.4).

For completeness, we also report the low average AUC of only 0.675 using raw pixel values in Fig. 3.9. This lack of performance compared to deep features emphasizes the requirement for a rich feature space, such that the inlier distribution does not overlap with novelties through its null space, causing a large *blind spot* for novelty detection. Such an overlap happens naturally when the anomalous patterns are projected onto the subspace of the normal data, and the corresponding features are not present in the given training data. Consequently, anomalous patterns cannot be detected as they are mapped to null space. A rich feature space with a diverse set of pre-learned patterns mitigates that effect.

With RotationNet, we could achieve the reported AUC of 0.86 only when the internal network got pre-trained on ImageNet [52], but not when initialized randomly. However, despite that, for general unsupervised feature learning, the method remains extremely powerful on CIFAR-10.

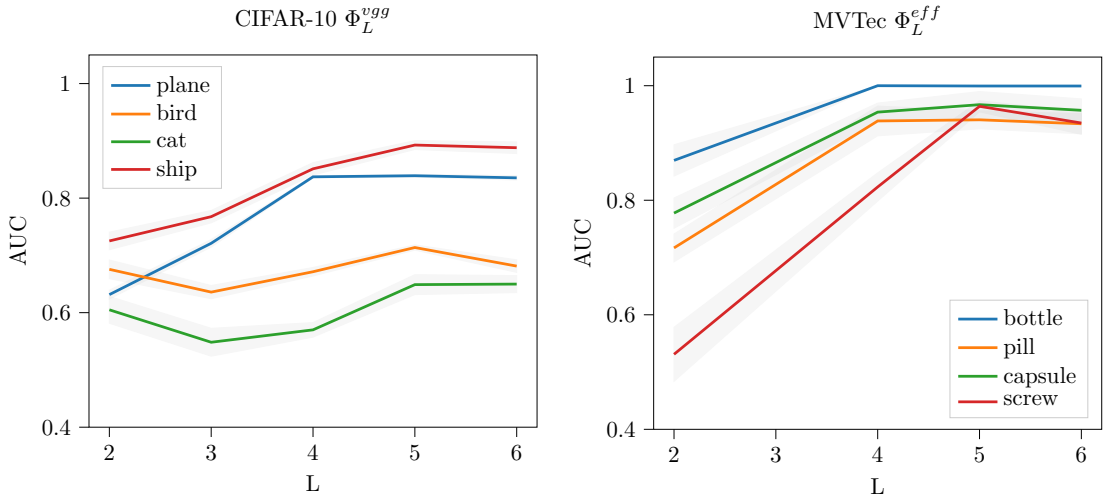


Figure 3.7: Impact of the chosen feature block index L of the neural network on the novelty detection performance. We selected four sensitive classes from the MVTec dataset [14] and the CIFAR-10 dataset [95] for evaluation. On average, a deeper block improves detection performance. The best results were obtained for $L = 5$.

	OC-SVM	**KDE**	**VAE**	**LSA**	**DSVD**	**μshift$^{\text{pixel}}$**
plane	0.630	0.658	0.688	**0.735**	0.617	0.731
car	0.440	0.520	0.403	0.580	0.659	**0.711**
bird	0.649	0.657	0.679	**0.690**	0.508	0.498
cat	0.487	0.497	0.528	0.542	0.591	**0.609**
deer	0.735	0.727	0.748	**0.761**	0.609	0.582
dog	0.500	0.496	0.519	0.546	**0.657**	0.620
frog	0.725	**0.758**	0.695	0.751	0.677	0.724
horse	0.533	0.564	0.500	0.535	0.673	**0.718**
ship	0.649	0.680	0.700	0.717	0.759	**0.805**
truck	0.508	0.540	0.398	0.548	0.730	**0.751**
Avg.	0.586	0.610	0.586	0.641	0.648	**0.675**

Table 3.2: Comparing detection capabilities of globally distributed novelty on the CIFAR-10 dataset with standard baselines. Here, for fair comparison, raw pixels are used for μshift.

	OC-SVM	**KDE**	**VAE**	**LSA**	**DSVD**	**μshift$^{\text{pixel}}$**
0	0.988	0.885	**0.998**	0.993	0.980	0.997
1	0.999	0.996	**0.999**	0.999	0.997	0.993
2	0.902	0.710	0.962	0.959	0.917	**0.986**
3	0.950	0.693	0.947	0.966	0.919	**0.979**
4	0.955	0.844	0.965	0.956	0.949	**0.971**
5	0.968	0.776	0.963	0.964	0.885	**0.981**
6	0.978	0.861	0.995	0.994	0.983	**0.995**
7	0.965	0.884	0.974	**0.980**	0.946	0.973
8	0.853	0.669	0.905	0.953	0.939	**0.969**
9	0.955	0.825	0.978	**0.981**	0.965	0.977
Avg.	0.951	0.814	0.969	0.975	0.948	**0.982**

Table 3.3: Comparing detection capabilities of globally distributed novelty on the MNIST dataset with standard baselines. Here, for comparison, raw pixels are used for μshift.

3.4 Local anomaly score

Due to global averaging, the naive global mean-shift anomaly score is not flexible enough to localize anomalies properly. To improve, a generalization of the mean-shift detection capabilities to local mean-shifts is required, and hence a modified test statistic. To this end, we define a local version by computing an entire field of $\boldsymbol{\mu}$-vectors uniformly distributed across the image instead of a single vector.

As shown on the right-hand side in Fig. 3.2, this is equivalent to computing the global anomaly score only for local parts of the image with a shared covariance matrix across all locations. To leverage the spatial structure, we organize the S extracted patches $\mathbf{x}(s)$ as a $\sqrt{S} \times \sqrt{S}$ feature map $\tilde{\mathbf{x}}(x, y)$, where the positions (x, y) correspond to their relative locations in the input image, i.e., the order of the patches and their relative spatial position is unchanged. Note that S is a square number because we assume that the input images are square-sized RGB images, i.e., $H = W$. Next, we compute the $\boldsymbol{\mu}$-vectors by averaging across a local neighborhood whose size is given by A. The resulting $\frac{\sqrt{S}}{\rho} \times \frac{\sqrt{S}}{\rho}$ field $\mathbf{S}$ consists of the $\boldsymbol{\mu}(x, y)$-vectors, with

$$\rho = \frac{A - R}{\tau} + 1. \tag{3.15}$$

As in the global case, the $\boldsymbol{\mu}(x, y)$ of the training data is computed by averaging over all available training examples

$$\boldsymbol{\mu}(x, y) = \frac{1}{N} \sum_{i}^{N} \mathbf{S}_{\tilde{\mathbf{x}}_\mathbf{i}}(x, y), \tag{3.16}$$

where $\mathbf{S}_{\tilde{\mathbf{x}}}$ is the pooled feature map

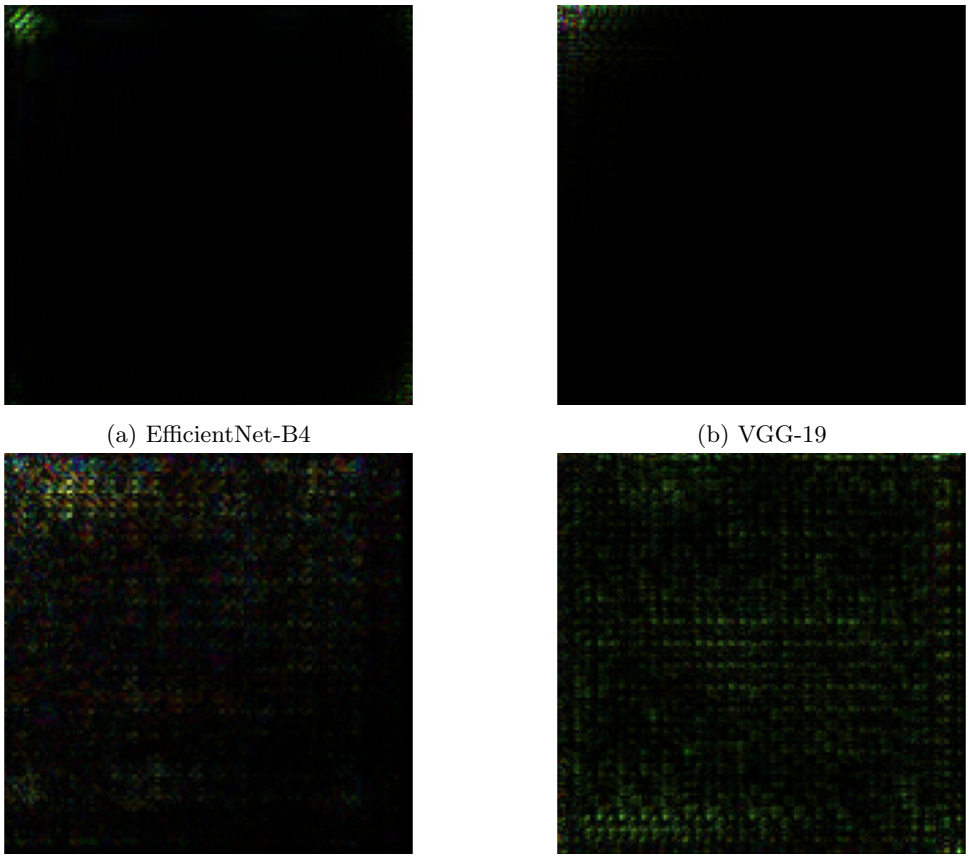
$$\mathbf{S}_{\tilde{\mathbf{x}}}(x, y) = \frac{1}{\rho^2} \sum_{m=1}^{\rho} \sum_{n=1}^{\rho} \tilde{\mathbf{x}}(x + m, y + n), \tag{3.17}$$

and $\tilde{\mathbf{x}}(x, y) \in \mathbb{R}^{D \times \sqrt{S} \times \sqrt{S}}$ is the reshaped version of $\mathbf{x}(s) \in \mathbb{R}^{S \times D}$.

The generalized anomaly score is then computed by taking the maximum over the field of local mean-shifts, given by

$$\mu\text{shift}(\mathbf{x}) = \max_{x,y} \tilde{T}^2(\mathbf{S}_{\tilde{\mathbf{x}}}(x, y) - \boldsymbol{\mu}(x, y); \mathbf{0}, \boldsymbol{\Sigma}). \tag{3.18}$$

Note that the covariance matrix $\boldsymbol{\Sigma}$ is exactly the same as in the global case and just the mean estimates are computed differently. In fact, for $\rho = \sqrt{S}$, the global case appears as a special case. A second special case appears when $R = H$ and hence $\rho = 1$. Here, the extracted patches represent entire images.



(a) EfficientNet-B4 (b) VGG-19

(c) InceptionNet (d) Wide-ResNet-50

Figure 3.8: Gradient maps with respect to the input $\mathbf{x} = \mathbf{0}^{1\times3\times150\times150}$ for popular architectures. For visualization, we computed $|\nabla_{\mathbf{x}} \sum \phi_{0,0}(\mathbf{x})|$ for the upper-left pixel of the feature map $L = 5$ using backpropagation.

	EfficientNet-B4			**VGG-19**			**Wide-Resnet-50**		
Block	Field	Stride	Features	Field	Stride	Features	Field	Stride	Features
1	2	2	48	2	2	64	2	2	64
2	2	2	24	6	2	128	4	4	64
3	4	4	32	14	6	256	4	4	256
4	8	8	56	14	6	512	8	8	512
5	16	16	112	30	14	512	16	16	1024
6	16	16	116	30	14	512	32	32	2048

Table 3.4: The minimal patch size R, the internal stride, and the number of features for different feature blocks of the analyzed convolutional neural networks (CNNs). Note the significantly larger receptive field of the VGG-19 for deeper blocks that simplify detecting global novelties, while the EfficientNet-B4, with its smaller receptive field and larger non-overlapping strides, can focus on locally concentrated novelty.

3.4.1 Local mean-shift region A

Generally, we differentiate between two extreme cases for novelty detection in this work: (1) Global novelty and (2) local novelty. To cover both cases, we found that it is important to first adjust the patch size R to match the desired anomaly fraction of the given input resolution, such that the anomaly *falls* into the receptive field of the computed feature map. In simple terms, for globally distributed novelties, select a large patch size, and for local anomalies, a small one. As already noticed by others, the EfficientNet-B4 works very well for local novelty detection [146], whereas the VGG-19 is better suited for global case [52]. We argue that the main reason for this is that the EfficientNet-B4 uses almost solely 1×1 convolutions, which retain less blurry local features. To validate this assumption, we computed the receptive field sizes for different selected CNN architectures. Tab. 3.4 shows the results. Note that the receptive field can be computed by varying the input size R until the feature map Φ_L of the desired block L has size 1×1.[2] The stride of the receptive field τ is then the remaining input size H divided by the remaining feature map size $\tilde{H}_L$, given by

$$\tau = \frac{H - R}{\tilde{H}_L - 1}. \tag{3.19}$$

It can be seen that the EfficientNet-B4 has a significantly smaller and non-overlapping receptive field compared to, e.g., the VGG-19. To emphasize this point, Fig. 3.8 highlights the different receptive fields of popular architectures as gradient maps with respect to the input.

Finally, the last remaining hyperparameter is the local averaging region A for computing the local mean-shift statistics. Note, the parameter A is similar to the patch size R as it also impacts the effective receptive field of the entire model and hence the sensitivity for globally distributed and locally concentrated novelty. Our final architecture-independent parameter vector is denoted by

$$\theta = \{\Phi_L, R, \tau, A\}. \tag{3.20}$$

We visualized the local mean-shift for the *Bottle* class in Fig. 3.3 as an example.

3.4.2 Efficient feature computation

The computation of the feature map per extracted image patch is quite expensive in practice. For mitigation, we propose computing the features of all patches simultaneously by computing the

[2]Note, for some architectures, there are analytical formulas available (e.g., [9]).

feature map of the entire image in a single forward pass through the neural network. However, one needs to be careful, as the patch size R and stride τ are now restricted by the internal details of the selected architecture and their corresponding receptive fields as shown in Tab. 3.4. E.g. for block 5 of EfficientNet-B4, a single pixel in the feature map corresponds to 16 pixels in the input space, and the stride is fixed to $\tau = 16$. This also limits the freedom of the local averaging region A to a multiple of 16.

3.4.3 Evaluating local novelty detection

For evaluating the local novelty detection capabilities, we use the MVTec [14] dataset, which comprises 15 different defect detection scenarios. All examples are RGB images with size 224×224. The defects are locally concentrated and consist of scratches, scars, small holes, and other industrial defects. Fig. 3.1 shows representative examples of the dataset. Per class, respectively scenario, the MVTec dataset provides a training dataset consisting solely of non-defective examples and a test dataset that includes both non-defective and defective examples. For training, we use all available non-defective examples from the training dataset. Depending on the scenarios, there are between 50 and 500 examples available. Tab. 3.5 shows the results of the experiment averaged across five folds of cross-validation using varying training and test splits again. Because there are no defective images in the training set of MVTec, we only swapped the non-defective training and test data during cross-validation. In other words, the defective examples of the test dataset were kept constant, and only the non-defective examples of the test and training datasets were varied. In terms of hyperparameters for μshift, we selected $\theta = \{L = 5, R = 64, \tau = 16, A = 80\}$. In order to find the best hyperparameters, we performed a grid-search $L \in [2, 6], R \in [48, 144], A \in [64, 164]$ evaluating the average performance across all classes (cf. Fig. 3.7 and Fig. 3.9). The sensitivity of the effective mean-shift region A on the detection performance can be seen in Fig. 3.9. For the EfficientNet-B4, for instance, $A = 96$ gives an AUC of 98.5, $A = 64$ an AUC of 98.3. We also tested the method with the commonly used WideResnet-50 features and achieved 98.1 AUC on average for the same set of hyperparameters.

Generally, the average AUC depends strongly on the average anomaly sizes: For instance, while the pill class benefits from a small averaging region, the screw class performs significantly better with a larger one.

Note that we could reach the reported 99.0 AUC of PatchCore only in a single fold of cross-validation, but not on average over different folds of cross-validation and therefore report slightly lower average scores than in [148]. The same appears to be the case for CutPaste, and we could only touch the reported 90.9 AUC. However, this is still impressive for a self-supervised scheme that does not rely on pre-training or transfer learning. We also tested the related methods PatchSVDD [191], and PaDiM [41]. PatchSVDD reached on average 92.1 AUC, PaDiM scored 97.9 AUC using the EfficientNet-B4 feature space.

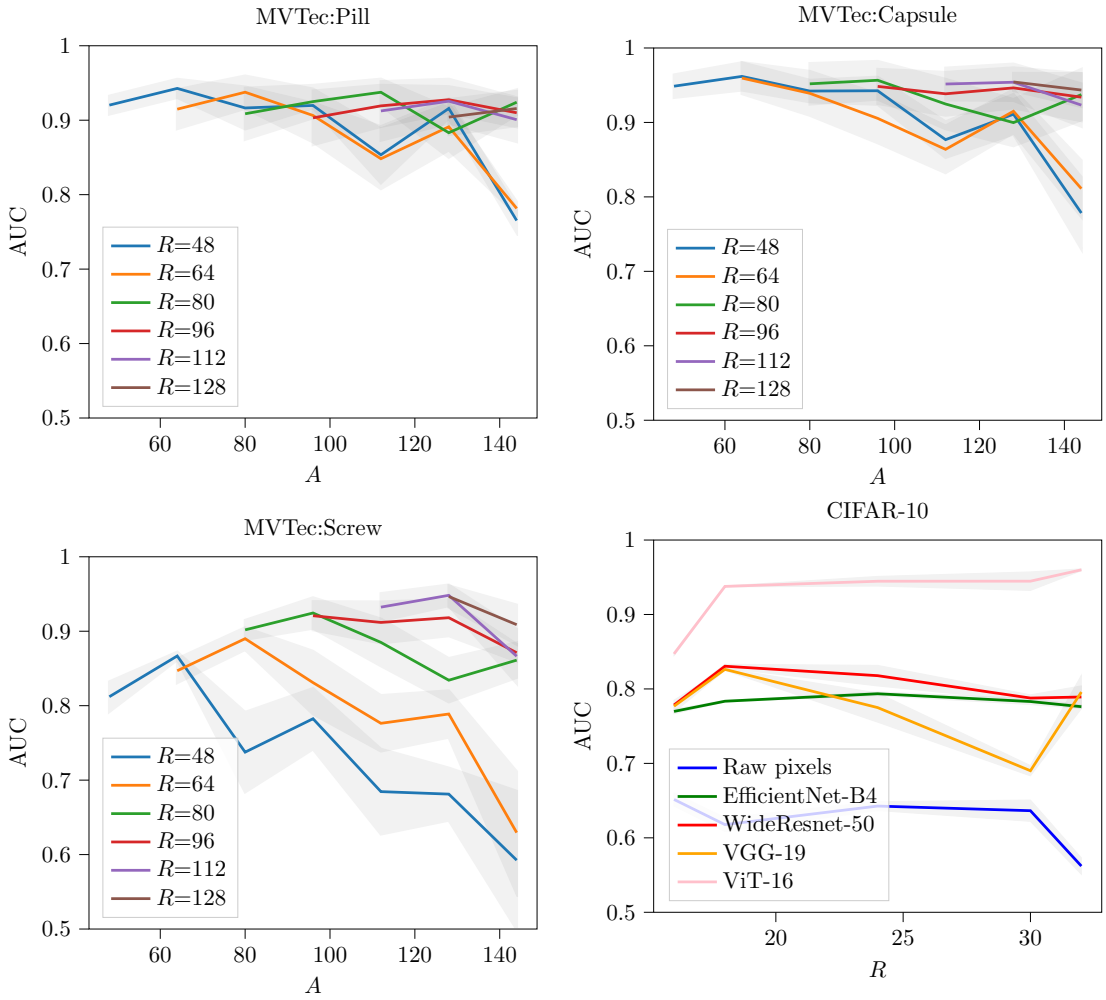


Figure 3.9: Varying the patch size R and the averaging region A impacts the detection rate significantly. Tests were conducted using the critical classes pill, capsule and screw classes from the MVTec dataset, and the entire CIFAR-10 dataset. The MVTec tests used the EfficientNet-B4 architecture. For CIFAR-10 we tested several popular architectures.

	μ**shift**$^{\text{eff}}$	μ**shift**$^{\text{vit}}$	μ**shift**$^{\text{vgg}}$	**PatchCore** [148]	**Mah.Ad** [146]	**CutPaste** [106]	**RotNet** [52]	**Deep-SVM**[160]
bottle	0.999±0.001	0.998±0.002	**1.000±0.000**	**1.000 ±0.000**	0.998±0.001	0.985±0.000	0.790±0.037	0.992±0.004
carpet	**1.000±0.000**	0.999±0.005	0.966±0.008	0.998±0.004	0.933±0.000	0.579±0.000	0.457±0.084	0.973±0.025
leather	**1.000±0.000**	**1.000±0.000**	0.932±0.016	**1.000±0.000**	**1.000±0.000**	0.987±0.000	0.396±0.105	0.961±0.011
pill	0.939±0.017	0.960±0.025	0.822±0.020	**0.983±0.010**	0.902±0.012	0.887±0.020	0.776±0.078	0.500±0.000
tile	0.991±0.000	0.986±0.005	0.961±0.006	**0.998±0.001**	0.987±0.000	0.841±0.020	0.473±0.028	0.495±0.004
wood	0.994±0.004	0.992±0.002	0.982±0.008	0.982 ±0.006	**0.996±0.003**	0.895±0.000	0.675±0.073	0.500±0.000
cable	**0.993±0.000**	0.958±0.005	0.951±0.002	0.961 ±0.008	0.944±0.000	0.833±0.030	0.669±0.037	0.500±0.000
grid	0.994±0.004	0.938±0.002	0.859±0.053	0.909 ±0.013	0.904±0.002	**0.999±0.000**	0.627±0.107	0.596±0.022
toothbr.	**0.999±0.002**	0.995±0.004	0.926±0.033	0.997 ±0.002	0.981±0.000	0.947±0.001	0.702±0.107	**0.999±0.002**
zipper	**0.996±0.004**	0.946±0.003	0.969±0.007	0.995 ±0.003	0.984±0.007	0.995±0.001	0.689±0.105	0.959±0.013
capsule	0.965±0.023	0.950±0.004	0.948±0.025	**0.980 ±0.009**	0.923±0.018	0.802±0.001	0.487±0.091	0.842±0.037
hazeln.	**1.000±0.000**	0.986±0.003	0.982±0.008	**1.000±0.000**	0.992±0.000	0.988±0.000	0.684±0.058	0.512±0.000
metaln.	**0.998±0.003**	0.977±0.000	0.943±0.017	0.990±0.006	0.928±0.005	0.915±0.000	0.712±0.058	0.615±0.015
screw	0.963±0.010	0.839±0.005	0.877±0.032	**0.987 ±0.003**	0.720±0.001	0.892±0.020	0.472±0.062	0.758±0.090
transist.	0.984±0.005	0.928±0.008	0.951±0.015	**0.997 ±0.003**	0.962±0.002	0.844±0.020	0.824±0.022	0.610±0.022
Avg.	**0.987±0.002**	0.964±0.003	0.938±0.004	0.985±0.002	0.961±0.001	0.893±0.01	0.824±0.16	0.721±0.005

Table 3.5: Evaluating the detection of locally concentrated novelty on the MVTec dataset using five-fold cross-validation.

3.5 Complexity, runtime and data efficiency

The complexity and runtime differ heavily between training and test time. For training, the most expensive part is computing the $D \times D$ covariance matrix in Eq. 3.5. With an efficient estimation algorithm, this can be achieved in $O(\min\{(NS)^2D, (NS)D^2\})$. The mean estimation itself is linear $O(NSD)$. At test time, the most expensive computation is computing the T^2 statistics in Eq. 3.4 that needs a $S \times D$-dimensional matrix-vector multiplications $O(SD^2)$.

We noticed that depending on the dimensionality D and the chosen feature space, the computational costs of computing the feature maps quickly exceed the cost of our algorithm. There is a fixed overhead that depends on the size of the input $O(HW)$. E.g., for EfficientNet-B4, in our experiments, the computation of a single MVTec example took 36ms for the feature map and 30ms for the anomaly score. The estimation of the covariance matrix took 20s for a single class. The runtime was measured on standard CPU hardware (Intel i7-6700) without using GPU acceleration. By using a single GPU (GTX 3080Ti) the runtime could be reduced to 1ms for the feature map and 1ms for the anomaly score. Computing the covariance matrix took $3s$.

Therefore, implementing the mean-shift detection by an additional CNN block consisting of a 2D convolution for the Mahalanobis distance and using 2D pooling for the averaging region, the model reached about 500 FPS for the entire pipeline on our hardware. Note that this is much faster than, e.g., the 7 FPS of the PatchCore GPU-model (using default 0.1 sampling ratio) [148]. The increased frame rate is mainly caused by avoiding the k-nearest neighbor search across the entire patch database for every prediction.
As already mentioned, we also evaluated the data efficiency of the models with respect to performance in Fig. 3.10. Again, PatchCore and μshift perform similarly with respect to the number of needed training examples to reach a particular performance level. E.g., 90 % AUC could be achieved with only 10 non-defective examples of the MVTec dataset [14]. In this scenario, we also tested the recently proposed hierarchical method for few-shot anomaly detection (HTDGM) [164], but could not surpass the industrial critical AUC of 90 %. Second, we noticed that the method does not scale well with an increasing number of training examples.

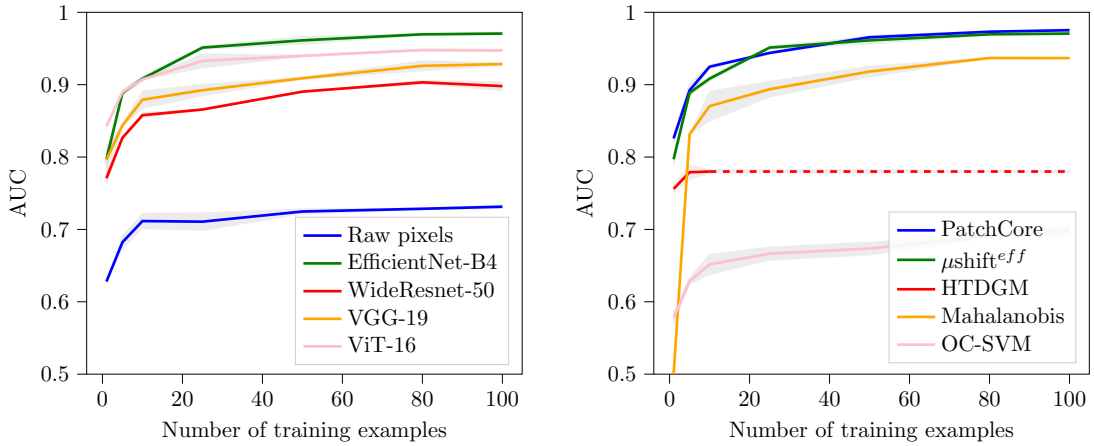


Figure 3.10: Data efficiency on the MVTec dataset across several analyzed architectures. With more than 10 non-defective training examples, an average AUC above 90 % could be achieved using the EfficientNet-B4 or the Vision Transformer Network (ViT) as feature space.

3.6 Conclusion

We found that the success of our approach critically depends on the details of how the patch ensemble is extracted from the input images. The most critical parameters are the number and the size of the patches and by which features the patches are represented. Since mean shift can only be detected when the outlier ensemble is sufficiently separate from the inlier distribution, the overlap acts as a blind spot for novelty detection. In the earlier version of the algorithm based on raw pixels [66], we proposed using a hyperparameter selection rule based on a negentropy approximation [35] to minimize overlapping of the distributions. However, further experiments showed that such an approach does not generalize well towards arbitrary datasets and pre-trained deep features. One reason is that the method prefers larger patch sizes, which is beneficial for global novelty but not for industry-relevant local novelty detection.

Another interesting point appears when comparing our method to PaDiM [41]. As already mentioned, this method is similar to ours when the mean-shift region is equal to the patch size, i.e., $A = R$, and hence the ensemble size S is one. The advantage of our ensemble approach is manifested by the *zigzag* pattern in Fig. 3.9. Here, increasing the local mean-shift area A just slightly over the patch size R significantly increases the detection performance, regardless of the chosen patch size.

For the task of novelty detection, we proposed a method that is capable of detecting novelties effectively using deep mean shifts. By attaching our method on top of a pre-trained neural network, we achieved state-of-the-art performance in standard benchmarks, such as the MVTec defect detection and CIFAR-10 one-class classification challenge. Because of the simple design, the method is easy to implement and provides a fast execution time. By using a GPU, we could reach 500 FPS in our tests. Additionally, because the model only relies on low-order statistics, it is very data efficient and achieves 90 % AUC on the MVTec challenge with only ten non-defective examples.

The main drawback of the method is that the model accuracy heavily depends on the specific problem at hand and the available knowledge about expected anomalies and their sizes. As shown in Tab. 3.1, swapping the feature space can cause a significant change in performance. Second,

not setting the correct patch size reduces the performance quickly. However, the same limitations also appear in other methods, such as RotationNet or PatchCore. For practitioners, it is of great importance to use domain knowledge and set hyperparameters accordingly. A central open question is how to derive those hyperparameters directly from data, which we leave for future work.

Another direction to further improve the model in practice is incorporating data from other sensor types, such as Laser or Lidar. This can be done quickly by providing the data as another channel of the feature map (cf. Eq. 3.1). We will investigate a method for fusing range data with images in Chapter 6.

CHAPTER 4

Negentropy estimation using independent component analysis

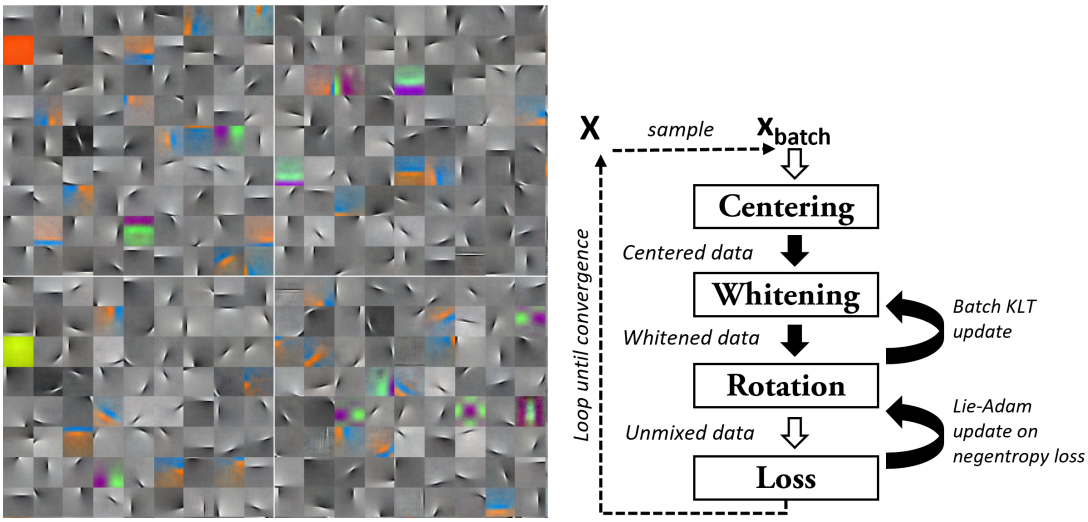


Figure 4.1: Examples of the first 484 independent components estimated from the ImageNet dataset ($1.2 \cdot 10^6$ examples) (left). Every tile represents a single column of the mixing matrix, which is reshaped to $3 \times 200 \times 200$ for illustration purposes. In Lie-ADAM, we used a learning rate of 0.01 and a batch size of 484. The model was trained with three runs through the dataset, which took $3h$ on standard hardware with a single GPU. A schematic overview of the algorithm is shown on the right.

The hyperparameter selection method based on typical set theory in the previous chapter (cf. Sec. 3.3.4) for the optimal patch size included the computation of the negentropy measure for large data matrices $\mathbf{X}$. This computation is not trivial for large datasets, as the data often does not fit into memory. For such large-scale negentropy estimation problems, we developed Lie-Adam ICA for computing independent component analysis (ICA). The following chapter is based on the Lie-Adam publications [65, 68].

4.1 Introduction

Independent component analysis (ICA) is a statistical signal processing technique for identifying statistically independent linear components [35]. Its main applications are blind source separation and feature extraction. Compared to principal component analysis, which only considers second-order statistics, ICA also incorporates higher-order moments such as skewness and kurtosis. The unique property of linear ICA is identifiability, which guarantees that the underlying components can be identified up to scale and permutation if, at most, a single source is Gaussian [80]. The use of independent components can be separated into two applications. First is, identifying independent features within a dataset and therefore estimating a specifically tailored feature set that acts as a new basis. This basis may be reduced in dimension and therefore applied as a dimension-reduction technique. Compared to principal component analysis (PCA) or factor analysis (FA), the underlying components do not need to be Gaussian-distributed. This is important, as that difference allows us to identify non-Gaussian components. A second application leverages that property and uses ICA as a tool for separating mixed sources. This task is called blind source separation (BSS) and is also known as the cocktail party problem. ICA guarantees identifiability if, at most, a single source is Gaussian and is able to estimate the components up to scale and permutation. Performing ICA in large-scale scenarios is of particular interest as more and more large high-resolution datasets become available such as, e.g., ImageNet, Lidar or RGBD-Video [165]. The problem of ICA is given by the matrix decomposition

$$\mathbf{X} = \mathbf{S}\,\mathbf{A}, \tag{4.1}$$

where $\mathbf{X}$ is the centered $n \times d$ data matrix, $\mathbf{S}$ the $n \times k$ source matrix, and $\mathbf{A}$ the $k \times d$ mixing matrix. The mixing matrix $\mathbf{A}$ and its pseudo-inverse, the unmixing matrix $\mathbf{A}^{-1}$, can further be decomposed into $\mathbf{A}^{-1} = \mathbf{W}\,\mathbf{R}$, where $\mathbf{R}$ is a $k \times k$ orthogonal matrix and $\mathbf{W}$ is the $d \times k$ whitening matrix. As every whitened signal stays whitened under orthogonal transformation, the problem can be split into three steps: (1) centering the data, (2) removing second-order correlations (whitening), and (3) searching for an orthogonal matrix that identifies the independent components by minimizing mutual information $\mathcal{I}$ between the components $s_1, \ldots, s_k$. The whole optimization procedure is given by

$$\arg\min_{\mathbf{R},\mathbf{W}} \mathcal{I}(s_1, \ldots, s_k) \tag{4.2}$$

with

$$\mathbf{S} = \mathbf{X}\,\mathbf{W}\,\mathbf{R} \tag{4.3}$$

subject to

$$\mathbb{E}[\mathbf{X}] = \mathbf{0} \tag{4.4}$$

and

$$\mathbf{R}^T\,\mathbf{R} = \mathbf{I} \tag{4.5}$$

and

$$\overline{\mathbf{W}}\,\mathbf{D}^2\,\overline{\mathbf{W}}^T = \mathbf{\Sigma}, \tag{4.6}$$

where $\mathbf{I}$ is the $k \times k$ identity matrix and $\mathbf{\Sigma}$ is the $d \times d$ covariance matrix of the data $\mathbf{X}$. Typically the whitening procedure is applied beforehand, and the optimization of the independent components is done separately. In signal processing, this separation corresponds to estimating the whitening matrix $\mathbf{W} = \mathbf{D}^{-1}\,\overline{\mathbf{W}}$ and estimating a rotation matrix $\mathbf{R} \in \mathrm{SO}(k)$. $\mathbf{D}$ is a diagonal matrix with the estimated standard deviations $\sigma_1, \ldots, \sigma_k$ of the data along the k principal directions given by the columns of $\overline{\mathbf{W}} \in \mathbf{V}_k(n)$. $\mathbf{V}_k(n)$ is called the Stiefel manifold and consists of all k-dimensional orthonormal bases in $\mathbb{R}^n$. We combine the whole procedure into a single neural network architecture that can be trained end-to-end by optimizing an appropriate contrast or loss function with

back-propagation [153]. Our proposed neural network stacks three layers: (1) A moving average layer, (2) a whitening layer, and (3) an orthogonal layer.
In this chapter, we are targeting the case where the data $\mathbf{X}$ does not fit into memory, and stochastic optimization for an efficient GPU-based neural network implementation is required. Here, stochastic optimization means that the data $\mathbf{X}$ is processed in mini-batches during gradient descent instead of computing the gradient over the entire dataset. Hence, just a stochastic estimate of the loss function is available. This use case is of special interest during data exploration when the number of extracted components is large, and the redundancy in the data is unknown. Current ICA algorithms based on Lie-Group techniques use L-BFGS (Broyden–Fletcher–Goldfarb–Shanno optimization) [109] and rely on a costly full-batch line search, which makes them very accurate but slow in practice [162]. On the other hand, algorithms based on Infomax [13], also referred to as maximum likelihood estimation of the ICA model, have good convergence rates [2] but are known to be hard to optimize in online scenarios [121]. Montoya-Martínez et al. [121] identified difficulties during optimization, such as reaching non-optimal plateaus of the loss function or problem-specific tuning of step and batch size parameters. Further, noisy gradient estimations complicate second-order optimization schemes like L-BFGS. Both aspects prevent current ICA algorithms from computing ICA on large high-dimensional datasets, so there is a need for a better optimization scheme for the entire ICA pipeline.

Contributions. Our contributions to that pipeline (cf. Fig. 4.1) are: (1) showing the importance of orthogonality constraints in large-scale ICA and the pre-whitening requirement for stability, (2) improving the geodesic flow update rule by using the ADAM optimizer in combination with the Caley approximation for the matrix exponential, and (3) demonstrating the resulting scaling capabilities by computing the first 484 independent components of the ImageNet challenge. By using b-sized mini-batches, the space complexity of the entire pipeline for d-dimensional inputs and k components is limited to $\mathrm{O}(d(k+b))$.

4.2 Related work

Estimating independent components (ICs) has been tackled in several different ways [35]. Roughly there exist three main approaches: (1) maximizing higher-order cumulants (e.g., in JADE [27]). (2) maximizing mutual information $\mathcal{I}(\mathbf{X}, \mathbf{S})$ between input and output, which is called Infomax [13], and (3) Maximizing non-Gaussianity of $\mathbf{S}$ by maximizing negentropy [78], Further, there are differences in data handling: Offline approaches require access to the entire dataset, online approaches process items one by one, and stochastic approaches process small subsets of data (mini-batches). Hyvärinen [75] made a complete overview of the different techniques. All approaches need to ensure that $\mathbf{S}$ is approximately decorrelated and s has a unit variance. These constraints are either guaranteed by first whitening the data and enforcing $\mathbf{R}$ is orthogonal or by maximizing output entropy as in Infomax [25]. The latter is the same as a maximum-likelihood estimation of a chosen source model $p(s_i)$. Orthogonality constraints yield more accurate results but are slower to optimize [2]. Enforcement either is done by orthogonalization between each update (e.g., FastICA [78]) or implicitly using Lie group techniques with the corresponding matrix exponential [139]. Traditionally the fixed-point algorithm FastICA is faster, but Selvan et al. [162] showed that algorithms relying on Lie group techniques are more accurate. However, the recently proposed Picard-O algorithm overcomes some of the performance shortcomings by using an optimized Quasi-Newton procedure based on L-BFGS [2] and a polynomial approximation of the Riemannian Hessian. In the non-orthogonal Infomax setting, there are also improvements by using ADAM instead of L-BFGS as shown by Scarpiniti et al. [158]. Their experiments lack the comparison with L-BFGS, and only four- and five-dimensional problems were tested. Majorization-minimization ICA (MM)

proposed by Ablin et al. [3] is the most recent algorithm for estimating ICA online. The update rule guarantees convergence through an expectation-maximization (EM) scheme.

The interpretation of ICA as a neural network has a long history, too. Both Neural PCA [128] and Neural ICA [86] have been investigated over the years. Plumbley [138] proposed a way to express the backpropagated gradient with respect to the general linear group $GL(\mathbb{R}^n)$ of invertible matrices. This offers two directions. First, using standard backpropagation and therefore making use of recent developments in the field of deep learning [90]. Second, separating concerns of propagating gradients and enforcing orthogonalization constraints by explicitly computing the matrix exponential.

Kessy et al. [87] gives an overview of different offline whitening strategies and their particular properties regarding correlation and compression. Unfortunately, they left out ICA as a whitening algorithm with unique independence properties and a solution somewhere between zero-phase component analysis (ZCA) and PCA. For incremental processing, Arora et al. [10] made a comprehensive overview of different algorithms for both incremental and stochastic computation. Unfortunately, their evaluation lacks Batch KLT (Karhunen-Löwe-Transformation) proposed by Levy and Lindenbaum [105] and later extended to the case of uncentered data by Ross et al. [147]. In principle, on the side of neural PCA, the well-known Generalized Hebbian Algorithm (GHA) [155] is a proper choice. However, as we see later in experiments, a batch version of the Karhunen-Löwe-Transformation converges faster.

4.3 Large-scale ICA

Again, independent component analysis (ICA) computes the matrix decomposition

$$\mathbf{X} = \mathbf{S}\,\mathbf{A}, \tag{4.7}$$

with mixing matrix $\mathbf{A}$ and independent sources arranged as columns of $\mathbf{S}$. The following section describes the relevant algorithmic decisions for conducting ICA in large-scale scenarios on CPU and GPU hardware using as few hyperparameters as possible.

4.3.1 Optimizing independence

ICA is typically done in two steps: first, the data are whitened, and second, an orthogonal transformation $\mathbf{R}$ is chosen to maximize the independence between the components in $\mathbf{S}$ [35]. The independence between several random variables $s_1, \ldots, s_k$ is generally measured by the mutual information $\mathcal{I}(s_1, \ldots, s_k)$. A property of orthogonal transformations is that they do not change the shape and, consequently, not the differential entropy of a distribution. Hence, instead of the multivariate mutual information, we can minimize the differential entropy of the individual decorrelated components [62, 63]. The negentropy approximation $J(s_i)$ [35] is based on the maximum entropy principle and measures non-Gaussianity [162, 80]. The mutual information is then approximated by

$$\mathcal{I}(s_{1:k}) \propto -\sum_i J(s_i) = -\frac{1}{k}\sum_i (\mathbb{E}[\mathrm{G}(s_i)] - \mathbb{E}[\mathrm{G}(z)])^2 \tag{4.8}$$

It measures the difference between the expectation of the whitened component s_i and a Gaussian variable $z \sim \mathcal{N}(0,1)$ under the specified source model $\mathrm{G}(s_i)$. For any chosen non-quadratic G, the independence of the components is maximized by optimizing for $p(s_i)$ [74], which is a task-specific chosen probability density function (pdf) for the components s_i.

In the ICA literature, three types of sources are considered: Gaussian, Sub-Gaussian, and Super-Gaussian components [80]. These components can be distinguished statistically by kurtosis: Gaussian sources have zero, sub-Gaussian have negative and super-Gaussian sources have positive kurtosis.

Here, we chose the traditional generalized kurtosis measure [162] for $\mathrm{G}(\cdot)$, which is given by

$$G_1(s_i) = \frac{1}{a_1} \log \cosh(a_1 s_i), \text{ and} \tag{4.9}$$

$$G_2(s_i) = -\frac{1}{a_2} \exp(-a_2 \frac{s_i^2}{2}). \tag{4.10}$$

Like Selvan et al. [162] we set $a_1 = 1.2$ and $a_2 = 0.99$. Note that $p(\mathbf{s})$ in the ICA model does not necessarily need to be normalized and hence is denoted by $G(\mathbf{s})$.

Using negentropy approximations solves a central problem in ICA [74]. For any chosen non-quadratic G, the independence of the components is maximized by optimizing for $G(s_i)$. As $\mathbb{E}[p(z)]$ is a constant, this is achieved by either maximizing or minimizing $\mathbb{E}[G(s_i)]$. The desired direction depends on the statistics of the components s_i and the underlying source model $p(s_i)$. In practice, we often distinguish between sub-Gaussian and super-Gaussian component models [80, 2]. As shown by Hyvärinen et al. [80], the components in image processing are typically super-Gaussian, which is strongly related to sparsity. Sometimes it is unknown whether the underlying sources are sub- or super-Gaussian or a mixture of both. Hence, automatically switching between maximizing or minimizing $\mathbb{E}[G(s_i)]$ is needed [104]. Extended Infomax [104] and Picard-O [2], for instance, achieve this by computing the component-wise kurtosis K_i based on a generic stability analysis due to Cardoso and Laheld [26] and changing the direction of the gradient with $\mathrm{sign}(K_i)$. In this sense, negentropy approximations are very similar and indeed a generalized kurtosis measure [74], which is identical for $G(s) = s^4$, assuming the variables s_i are centered and scaled to have unit variance. For whitened data, the multivariate kurtosis can be computed by averaging kurtosis component-wise. Note that kurtosis mainly measures outliers and hence leads to wrong results if the dataset $\mathbf{X}$ is polluted by outliers [188]. See Hyvärinen et al. [80] for further details.

Another important point appears when comparing the optimization procedure to log-likelihood optimization. Here, the log-likelihood of the model [104, 2, 25] is given by

$$\mathcal{L}(\mathbf{X}, \mathbf{A}) = \log \, |\det \, \mathbf{A^{-1}}| + \log \sum_{ji} \mathrm{sign}(K_i) p(s_{ji}). \tag{4.11}$$

This is a much easier task than optimizing the negentropy directly, which has at least two stationary points due to its quadratic form. To illustrate this, we unmixed a toy mixture consisting of two Laplacian variables $\mathrm{Laplace}(0,1)$ and a uniform variable $\mathcal{U}(-2,2)$. Fig. 4.2 shows the comparison of the two strategies. It is easy to see that minimizing negentropy needs to overcome a local minimum yielding a significant slowdown compared to optimizing $\mathrm{sign}(K_i) G(s_i)$ directly.

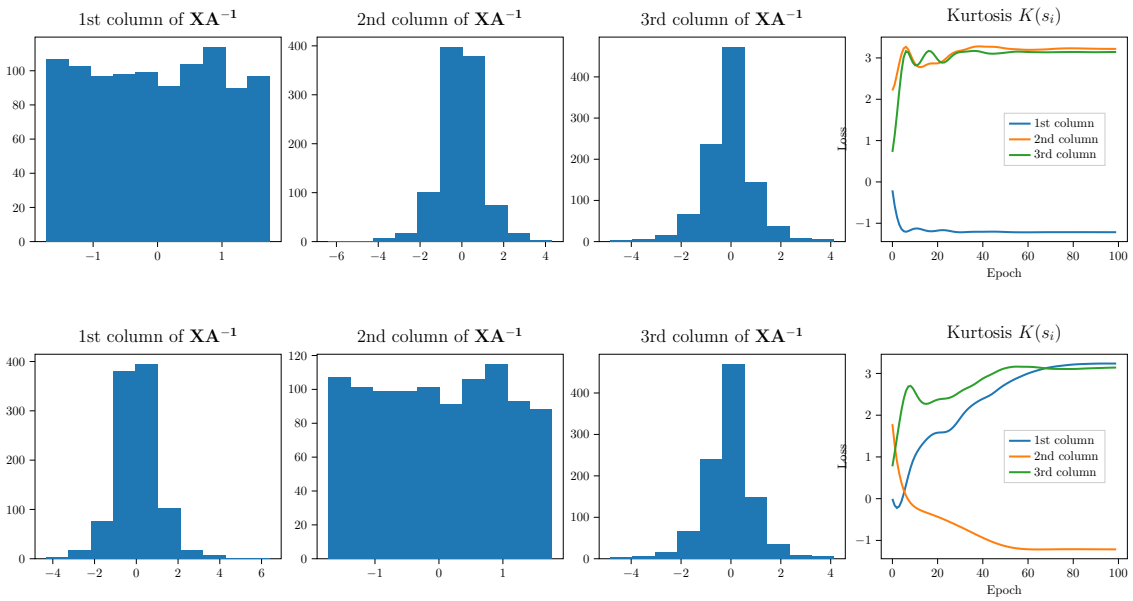


Figure 4.2: Toy example demonstrating the unmixing of (top) a three-dimensional mixture by minimizing $\text{sign}(K_i)G(s_i)$ and (bottom) by minimizing negentropy. The plot shows the measured kurtosis per trained epoch.

In large-scale ICA, the kurtosis signs are broadly unknown, and an automatic switching mechanism is needed [26]. Where needed, we also integrate this technique.

4.3.2 Parallelism and GPU-hardware

We must reduce non-parallel computing steps in the algorithm to use highly parallel GPU hardware. In existing algorithms, there are two critical steps involved that complicate parallelism. The first is line search which is part of L-BFGS, for instance, as this includes a single-threaded loop iterating over the gradient direction. Second, greedy approaches, as in Incremental PCA [20], and Robust ICA [195] are inherently recurrent and, therefore, suboptimal for parallelism. This is because, after extracting a single component, the orthogonality constraint must be enforced to prevent the algorithm from extracting the same components multiple times. We believe in large-scale ICA, and we need to ensure that we extract many components in parallel, but we will evaluate the incremental approach as well.

4.3.3 High input dimensionality

High input dimensionality impacts the ICA algorithm heavily. Standard algorithms compute a square mixing matrix, which in the case of ImageNet would require storing 150528^2 parameters requiring 90 Gb in single precision for $224 \times 224 \times 3$ RGB image input. This is especially a limiting factor in GPU computing, where the maximum available memory is comparatively small. Such large parameter matrices also considerably impact the stability of the algorithms. For mitigation, data dimension is typically reduced beforehand by applying PCA, which is also needed for whitening [80]. Dimension reduction allows for non-square unmixing, in which fewer independent components than the number of input dimensions are unmixed. There are several algorithms for performing PCA in online scenarios available [10].

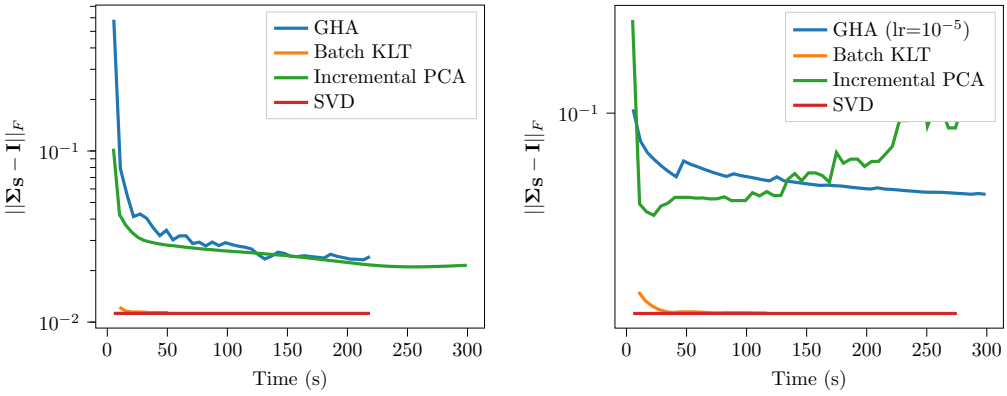


Figure 4.3: Convergence of the 100×100 (left) and 1000×1000 (right) covariance matrix $\mathbf{\Sigma_S}$ pertaining to the largest eigenvalues to the identity matrix as measured by the Frobenius norm $||.||_F$ on the CIFAR-10 dataset (50.000 examples of size $3 \times 32 \times 32$). As baseline, we show Singular Value Decomposition (SVD) which runs offline. Batch size was set to 100 and 1000, respectively.

As whitening is a mandatory step in our ICA procedure, we compared the convergence of these algorithms on the well-known CIFAR-10 dataset. Here, besides Singular value decomposition (SVD), we focus on algorithms that also work with mini-batches:

1. Singular value decomposition (SVD) is an offline technique for computing the whitening matrix and represents a principled baseline method for this task.

2. Incremental PCA [20] is also known as incremental SVD and was introduced It is an online algorithm for PCA that computes each sample one by one. Theoretically, it guarantees whitening within a single run through the data.

3. Batch KLT [147] is a mini-batch variant of sequential SVD [105] that corrects a bias in case of non-centered data. Batch KLT guarantees whitening within a single run through the data, too.

4. Generalized Hebbian Algorithm (GHA) [155] is based on Oja's rule [128] and uses a special update scheme ensuring orthogonality.

Batch KLT combines several advantages, such as fast convergence, high accuracy, constant memory usage, and the absence of tuning parameters. The main drawback is that it needs to compute a smaller SVD for every mini-batch in every update step. Hence, it restricts the mini-batch size to at least the number of extracted components k and means large mini-batches may not fit into memory. GHA converges to the *true* principal components when a correct learning rate policy is given. The learning rate in the GHA is a critical parameter, and we found that a value of 10^{-5} works well. We logged how *white* the validation data is by computing the Frobenius norm $|| \cdot ||_F$ between the covariance matrix $\mathbf{\Sigma_S}$ and the identity matrix $\mathbf{I}$. As required, we set the batch size to the number of extracted components, i.e., either 100 or 1000. Again, this comes at the price of setting the batch size to at least the number of components. Fig. 4.3 shows the convergence over time. Clearly, Batch KLT is superior for stochastic whitening in this particular scenario. See [107] and [32] for a great overview of the runtime and space complexities of the algorithms. We use Batch KLT in all further experiments and our default in Large-scale ICA. The implementation of Batch KLT as neural network layers for computing $\mathbf{W}$ is shown in Alg. 1.

Algorithm 1 Batch KLT [147] based whitening layer with *forward*() and *backward*() methods.

$\hat{\mathbf{W}}_\mathbf{0} \leftarrow \mathbf{I}$
$\mathbf{D}_0 \leftarrow \mathbf{I}$
$\bar{\mathbf{B}}_0 \leftarrow \mathbf{0}$
$n \leftarrow$ Number of examples
procedure FORWARD($\mathbf{B}_i$, $\bar{\mathbf{W}}$, $\mathbf{D}$, $\bar{\mathbf{B}}_i$)
    $\hat{\mathbf{B}}_i \leftarrow \mathbf{B}_i - \bar{\mathbf{B}}_i$
    $\hat{\mathbf{S}} \leftarrow \mathbf{B}_i \bar{\mathbf{W}} \mathbf{D}^{-\mathbf{1}}$
    return $\hat{\mathbf{S}}$
procedure BACKWARD($\mathbf{B}_i$, $\hat{\mathbf{W}}_{i-1}$, $\mathbf{S}_{i-1}$, $\bar{\mathbf{B}}_{i-1}$)
    $\bar{\mathbf{B}}_i, \mathrm{n}_{seen} \leftarrow \mathrm{incrMean}(\mathbf{B}_i, \bar{\mathbf{B}}_{i-1})$
    if $\mathrm{n}_{seen} < n$ **then**
        $\hat{\mathbf{B}}_i \leftarrow \mathbf{B}_i - \mathbb{E}[\mathbf{B}_i]$
        if $i > 0$ **then**
            $\mathbf{q} \leftarrow \sqrt{\frac{n \cdot n_{seen}}{n \cdot (\bar{\mathbf{B}}_i - \mathbb{E}[\mathbf{B}_i])}}$
            $\hat{\mathbf{B}}_i \leftarrow [\mathbf{S}_{i-1} \hat{\mathbf{W}}_{i-1}, \hat{\mathbf{B}}_i, \mathbf{q}]$
        $\mathbf{U}, \mathbf{S}_i, \bar{\mathbf{W}}_i \leftarrow \mathrm{svd}(\hat{\mathbf{B}}_i)$
        $\mathbf{D}_i \leftarrow \sqrt{\mathbf{S}_i^2 / (\mathrm{n}_{seen} - 1)}$
        return $\bar{\mathbf{B}}_i, \bar{\mathbf{W}}_i, \mathbf{D}_i, \mathbf{S}_i$

Besides pre-whitening, ICA can be performed incrementally by optimizing higher-order cumulants, where just a single component is computed at each iteration, e.g., Robust ICA [195]. Consequently, all frequency components can be extracted simultaneously, and no dimension reduction needs to be conducted.

4.3.4 High output dimensionality

When the input dimensionality is large, the output dimensionality of the algorithm can be large, too. This impacts the algorithm's stability as the gradient update is done in very high-dimensional parameter space. We are targeting scenarios where the number of ICs k is between 500 and 1000 components, as this is a typical layer width in deep learning. Algorithms based on Infomax perform gradient updates in the entire parameter space of square invertible matrices, the general linear group $GL(k)$, by maximizing the output entropy [25]. Optimization in higher dimensions causes problems as this space is not compact and includes diverging series [49]. For improving stability, orthogonality constraints are integrated [2, 78], and the search space is limited to the special orthogonal group $SO(k)$ [138]. This group has three appealing properties for our scenario: (1) the number of parameters is reduced to $k(k-1)/2$, (2) the special orthogonal group is compact, and hence there are no diverging series, and (3) $SO(k)$ is a Lie group with a tangent space, called the Lie algebra, that can be used for gradient updates. The so-called *geodesic flow* [127, 138] makes use of this and computes ICA with orthogonality constraints using Lie group techniques. An improved method integrates L-BFGS for acceleration [162]. Traditionally the fixed-point algorithm FastICA [78] is faster, but Selvan et al. [162] also showed that algorithms relying on Lie group techniques are more accurate. The recently proposed Picard-O algorithm overcomes some of the performance shortcomings by using an optimized Quasi-Newton procedure based on L-BFGS [2]

and a polynomial approximation of the Riemannian Hessian. We follow the direction of Plumbley [139] and implicitly impose the orthogonality constraints by using Lie group techniques.

The observation that all ICA solutions form a $k \times k$ orthogonal matrix, such that $\mathbf{R}^T \mathbf{R} = \mathbf{I}$, is the starting point for applying Lie group techniques. That is because the set of orthogonal matrices forms a Lie group [12]. When restricting to matrices with determinant one, this set is called the special orthogonal group $\mathrm{SO}(k)$. The corresponding Lie algebra is called $\mathfrak{so}(k)$. It consists of all skew-symmetric matrices and becomes the space for gradient descent. Every skew-symmetric matrix $\mathbf{\Theta}$ can be uniquely parameterized by a vector $\mathbf{r}$ of dimension $k(k-1)/2$ giving rise to a vector space. The components of that vector are called Plücker coordinates. Further, every skew-symmetric matrix $\mathbf{\Theta}$ can be related to an orthogonal matrix $\mathbf{R}$ by

$$\mathbf{R} = \exp(\mathbf{\Theta}), \tag{4.12}$$

where $\exp(\cdot)$ is the matrix exponential. For instance, the vector $\mathbf{r} = (a, b, c)$ describes the skew-symmetric matrix

$$\mathbf{\Theta}_\mathbf{r} = \begin{pmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{pmatrix}. \tag{4.13}$$

Further, as the group $\mathrm{SO}(k)$ is closed under multiplication, i.e., multiplying two orthogonal matrices gives a third orthogonal matrix. This property leads to the *geodesic flow* update rule suggested by Nishimori [127] and Fiori [48]: The gradient of the loss function $\nabla_\mathbf{r}\mathcal{I}$ represents an infinitesimal rotation and hence an element of $\mathfrak{so}(k)$. However, in order to compute a valid gradient step beyond the neighborhood of $\mathbf{r}$, the gradient direction needs to be expressed by the Lie bracket. We refer to Plumbley [138] for the full derivation of the relation between the two gradient expressions using the commutator

$$\nabla_{\mathbf{\Theta}_\mathbf{r}}\mathcal{I} = (\nabla_\mathbf{r}\mathcal{I})^T \mathbf{r} - \mathbf{R}^T (\nabla_\mathbf{r}\mathcal{I}). \tag{4.14}$$

From there, we compute the corresponding parameter vector $\mathbf{r}$ by taking the upper triangular matrix of $\nabla_{\mathbf{\Theta}_\mathbf{r}}\mathcal{I}$ corresponding to the Plücker coordinates. The geodesic flow update rule is then given by

$$\mathbf{R}_{i+1}^T = \exp(-\eta\, \mathbf{\Theta}_{\mathbf{r}_i})\, \mathbf{R}_i^T, \tag{4.15}$$

with step size η and the skew-symmetric matrix $\mathbf{\Theta}_{\mathbf{r}_i}$ parametrized by $\mathbf{r}_i$ at the i-th iteration step. Unfortunately, rotation matrices for $k > 2$ are not commutative. Hence we cannot make additive steps of descent in $\mathfrak{so}(k)$ and need to map between the Lie algebra and the manifold in every iteration. This procedure is not for free, as we need to evaluate the matrix exponential between every iteration. Additionally, in order to update the solution at the current iterate $\mathbf{r}_i$, we need to translate the geodesic gradient direction by matrix multiplication. In the case of the special orthogonal group $\mathrm{SO}(k)$, translation on the manifold is given by matrix multiplication [48]. As remarked by Plumbley [139], even if it is not possible to take several consecutive steps, one can take longer steps along the one-dimensional gradient direction in $\mathfrak{so}(k)$. These step sizes may be computed with Quasi-Newton schemes such as L-BFGS [162].

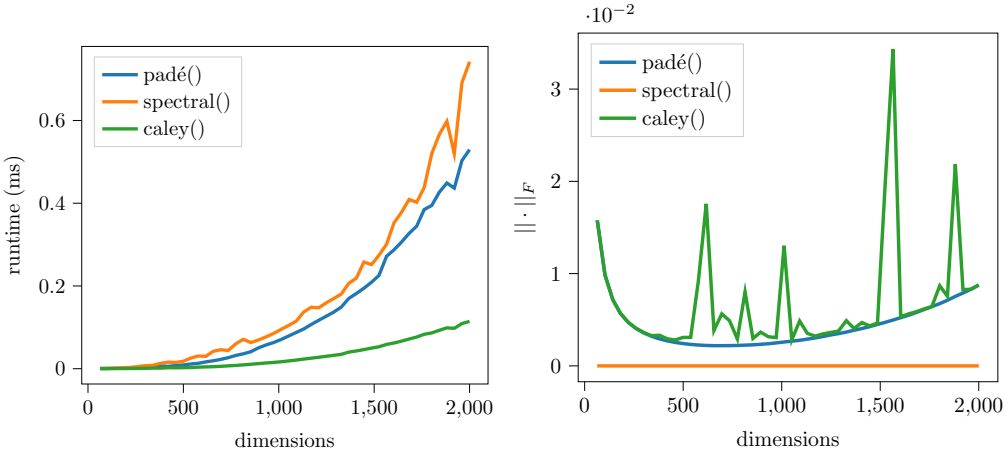


Figure 4.4: Runtime and precision of the matrix exponential methods comparison between spectral $\exp(\mathbf{M})$, $\text{cayley}(\mathbf{M})$, $\text{padé}(\mathbf{M})$ for $\mathbf{M} \sim \mathcal{N}(0, 0.1)$ using PyTorch. Interestingly both the Cayley approximation $\text{cayley}(\mathbf{M}) \approx (\mathbf{I} - \frac{\mathbf{M}}{2})^{-1}(\mathbf{I} - \frac{\mathbf{M}}{2})$ and the Padé algorithm give similar accuracies (10^{-4}) up to 500 dimensions. However, for larger dimensions, the Caley approximation shows large peaks.

The accuracy and speed of the method depend critically on the computation of the matrix exponential and the overall number of iterations. In principle, the exponential can be computed by spectral decomposition and computing the exponential of the eigenvalues. Instead, the Padé-algorithm [6] can be used, which gives a significant speedup over spectral decomposition. Furthermore, we also tested the Cayley approximation, which is considerably faster and almost as accurate for most practical problems with $k < 1000$, and therefore our choice for being included in the algorithm (cf. Fig. 4.4).

4.3.5 Large datasets

Computing ICA on large datasets is particularly difficult as the entire dataset, sometimes over millions of input examples, does not fit into memory. As the loss function is computed as a sum over all input examples, performing parameter updates by gradient descent is very costly. As a consequence, we cannot use offline or full-batch algorithms and need to switch to online or mini-batch processing. Mini-batch processing induces a source of stochasticity impacting the gradient estimation and its variance. This is problematic for methods that rely on computing optimal step sizes per iteration by line search as they become strongly sub-optimal. Momentum-based methods estimate curvature information by averaging over past mini-batches, which is more robust and allows a significant speedup [158]. Such methods arose from deep learning, where both the number of training examples and the number of parameters are large. Scarpiniti et al. [158] already integrated the well-known ADAM algorithm into Infomax and improved its speed for low-dimensional problems ($d \leq 5$). This kind of optimization is especially attractive in non-convex problems or when only a stochastic estimate of the gradient is available, and hence line-search techniques are unreliable. However, in our experiments, this approach did not converge for high-dimensional data. We take this idea for ICA and also use the ADAM algorithm [90] for accelerating stochastic gradient descent. In machine learning, especially in deep learning, ADAM optimization is used when adaptive learning rates are desired. Hence, instead of using the L-BFGS algorithm for optimization [162, 2], we choose ADAM as an accelerated gradient descent technique. Similar to L-BFGS, ADAM implements a Quasi-Newton scheme. Instead of approximating the

Hessian matrix itself, first-order gradient information is used to estimate the gradient scale online from exponential decaying past gradients. One interpretation of this is that it can be seen as a windowed estimate of the diagonal of the Hessian matrix of the loss function [40]. As we are targeting end-to-end optimization without pre-whitened data, a second advantage appears. During subspace identification, the ICA gradients are extremely noisy. Experiments showed that L-BFGS optimization is unreliable in such cases and numerically unstable.

As mentioned, we are optimizing in the Lie algebra of skew-symmetric matrices, which are parametrized by the Plücker coordinate vector $\mathbf{r}$. The main advantage of its vector space structure is the fact that we can optimize the components of $\mathbf{r}$ independently of each other. We leverage this property and use ADAM [90] to control the learning rate per vector component. Hence, the update rule becomes

$$\mathbf{R}_{i+1}^T = \exp(-\boldsymbol{\Theta}_{\hat{\mathbf{r}}_i})\,\mathbf{R}_i^T, \tag{4.16}$$

where $\hat{\mathbf{r}}_i = \mathbf{H}_{\text{adam}}\,\mathbf{r}_i$ is the new coordinate vector scaled by the diagonal matrix $\mathbf{H}_{\text{adam}}$ estimated by ADAM. The diagonal terms represent estimated curvature information. We refer to Kingma and Ba [90] for a detailed description of ADAM. The full model is trained by minimizing statistical independence through back-propagation, which makes swapping contrast functions straightforward.

The final optimization objective problem is

$$\arg\min_{\mathbf{R}} \mathcal{I}(\mathbf{X}) = \sum_{j}^{N} \sum_{i}^{k} \text{sign}(K_i)\text{G}(s_{ji}), \tag{4.17}$$

$$\text{where } \mathbf{S} = (\mathbf{X}\,\mathbf{W}\,\mathbf{R}), \tag{4.18}$$

where $\mathbf{X}$ is the centered data matrix and $\mathbf{W}$ is the whitening matrix estimated by Batch KLT. In scenarios without access to the whole dataset, column-wise centered input data is sometimes impossible. To account for this, we compute the column-wise mean $\boldsymbol{\mu}$ incrementally for every example $\mathbf{x}_i$ during the first epoch by using the well-known formula

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + \frac{\mathbf{x}_i - \boldsymbol{\mu}_{i-1}}{i} \tag{4.19}$$

and keep it fixed afterward. In the following, we refer to our approach as Lie-Adam. The implementation as a neural network layer is given in Alg. 2.

To illustrate the optimization of the loss, we plotted the loss during optimization. Fig. 4.5 shows the components of the gradient $\nabla_{\mathbf{R}}\mathcal{L}$ during optimization of the toy unmixing task with two Laplace and two uniform components (cf. Fig. 4.2). It clearly shows the smooth structure of the optimization manifold with oscillating gradient values which get exploited by the momentum of ADAM. The oscillation lets the model easily converge to a better minimum. This particular model is trained offline.

Algorithm 2 Lie group based ICA layer with *forward*(), *loss*() and *backward*() methods.

$\hat{\mathbf{R}}_\mathbf{0} \leftarrow \mathbf{I}$
$G \leftarrow -\frac{1}{a}\exp(-a\frac{s_i^2}{2})$
expm $\leftarrow$ cayley
$\eta \leftarrow 10^{-2}$

procedure FORWARD($\mathbf{B}_i$, $\mathbf{R}$)
    return $\mathbf{B}_i\mathbf{R}$

procedure LOSS($\mathbf{S}$)
    $K = \text{Kurt}[\mathbf{S}]$
    return $\sum_j^N \sum_i^k \text{sign}(K_i) G(s_{ij})$,

procedure BACKWARD($\mathbf{R}$, $\nabla_\mathbf{R}$)
    $\nabla_{\mathbf{\Theta}_\mathbf{r}} L = (\nabla_{\mathbf{r}_i} L)^T \mathbf{r}_i - \mathbf{R}_i^T (\nabla_{\mathbf{r}_i} L)$
    $\hat{\mathbf{r}} = \mathbf{H}_{\text{adam}}\, \mathbf{r}$
    $\mathbf{R}^T = \text{expm}(-\mathbf{\Theta}_{\hat{\mathbf{r}}})\, \mathbf{R}^T$

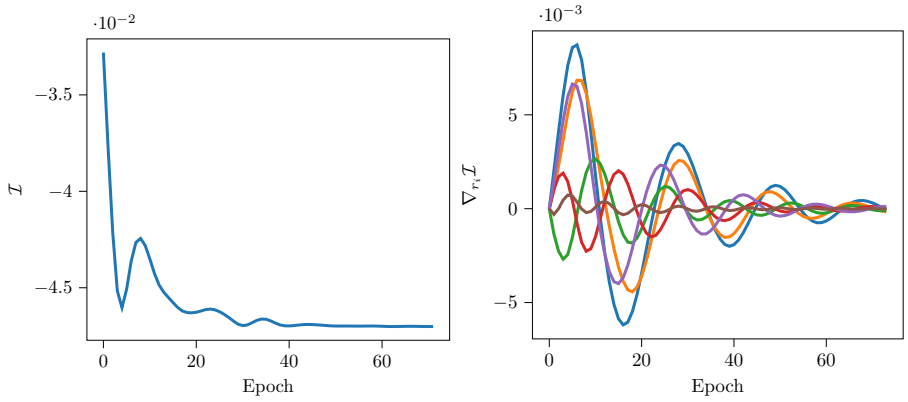


Figure 4.5: (left) Shows the loss $\arg\min_\mathbf{R} \mathcal{I}(\mathbf{X})$ during optimization, and (right) shows the derivative per coordinate r_i in Lie-algebra (6 parameters). Unmixing two Laplace components $s \sim \text{Laplace}(0, 1)$ and two uniform components $s \sim \mathcal{U}(-2, 2)$with a random mixing matrix $A \sim \mathcal{N}(0, 1)$.

4.4 Experiments

We compared Lie-Adam[1] with standard state-of-the-art ICA algorithms: For FastICA [80], we used the parallel implementation of scikit-learn [133], for L-BFGS with Lie-Group techniques [162] we used Lie-Adam and replaced ADAM by L-BFGS, for Picard-O [2] and Majorization-minimization ICA (MM) [3], we used the Python packages provided by the authors,[2][3]. for Infomax, we used both the Picard-O implementation [158] and Lie-Adam with the Infomax update rule. We implemented Lie-Adam with PyTorch [132] and defined all gradients as modules in Autograd, the differentiation package of PyTorch. We ran all experiments with single-precision on standard hardware consisting of an Intel i7 7800k, 32 GB RAM, and a 16 GB GeForce 1080Ti. For evaluation, we use the following metrics:

Non-Gaussianity. Non-Gaussianity, i.e., negentropy, is a proxy measure for independence and is computed using negentropy approximation:

$$J_{\text{neg}}(\mathbf{S}) = \frac{1}{k}\sum_{i}^{k}(\mathbb{E}[G(s_i)] - \mathbb{E}[G(\gamma)])^2. \tag{4.20}$$

Mean correlation coefficient. The mean correlation coefficient (MCC) [88] measures the maximum correlation between the mixed and unmixed images and is given by

$$\mathbb{E}[|\max_j(\mathbf{X}^T\,\mathbf{S})_{ij}|]. \tag{4.21}$$

.

Amari distance. The Amari distance is given by

$$d_{Amari}(\widehat{\mathbf{A}}, \mathbf{A}) = \sum_{i}^{n}(\sum_{j}^{n}\frac{\mathbf{P}_{ij}}{\max_k \mathbf{P}_{ik}} - 1) + \sum_{j}^{n}(\sum_{i}^{n}\frac{\mathbf{P}_{ij}}{\max_k \mathbf{P}_{kj}} - 1), \tag{4.22}$$

where $\mathbf{P} = \sqrt{(\widehat{\mathbf{A}}^{-1}\,\mathbf{A})^2}$. It is zero if $\widehat{\mathbf{A}} = \mathbf{A}$ up to permutation and scale [122]. This particular property makes it a more appropriate matrix measure compared to the Frobenius in this scenario.

[1] `https://github.com/matherm/Lie-Adam.git`
[2] `https://github.com/pierreablin/picard`
[3] `https://github.com/pierreablin/mmica`

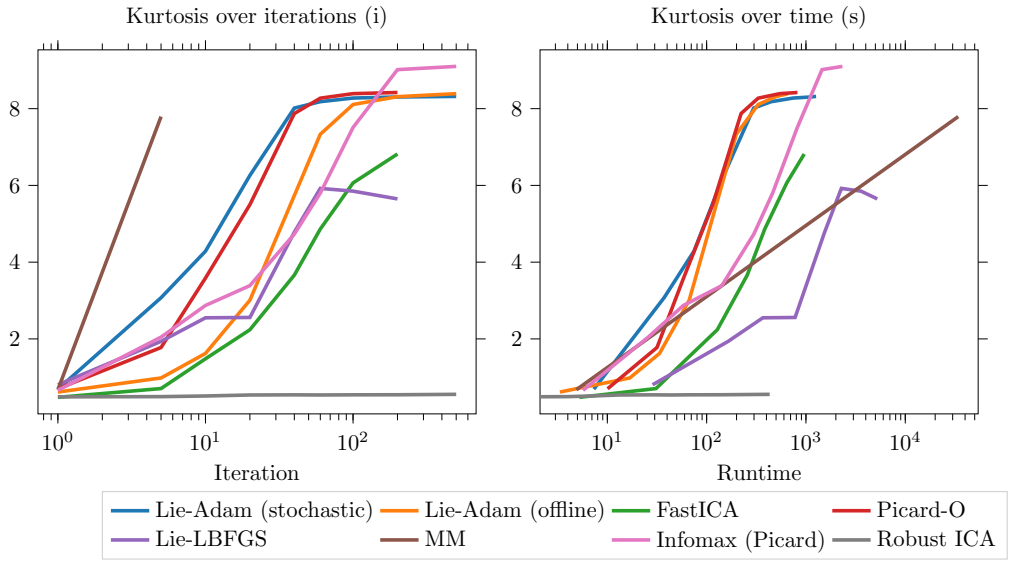


Figure 4.6: The first plot shows the evolution of kurtosis of the computed ICs measured over iterations and the second plot over runtime for the STL10 dataset (10^5 examples of size $3 \times 96 \times 96$). The learning rates are 0.01 in the offline scenario and 0.001 for the stochastic scenario. The batch size and k are 484.

Method	**Negentropy**	**MCC**	**Amari distance**
Lie-Adam (offline, $lr = 0.1$)	$\mathbf{3.3e^{-03} \pm 1.1e^{-03}}$	$\mathbf{3.7e^{-01} \pm 1.8e^{-02}}$	$\mathbf{3.0e^{-00} \pm 5.0e^{-04}}$
Lie-Adam ($bs = 6250, lr = 0.001$)	$2.1e^{-03} \pm 7.0e^{-04}$	$\mathbf{3.7e^{-01} \pm 3.0e^{-03}}$	$4.4e^{-00} \pm 1.8e^{-00}$
Lie-Adam ($bs = 625, lr = 0.0001$)	$1.6e^{-03} \pm 6.0e^{-04}$	$\mathbf{3.7e^{-01} \pm 1.5e^{-02}}$	$4.6e^{-00} \pm 1.9e^{-00}$
Lie-LBFGS	$3.1e^{-03} \pm 7.0e^{-04}$	$3.6e^{-01} \pm 3.1e^{-03}$	$3.9e^{-00} \pm 1.6e^{-00}$
Picard-O	$8.0e^{-04} \pm 0.0e^{-04}$	$3.5e^{-01} \pm 0.0e^{-00}$	$5.9e^{-00} \pm 1.6e^{-00}$
FastICA	$2.7e^{-03} \pm 7.0e^{-04}$	$3.4e^{-01} \pm 2.1e^{-02}$	$5.6e^{-00} \pm 7.1e^{-01}$
Infomax-Picard	$9.0e^{-04} \pm 0.0e^{-00}$	$3.4e^{-01} \pm 0.0e^{-00}$	$5.5e^{-00} \pm 1.2e^{-00}$
Infomax-SGD (offline, $lr = 0.01$)	$1.5e^{-03} \pm 0.0e^{-00}$	$3.3e^{-01} \pm 0.0e^{-00}$	$5.1e^{-00} \pm 0.0e^{-00}$
MM	$4.0e^{-04} \pm 0.0e^{-00}$	$2.9e^{-01} \pm 0.0e^{-00}$	$4.6e^{-00} \pm 0.0e^{-00}$
Infomax-Adam (offline, $lr = 0.01$)	$1.5e^{-03} \pm 0.0e^{-00}$	$1.9e^{-01} \pm 0.0e^{-00}$	$3.8e^{-00} \pm 0.0e^{-00}$

Table 4.1: Unmixing three images using G_2 contrast function (mean and standard deviation over ten different initialization (same initializations for all algorithms).

4.4.1 Unmixing image signals

There are two different applications or views on ICA, which sometimes confuses: (1) the unmixing task and (2) the decomposition task that identifies a statistically independent basis of the data. In the unmixing task, a data matrix $\mathbf{X}$ of mixed signals organized as column vectors is given. ICA's task is to unmix the signals by identifying the unmixing matrix $\mathbf{A}^{-1}$ [162]. One may denote the unmixed signals $\mathbf{S}$ as independent sources. We take the three 500×500 grayscale images and mix them by a known mixing matrix

$$\mathbf{A} = \begin{pmatrix} 0.8311 & 0.1345 & 0.5397 \\ -0.4464 & 0.7401 & 0.5030 \\ -0.3317 & 0.6589 & 0.6751 \end{pmatrix}. \tag{4.23}$$

We ensured that two images were sub-Gaussian and one was super-Gaussian for a fair comparison. Before mixing, the images were centered by subtracting the per-image mean and scaled to have a unit norm. This is also known in ICA literature as removing the DC-component and contrast normalization [80]. We organize the mixed images as columns yielding a 250.000×3 data matrix $\mathbf{X}$. For an equal comparison, we pre-whitened the data $\mathbf{X}$ by multiplying with the whitening matrix

$$\mathbf{W}_{\text{svd}} = \left(\frac{\mathbf{S}_{\text{svd}}^2}{n-1} \right)^{-\frac{1}{2}} \mathbf{V}_{\text{svd}} \tag{4.24}$$

obtained from standard SVD by computing $\mathbf{X} = \mathbf{U}\,\mathbf{S}_{\text{svd}}\,\mathbf{V}_{\text{svd}}^T$. Fig. 4.7 illustrates the mixed images.

As all considered implementations of the algorithms implement G_2, we parametrize all algorithms to use G_2 for estimating the unmixing matrix $\widehat{\mathbf{A}}^{-1}$. The results are summarized in Tab. 4.1. While Lie-Adam (offline, $lr = 0.1$) achieves the best mean correlation coeeficient (MCC) [88] and finds the best unmixing matrix in terms of Amari distance, the minibatch versions are similar to the compared algorithms. As MCC is not directly optimized, better MCC indicates good generalization properties. Coupling ADAM with the Infomax update rule becomes very unstable across experiments. Picard-O gives the most independent components in terms of negentropy. MM achieves an impressive high negentropy but remarkably lacks similarity as measured by correlation, which might be an indicator of overfitting. We conclude that all algorithms are performing relatively similarly, and depending on the measure, one is better than the other.

Figure 4.7: Three 500×500 images $\mathbf{S}$ mixed by $\mathbf{A}$.

Figure 4.8: Three 500×500 images $\mathbf{X}$ unmixed by $\widehat{\mathbf{A}}^{-1}$ estimated with stochastic Lie-Adam.

The important point is that Lie-Adam also works when using mini-batches and therefore is scalable to large-scale datasets. Fig. 4.8 shows the unmixed images after estimating $\widehat{\mathbf{A}}^{-1}$ with Lie-Adam.

4.4.2 Computing ICs in images

In this section, ICA is used for identifying a statistically independent basis $\mathbf{A}$ of the data $\mathbf{X}$ of samples organized as rows-vectors is given. Similar to principal components (PCs) in PCA, the identified basis vectors (columns of $\mathbf{A}$) are denoted as independent components (ICs). We tested the computation of the independent components (ICs) on the STL10 dataset [34] ($n = 10^5, d = 27648, k = 990$) for FastICA, Infomax, Picard-O, Lie-Adam (offline), and Lie-Adam (stochastic). STL10 consists of $1.05 \cdot 10^5$ RGB images of size 92×92 with 2.6 GB in total, yielding a $10^5 \times 8464$ data matrix $\mathbf{X}$. We leave out 5k samples and use them as validation data $\mathbf{X}_{\text{val}}$ for reporting results.

Fig. 4.6 shows the tracked metrics on the validation data while extracting $k = 990$ components, i.e., 99 % of the explained variance. As the minimum batch size for the full pipeline, including Batch KLT whitening [105], would be $bs = 990$, we set the batch size to that. The learning rates are $lr = 0.01$ in the offline scenario and 0.001 for the stochastic one. When looking at the plots, we see the superiority of Lie-Adam and Picard-O. The other algorithms get stuck in local minima or are slower. The best convergence rate has, as expected, the MM algorithm. In terms of kurtosis, the algorithm is still comparable. The runtime plot shows that Lie-Adam and Picard-O are the fastest, while MM is the slowest. When looking at the convergence, we see that Lie-Adam follows almost the identical path as Picard-O. Note, however, that Picard-O is not scalable to large datasets. As in every stochastic algorithm, learning rate and batch size are the most important hyperparameters. We found that larger batch sizes lead to more accurate and better results, while smaller batch sizes improve convergence in the early phase of the optimization. In all experiments, learning rates between $[10^{-3}, 10^{-2}]$ gave good results.

4.4.3 Computing ImageNet ICs.

The final experiment puts all together and computes the first $k = 484$ independent components of the ImageNet dataset ($n = 1.281 \cdot 10^6, d = 150528, k = 484$). The ImageNet challenge [154] consists of 1.281 million RGB images of size 200×200 with 144 GB in total yielding a $1.281 \cdot 10^6 \times 120000$ data matrix $\mathbf{X}$. This is not trivial as this requires storing and processing 1.2 million data samples, which requires 144 GB of RAM. With existing ICA algorithms, this learning task would require pre-whitening in a preprocessing step and applying spectral screening to reduce redundant or similar data points [162]. In Lie-Adam, we used a learning rate $lr = 0.01$ and batch size $bs = 484$ and trained the model for three runs through the dataset, which took $3h$ on our hardware. That runtime could be achieved by GPU acceleration through CUDA and the Batch KLT layer that whitens the data online during the first epoch. Fig. 4.1 shows the computed ICs from the ImageNet dataset. As expected, the components look like the usual ICs from natural images. However, we considerably improved the sharpness and clearness of the found filters.

4.5 Conclusion

We investigated the task of computing independent components in large-scale scenarios using accelerated Lie group techniques in combination with Batch KLT [147]. The combination of fast whitening, stable gradient descent, and good convergence rates is a huge improvement for scaling up current Lie-group-based ICA techniques. The proposed Lie-Adam approach offers two modes: in offline mode, it works very similar to L-BFGS-based algorithms like Picard-O [2]. However, in stochastic mode, it offers fast convergence rates while maintaining highly accurate solutions. We showed that the computation can be done efficiently without computing the matrix-exponential explicitly. The derivatives for gradient descent are computed in the corresponding Lie-algebra of the special orthogonal group, which turned out to be well-suited for the use of ADAM [90]

in combination with mini-batches. Compared to Picard-O, we do not compute the derivative through a relative gradient scheme incorporating a Taylor-expansion of the matrix-exponential. Our approach allows us to formulate the ICA update equations as a standard neural network and leverage recent approaches in deep learning. Despite the convergence guarantees of MM, its convergence rate was relatively slow in our experiments. To prove the capabilities of our algorithm, we extracted independent components of ImageNet, which is a 144 GB Dataset on standard hardware. To our knowledge, we are the first who present ICs of this popular large-scale dataset.

The ability to efficiently compute ICA enables further research directions: First, ICA for large-scale datasets like video and neural network data. Second, as our model is a fully-featured neural network, it makes integrating recent autoencoder techniques for source modeling straightforward. Finally, computing negentropy for large data matrices enables the hyperparameter selection method for selecting the optimal patch size for the μshift algorithm in Sec. 3.3.4.

CHAPTER 5

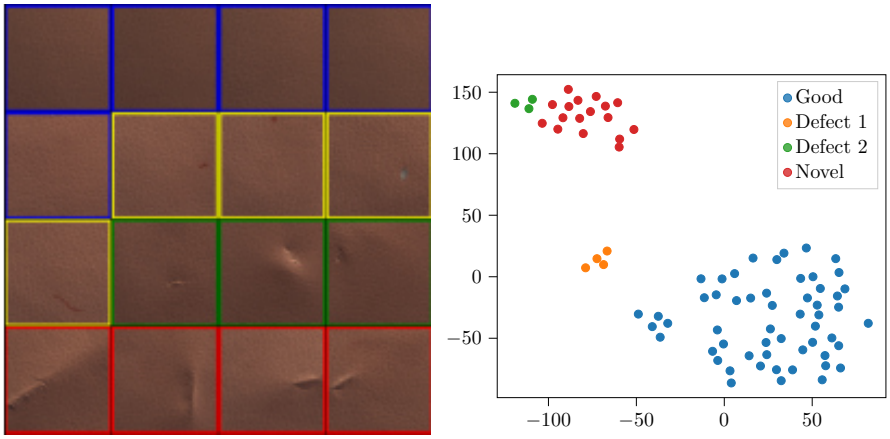
Incremental one-class learning using null space updates

Until now, we considered in the previous chapters that the available training data arises from a single class and is defect-free. We proposed using novelty detection based on one-class classifiers for such a scenario. However, another one-class learning case appears as one-class incremental learning (one-class-IL) in real-world scenarios. This case is relevant in industrial defect detection scenarios, where novel defects usually appear during operation. Existing rolled-out classifiers must be updated incrementally in this scenario with only a few novel examples. In addition, it is often required that the base classifier must not be altered due to approval and warranty restrictions. In machine learning literature, one-class incremental learning is a special case of class-incremental learning, where only a single class is incrementally added to an existing classifier instead of multiple classes. The following chapter describes our algorithm that targets such one-class-IL scenarios. It is based on the R-NSCL-associated publication that is submitted and under review [69].

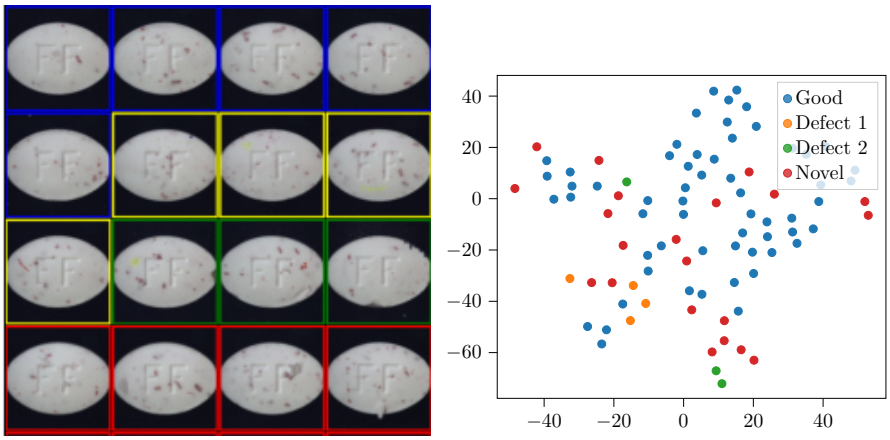
5.1 Introduction

Many real-world applications require learning from data that changes over time. Recent machine learning models, especially artificial neural networks, can learn a new task very well but tend to forget the previous ones. This is called catastrophic forgetting [115, 117], and it affects deep networks when trained on a series of tasks, making it hard for them to learn continually or in an incremental manner. This is particularly problematic for industrial defect detection, where new tasks often appear during operation and mainly consist of a single novel defect class that was unknown beforehand. Fig. 3.1 shows typical industrial production defects from the well-known MVTec dataset [14].

In this chapter, we focus on a special case of class-incremental learning (class-IL) [145] and incrementally add a single class to an existing classifier (one-class-IL). This differs from traditional continual learning scenarios, where the objective is to learn a unified classifier over incrementally observed sets of classes [177]. In this traditional setting, the primary challenge is to avoid forgetting previous classes while learning new ones. The ultimate goal is to improve average performance across all classes [145, 177]. In contrast, an additional difficulty in an industrial one-class setting is that the performance concerning previously learned classes must not be changed due to approval and warranty restrictions.



(a) Cluster scenario: Leather



(b) Diffuse scenario: Pill

Figure 5.1: Two defect scenarios from the MVTec dataset [14], i.e., the cluster scenario *Leather* and the diffuse scenario *Pill*. Note the subtle defects on the pills' edges. T-SNE [179] visualizations for the two example scenarios show the well-clustered novel leather defects and the diffusely distributed novel pill defects. We used the last layer of the ViT-B16 [46] feature space for feature extraction.

.

Although several methods have been proposed to address the problem of continual learning, there is still no common understanding of best practices [141, 114]. Interestingly, there is an additional overlap between few-shot and continual learning research, where similar problems are solved. For few-shot learning, it has been shown recently that the particular choice of algorithm is not that important for incrementally learning new classes. A simple combination of a prototype network (ProtoNet) [169] with a pre-trained feature extractor achieves state-of-the-art performances across several types of incremental learning tasks [73]. A similar finding appeared in the class-IL community where a balanced sampler-based finetuning approach can outperform various specialized formulations in most continual learning settings [141]. Unfortunately, the proposed techniques do not focus on performance guarantees for previously learned classes in both research areas.

The recently proposed null-space learning framework (NSCL) [184] is an exciting novel direction. Restricting the newly learned features for the novel class to the null space of the feature covariance matrix of the previously learned classes potentially offers a solution to the problem of not changing performance on the base classes.

Contributions. In this chapter, we investigate the null-space learning framework (NSCL) [184] for one-class-IL. While the technique is compelling in learning sets of novel classes, we found that it overfits novel classes in the one-class setting, causing performance degradation of previously learned classes. Here, we identify unconstrained weight growth in null space as the underlying issue and propose a ReLU-regularization term that penalizes the scale of the newly learned features. This chapter focuses on defect characteristics that appear in industrial defect detection scenarios. Here, we identified two different critical regimes that we call *cluster* and *diffuse* (cf. Fig.5.1). Well-separated defect classes in feature space characterize the cluster scenario, whereas, in the diffuse regime, the novel examples are highly dispersed within the examples from the base classes. During training, only on the novel examples as required by one-class-IL, backpropagation simply increases the decision region around these examples, leading to a continuously rising false positive rate by misclassifying defect-free examples as defective, which is unacceptable in industrial scenarios. Our regularized null-space learning framework (R-NSCL) adds a regularization term to the standard cross-entropy classification loss that stabilizes null-space training in the one-class scenario by counteracting the excessive growth of the novel decision regions. Our regularization can be used where NSCL is implemented, and retaining performance on existing classes is more important than the accuracy of the novel classes. Like NSCL, our method works without access to examples from previously learned classes. Since data efficiency is critical to successful industrial deployment, we evaluate all methods in few-shot protocols, where only a few novel examples are available. We compare the performance to state-of-the-art algorithms by showing the method's capabilities on two industrial datasets, namely AITEX [167] and MVTec [14].[1]

5.2 Related work

One-class incremental learning (one-class-IL) is a particular case of class-incremental learning that focuses on learning a single novel class. The problem of learning single novel concepts sits somewhere between two well-known research areas, namely continual learning [177] and few-shot learning [73]. While the traditional continual learning research focuses on learning sets of classes incrementally [145, 184], few-shot learning focuses on learning novel concepts using only a small number of observed examples [73]. The significant difference between the two approaches is that few-shot learning does not necessarily keep track of the performance of previously learned concepts [73]. Therefore, few-shot learning focuses primarily on a single novel task, not on continually learning

[1] `https://github.com/matherm/regularized-null-space`

novel concepts. However, in the one-class incremental learning (one-class-IL) setting, these two approaches overlap and share the same problem structure: Learning a novel concept given a small number of novel examples.

In the area of few-shot learning, many different approaches have been proposed [186]. However, more and more studies show strong simple-but-effective baselines [73] are competitive and show better generalization properties compared to sophisticated few-shot learners [71]. These strong baselines are mainly based on a transfer learning pipeline [192], which consists of a standard neural network, such as EfficientNet [173] or InceptionNet [171], that is pre-trained on a large source dataset, such as ImageNet. Afterward, a simple linear [31] or centroid classifier [169] is attached on top of the fixed feature extractor. Optionally, the entire pipeline is finetuned to improve overall performance [73]. While the few-shot approach fits the problem of one-class-IL, it does not guarantee the performance on previously learned classes. This is problematic when the pre-trained feature space does not easily separate the novel examples from the existing classes.

Typical class-IL algorithms include a memory buffer to store exemplars from old classes, a forgetting constraint to retain previous knowledge, and a learning system that learns novel classes [120]. iCaRL was the first approach formally introducing such a learning pipeline [145]. iCaRL alternates between feature representation learning and classifier learning. It alleviates catastrophic forgetting via knowledge distillation and a replay-based approach. Therefore, it represents a finetuning variant that expects a certain number of examples from the base classes. Successive works usually contribute to one of the three aspects of class-IL. Other methods [29, 103] solely use simple finetuning to avoid forgetting and perform very similarly to iCaRL. The recently proposed null-space learning framework [184] introduced a novel forgetting constraint by projecting gradient updates onto the null space of previously learned features. Therefore the representation of previous classes is unchanged while learning novel concepts. Additionally, only the projection operator must be stored in memory instead of a set of exemplars of previous classes. However, as we show, in the one-class setting, there is no control to prevent the network from overfitting, potentially causing the resulting classifier to classify everything as the novel class in diffuse scenarios.

Advanced Null Space (AdNS) is the most similar work to ours, which addresses the overfitting problem by a similar regularization construction in a multiclass continual learning scenario [93]. However, their approach differs in that there is no general rule for setting the regularization strength, and each task has a separate non-orthogonalized multiclass classifier, possibly degrading the performance on the base classes. This is particularly an issue in an industrial defect scenario where the novel class is similar to the existing classes and, therefore, shares the same feature subspace.

5.3 R-NSCL learning

Our method extends the standard null-space learning (NSCL) [184] to the one-class setting by adding an additional regularization term. In the first part of the section, we introduce the null-space approach in its original form. In the second part, we introduce our regularization term that is added to the loss function. Here, the class-incremental and one-class-incremental methods only differ in the dataset used for training $\mathcal{D}_{train} = (\mathbf{X}, \mathbf{y})$. While the class-incremental method adds a set of classes, the label vector $\mathbf{y}$ consists of a single unique hot-encoded label in one-class learning.

5.3.1 Null-space learning

Null-space class learning (NSCL) is a projected gradient descent method, where the layerwise gradient is projected onto the null space of a pre-computed feature covariance matrix [184]. The

critical observation is that the fully connected layers of a neural network are linear operators operating on the data feature space of the previous layer. The data feature space typically lies on a lower dimensional manifold, so the fully-connected layers must have a null space. This observation allows the design of a training algorithm that updates the parameters only inside the null space and allows model updates without altering previously learned features. The original NSCL method only updates the last L fully-connected layers.

We start with a neural network $\phi(\mathbf{x}, \mathbf{W})$ with parameters $\mathbf{W}$. For every input $\mathbf{x}$ the network predicts a probability distribution over labels $\hat{\mathbf{y}}$ using a standard softmax output layer [96],

$$\hat{\mathbf{y}} = \text{softmax}(\phi(\mathbf{x}, \mathbf{W})). \tag{5.1}$$

We consider square-sized color images $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$ as inputs, where C is the number of channels, H the height, and W the width of the images. The neural network is trained by minimizing the standard cross-entropy loss using a training dataset $\mathcal{D}_{train} = (\mathbf{X}, \mathbf{y})$:

$$\arg\min_{\mathbf{W}} \mathcal{L}_{CE}(\mathbf{y}, \hat{\mathbf{y}}). \tag{5.2}$$

For optimization, the gradient-descent optimizer ADAM [90] is used, which requires the gradient $\nabla_{\mathbf{W}} \mathcal{L}_{CE}$ of the loss function with respect to the parameters. This gradient is computed using the standard backpropagation algorithm [153].

Based on that, the null-space class learning (NSCL) algorithm projects the computed gradient using a projection operator $\mathbf{P}$. The projected NSCL gradient is given by

$$\nabla_{\mathbf{W}}^{NSCL} \mathcal{L}_{CE}(\mathbf{y}, \hat{\mathbf{y}}) = \nabla_{\mathbf{W}} \mathcal{L}_{CE}(\mathbf{y}, \hat{\mathbf{y}})\, \mathbf{P}^T\,. \tag{5.3}$$

At every iteration step i, the ADAM algorithm computes a diagonal scaling matrix $\mathbf{H}_i$, which is applied to the gradient. The final parameter update rule is given by

$$\mathbf{W}_{i+1} = \mathbf{W}_i - \mathbf{H}_i \nabla_{\mathbf{W}_i}^{NSCL} \mathcal{L}_{CE}(\mathbf{y}, \hat{\mathbf{y}}). \tag{5.4}$$

To simplify the formulation of the projection operator $\mathbf{P}$, we define it layerwise using subscript l. The projection operator is based on the layerwise uncentered covariance matrix $\mathbf{\Sigma}_l \in \mathbb{R}^{F \times F}$ of the F-dimensional feature representation ϕ of the N training examples of the previously learned classes. Using the layerwise activations $\mathbf{F}_l \in \mathbb{R}^F$, the uncentered covariance matrix is computed by

$$\mathbf{\Sigma}_l = \frac{1}{N-1} \mathbf{F}_l^T\, \mathbf{F}_l\,. \tag{5.5}$$

We must divide the covariance matrix into data and null space to derive the null-space projection (cf. uncentered Principal Component Analysis (PCA)). This is achieved by first representing the covariance matrix in its eigenvector decomposition, given by

$$\mathbf{\Sigma}_l = \mathbf{U}_l\, \mathbf{S}_l\, \mathbf{U}_l^T, \tag{5.6}$$

where $\mathbf{S}_l$ is a diagonal matrix with directional variances measured along the orthogonal directions $\mathbf{U}_l$.

The eigenvectors in $\mathbf{U}$ are ordered by variance and can be split into an orthogonal basis for the data space $\mathbf{U}^*$ and the null space $\tilde{\mathbf{U}}$, given by:

$$\mathbf{U}_l = [\mathbf{U}_l^*, \tilde{\mathbf{U}}_l], \tag{5.7}$$

where $\tilde{\mathbf{U}} \in \mathbb{R}^{\sigma \times F}$ and $\mathbf{U}^* \in \mathbb{R}^{(F-\sigma)\times F}$. The cutoff dimension is not fixed because the transition between null and data space is typically smooth. Therefore, we introduce the hyperparameter σ that controls how many dimensions of the entire space are interpreted as null space. Again, as the eigenvectors are ordered by variance, using the last dimensions automatically selects dimensions with almost zero variance. Usually, σ is computed using a rule based on the fraction of explained variance, e.g., 90 %.

Using the derived basis, the null-space projection is given by:

$$\mathbf{P}_l = \tilde{\mathbf{U}}_l \tilde{\mathbf{U}}_l^T. \tag{5.8}$$

For unconstrained training and to allow initial training of the model without previous data, the projection operator is set to identity $\mathbf{P}_l = \mathbf{I}$.

5.3.2 Regularization

Without any modification, the null-space training learns incrementally new concepts without altering existing representations. However, there is an important exception to this statement: scale. To see this flaw, note that the projected gradient updates are unconstrained in scale, and the feature detectors in null space can grow without any limit. When there is more than one novel class, the gradient updates in null space are, to some extent, automatically constrained by the classification problem. However, in the one-class case, there is no such regulator. This leads to a fundamental problem. When the novel class is neither clustered nor easy to separate in null space (diffuse scenario), the optimizer enlarges the decision regions until the novel data examples are correctly classified or the loss function reaches a plateau. Because of missing regularization, the loss function is unconstrained and can reach its minimum by simply increasing the length of the weight vector, which automatically increases the size of the decision regions around the novel examples. The region growth causes overfitting, leading to an extended classifier that predicts previously learned classes as the novel class, which we visualize in Fig. 5.2. This is particularly problematic when the features of the previously learned classes and the novel class are shared, e.g., because of similar backgrounds or defects that are hard to spot (cf. pills in Fig. 3.1).

To enforce a limit in scale, we propose penalizing the growth of the feature detectors in null space. To do so, we first compute the Frobenius norm of the projected initial feature detectors as a reference for scale. Formally the initial layerwise scale is given by:

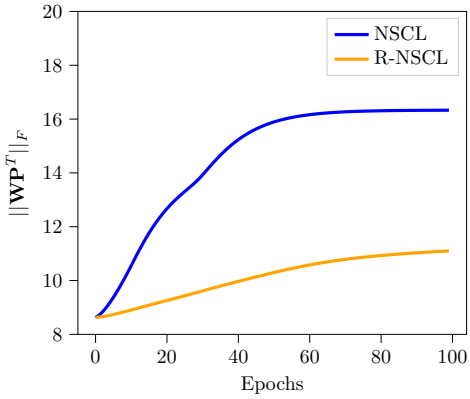
$$c_l = \|\mathbf{W}_l^0 \mathbf{P}_l^T\|_F. \tag{5.9}$$

The layerwise scale c_l specifies the maximum scaling factor for vectors in null space. This fact arises from a fundamental connection between the Frobenius norm and the spectral norm [137, 24]: The Frobenius norm is lower- and upper-bounded by the spectral norm, according to,

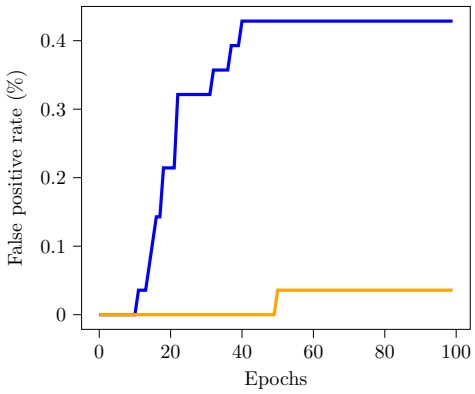
$$\|\mathbf{A}\|_2 \ \leq \|\mathbf{A}\|_F \leq \min\{\sqrt{n}, \sqrt{m}\}\|\mathbf{A}\|_2, \mathbf{A} \in \mathbb{R}^{n\times m}. \tag{5.10}$$

The spectral norm is given by

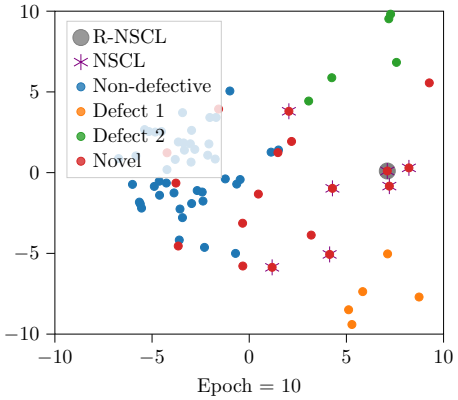
$$\|\mathbf{A}\|_2 = \sigma_{\max}, \tag{5.11}$$



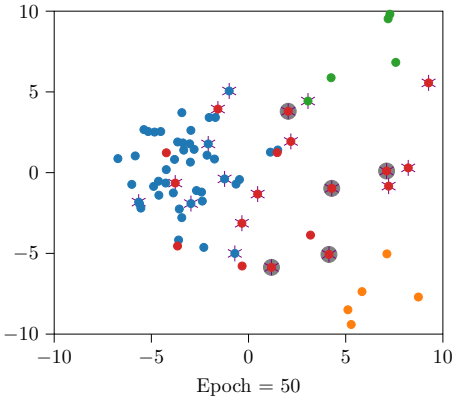
(a) Norm in nullspace



(b) False positive rate (FPR)



(c) Region growth $ep = 10$



(d) Region growth $ep = 50$

Figure 5.2: The growing norm in nullspace and the false positive rate (FPR), i.e., misclassifying defect-free good examples as defective, over epochs alongside 2D PCA visualizations for the diffuse novel capsule defects from the MVTec dataset [14]. We highlighted novel defect predictions with the star (NSCL) and disk symbols (R-NSCL). For NSCL, note the increasing number of misclassifications with increasing epochs, respectively increasing weight norm in nullspace. We used the last layer of the ViT-B16 feature space for feature extraction.

.

i.e., the most significant singular value of a given matrix, which, generally, is expensive to compute [143]. Therefore, the Frobenius norm offers an easy-to-compute bounded approximation.

We define our regularization term with respect to the previously computed layerwise scaling factor and penalize if features in null space surpass the initial scale c_l. This is achieved by applying the ReLU function, which is linear for positive values and zero otherwise. Hence, our regularization term becomes:

$$\mathcal{L}_R(\mathbf{W}) = \sum_{l=1}^{L} \mathrm{ReLU}(\|\mathbf{W}_l \, \mathbf{P}_l^T\|_F - c_l), \tag{5.12}$$

where L is the total number of fully connected layers. The regularization is not applied to the output layer, which prevents the model from learning in our experiments.

The combined training loss is given by:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}, \mathbf{W}) = \mathcal{L}_{CE}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda \mathcal{L}_R(\mathbf{W}), \tag{5.13}$$

where λ is a hyperparameter that controls the weight of the regularization term. Note that for $\lambda = 0$, our regularized null-space class learning (R-NSCL) is identical to NSCL.

5.3.3 Null-space initialization of the output layer

Given an existing M-class classifier, extending the model to a novel class is a vital implementation detail, as the total number of classes may not be known beforehand. To do so, we start with the weights of the existing output layer $\mathbf{W}_L \in \mathbb{R}^{M \times F_L}$. The novel class is then added using the imprinting technique [142]. This technique computes the features of the available novel training examples for the last layer $\phi_L(\cdot)$ and uses the arithmetic average as the new weight vector

$$\mathbf{w} = \sum_{i=1}^{K} \phi_L(\mathbf{x}_i). \tag{5.14}$$

Unfortunately, this imprinted vector may be located in data space $\mathbf{U}_L^*$. Therefore, we project the random vector onto the null space $\tilde{\mathbf{U}}_L$ before model training by applying the projection operator

$$\mathbf{w}^* = \mathbf{w} \, \mathbf{P}_L^T \, . \tag{5.15}$$

The new weight matrix of the output layer $\mathbf{W}_L^* \in \mathbb{R}^{M+1 \times F_L}$ is then given by

$$\mathbf{W}_L^* = [\mathbf{W}_L, \mathbf{w}^*]. \tag{5.16}$$

Note that our initialization is almost identical to the training of ProtoNet [169], which mimics Imprinting [142] with a cosine instead of the standard softmax classifier. However, both methods do not involve any projection operator, which is crucial for industrial inspection scenarios where defects share features with other defects, and especially non-defects.

We noticed that the covariance matrix is difficult to compute when the dimension of the feature space F is large compared to the number of examples N. This yields eigenvectors that are potentially orthogonal random projections with near zero eigenvalues, which decreases performance. To mitigate, instead of computing the covariance matrix first, we compute the needed eigenvectors $\mathbf{U}_l$ directly using the Truncated SVD algorithm [60], which automatically returns only non-zero eigenvectors. Applying Truncated SVD to the feature matrix yields the decomposition

$$\mathbf{V}_l \, \tilde{\mathbf{S}}_l \, \mathbf{U}_l^T = \mathbf{F}_l \, . \tag{5.17}$$

Again, the projection operator can be derived using the same $\mathbf{U}_l^T$.

5.4 Experiments

We evaluate our regularization-enhanced null-space class learning (R-NSCL) in several scenarios and compare it to several baselines. As a simple baseline, we use a standard finetuning approach, where the entire model is finetuned on examples from the novel and previously learned classes, i.e., the base classes. We use the well-known iCaRL finetuning algorithm [145] as an incremental-learning baseline, which also operates with all examples from the base and novel classes. Note that this is a considerable advantage, often not realistic in practice. As the few-shot learning baseline, we use the ProtoNet approach [169, 73] with the standard cosine classifier.

Because our training protocol targets industrial defect inspection, it differs from standard experiments in the incremental learning community. Therefore, we reimplemented all experiments using code available on GitHub.[2] [3] [4] For comparing the different finetuning baselines, we stop the training when the training accuracy of the novel examples reaches a certain level τ, or the maximum number of epochs ep is exceeded. For training the basis classifier, Finetune, and iCaRL, we use the balanced sampler to ensure that rare classes are not undersampled [23]. For ProtoNet, no finetuning is involved, and the needed centroid classifier can be derived directly using averages over feature vectors of available training data [169].

[2] iCaRL: `https://github.com/donlee90/icarl`
[3] ProtoNet: `https://github.com/yinboc/prototypical-network-pytorch`
[4] NSCL: `https://github.com/ShipengWang/Adam-NSCL`

	MVTec::Bottle		MVTec::Carpet		MVTec::Leather		MVTec::Tile		MVTec::Wood		MVTec::Hazelnut		MVTec::Metal nut		Average	
	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.
R-NSCL	47.9±08.3	-2.5±01.9	61.1±04.2	**00.0±00.0**	91.7±07.2	**00.0±00.0**	48.5±26.1	**00.0±00.0**	66.7±37.2	**-2.4±01.8**	51.3±04.0	**00.0±00.0**	**100.0±00.0**	00.0±00.0	66.8±24.8	-0.7±01.5
NSCL [184]	52.1±11.3	-2.5±01.9	77.8±11.0	**00.0±00.0**	91.7±07.2	**00.0±00.0**	60.6±16.9	**00.0±00.0**	66.7±37.2	**-2.4±01.8**	59.0±07.9	**00.0±00.0**	**100.0±00.0**	00.0±00.0	72.8±22.4	-0.7±01.5
AdNS [93]	**100.0±00.0**	-88.9±00.0	**93.8±04.2**	**00.0±00.0**	**100.0±00.0**	-3.2±00.0	**100.0±00.0**	-96.6±00.0	**100.0±00.0**	-6.0±04.1	**76.9±07.7**	-2.2±00.0	**100.0±00.0**	-3.6±00.0	**96.1±08.2**	-32.5±42.9
ProtoNet [169]	37.5±06.2	**00.0±00.0**	47.2±12.7	**00.0±00.0**	86.1±09.6	**00.0±00.0**	60.6±18.9	**00.0±00.0**	66.7±41.6	-3.6±00.0	56.4±04.4	**00.0±00.0**	98.1±03.2	**03.6±00.0**	64.7±25.5	**00.0±02.0**
iCaRL [145]	35.4±13.0	-3.7±03.7	80.6±04.8	**00.0±00.0**	88.9±04.8	**00.0±00.0**	72.7±00.0	**00.0±00.0**	86.7±11.5	-9.5±05.5	51.3±08.9	-7.2±01.3	**100.0±00.0**	-36.9±02.1	73.6±22.6	-8.2±12.7
Finetune	58.3±03.6	-3.7±00.0	52.8±09.6	**00.0±00.0**	**100.0±00.0**	**00.0±00.0**	75.8±10.5	**00.0±00.0**	80.0±34.6	-8.3±05.5	61.5±07.7	**00.0±00.0**	**100.0±00.0**	-0.0±03.6	75.5±21.9	-1.7±03.7

Table 5.1: One-class incremental-learning on the cluster scenarios of the MVTec dataset [14]. The results are for $k = 5$ shots using three different seeds for every experiment. Hyperparameters were carefully selected and set to $\theta = \{ep = 150, \tau = 100\%, \sigma = 0.9, \lambda = 10^{32}\}$. We measure the relative change in accuracy on the novel class (Acc. N.) and the base classes (Acc. B.).

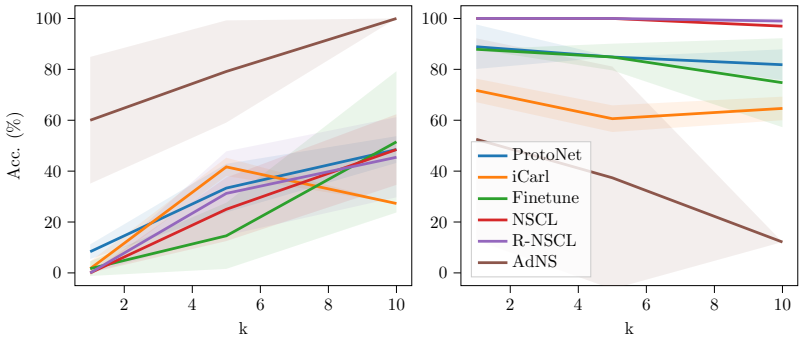
	MVTec::Pill		MVTec::Grid		MVTec::Capsule		MVTec::Screw		MVTec::Transistor		MVTec::Cable		MVTec::Zipper		Average	
	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.
R-NSCL	31.2±16.5	**00.0±00.0**	22.2±09.6	**-3.3±05.8**	22.9±07.2	**01.2±02.1**	46.7±05.8	**-1.7±02.9**	53.3±11.5	**2.3±02.0**	22.2±08.6	**00.0±00.0**	47.2±21.5	**00.0±00.0**	34.9±16.7	**-0.8±02.6**
NSCL [184]	25.0±12.5	**00.0±00.0**	38.9±09.6	-11.1±06.9	54.2±09.5	-9.5±05.5	58.3±24.7	-2.5±02.5	53.3±11.5	-5.7±02.0	**44.4±31.0**	**00.0±00.0**	52.8±24.0	**00.0±00.0**	45.3±20.5	-3.9±05.2
AdNS [93]	**79.2±20.1**	-62.6±43.7	**88.9±09.6**	-90.0±00.0	**60.4±07.2**	-16.7±02.1	**100.0±00.0**	-88.3±09.5	**93.3±11.5**	-73.6±27.9	38.9±09.6	-1.1±01.9	58.3±22.0	-4.4±01.9	**74.1±23.9**	-48.1±41.0
ProtoNet [169]	33.3±09.5	-15.2±00.0	50.0±16.7	-15.6±09.6	25.0±06.2	02.4±02.1	63.3±10.4	-19.2±02.9	40.0±20.0	-6.9±00.0	05.6±09.6	**00.0±00.0**	27.8±19.2	-3.3±00.0	35.0±21.2	-8.2±08.6
iCaRL [145]	41.7±03.6	-39.4±05.2	16.7±00.0	-38.9±10.7	20.8±03.6	-19.0±05.5	28.3±02.9	-50.0±02.5	**93.3±11.5**	-42.5±02.0	27.8±09.6	-21.1±03.8	**69.4±04.8**	-33.3±03.3	42.6±27.5	-34.9±11.7
Finetune	14.6±13.0	-15.2±05.2	33.3±00.0	-5.6±06.9	14.6±13.0	-2.4±07.4	41.7±15.3	-5.0±00.0	40.0±20.0	-3.4±03.4	38.9±25.5	-15.6±26.9	50.0±14.4	-2.2±01.9	33.3±18.8	-7.0±10.8

Table 5.2: One-class incremental-learning on the diffuse scenarios of the MVTec dataset [14]. Results are for $k = 5$ shots using three different seeds for every experiment. Hyperparameters were carefully selected and set to $\theta = \{ep = 150, \tau = 100\%, \sigma = 0.9, \lambda = 10^{32}\}$. We measure the relative change in accuracy on the novel class (Acc. N.) and the base classes (Acc. B.).

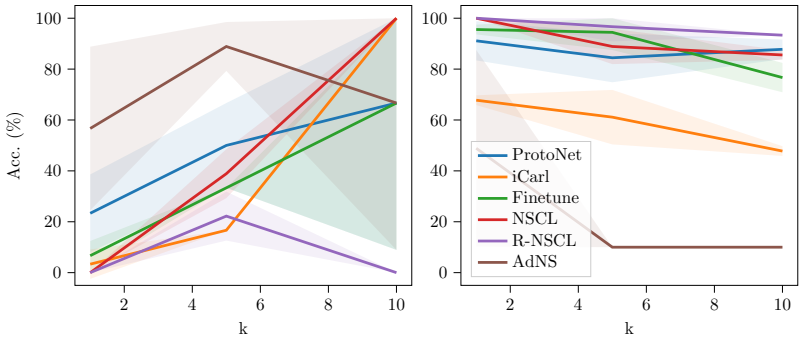
77

	AITEX::Crease		AITEX::Cut selvage		AITEX::Fuzzyball		AITEX::Broken yarn		AITEX::Broken end		Average	
	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.	Acc.N.	Acc.B.
R-NSCL	33.3±17.8	**00.0±00.0**	37.3±10.3	-1.5±00.1	33.1±04.0	**-0.4±00.6**	74.9±10.8	**00.3±00.0**	61.8±15.2	**00.1±00.2**	48.1±20.6	**-0.3±00.7**
NSCL [184]	**83.3±15.4**	-1.3±01.3	53.2±08.4	-4.5±02.8	**46.8±09.1**	-4.2±05.8	86.6±10.4	00.2±00.1	**78.5±07.4**	-1.0±00.8	**69.7±19.1**	-2.2±03.1
ProtoNet [169]	46.7±11.5	-2.0±01.6	**60.0±00.0**	-5.2±02.3	20.0±34.6	-3.2±02.6	**86.7±23.1**	-0.3±00.1	46.7±30.6	-0.2±00.1	52.0±30.0	-2.1±02.4
iCaRL [145]	55.3±10.4	-3.6±03.8	41.0±08.7	-15.1±05.8	29.1±10.7	-11.0±10.2	74.1±10.2	-4.1±02.2	56.2±12.8	-1.4±00.9	51.1±18.2	-7.0±07.1
Finetune	68.4±27.0	-0.8±01.7	30.7±07.8	**-1.3±01.8**	29.9±06.7	-1.5±02.6	79.4±10.0	-0.3±00.6	52.1±13.4	-0.8±01.7	52.1±24.1	-0.9±01.6

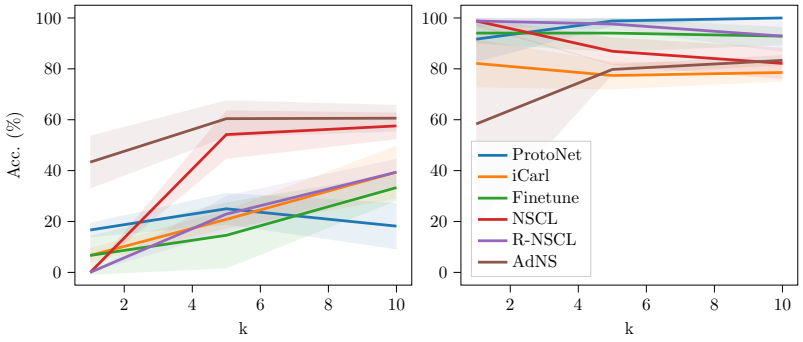
Table 5.3: One-class incremental-learning on the AITEX dataset [167]. The results are for $k = 5$ shots using three different seeds for every experiment. Hyperparameters were carefully selected and set to $\theta = \{ep = 150, \tau = 100\%, \sigma = 0.9, \lambda = 10^{32}\}$. We measure the relative change in accuracy on the novel class (Acc. N.) and the base classes (Acc. B.).



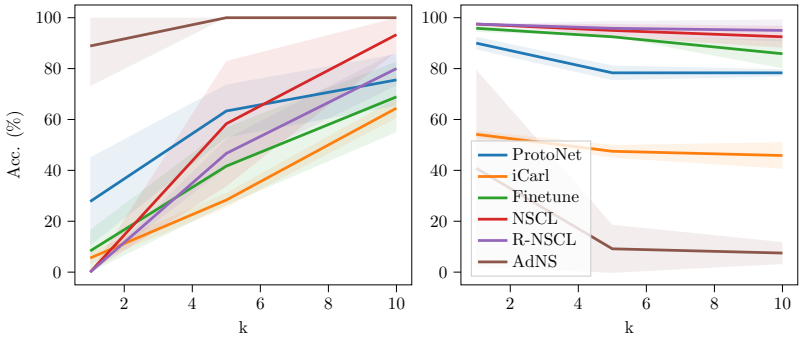
(a) Pill: novel class (left), base classes (right).



(b) Grid: novel class (left), base classes (right).



(c) Capsule: novel class (left), base classes (right).



(d) Screw: novel class (left), base classes (right).

Figure 5.3: Few-shot incremental-learning for a different number of novel examples k and several classes of MVTec [14]. We used an ImageNet pre-trained ViT-B16 [46] as a feature extractor for all experiments. The error bars are standard deviations using different seeds and novel examples.

5.4.1 Pre-trained vision transformer features

Strong-but-simple pipelines based on transferring pre-trained neural networks [73] show that the key to few-shot learning is the chosen feature space that needs to be sufficiently rich. Therefore, we tested several well-known extractors, i.e., ViT-B16 [46], EfficientNet-B4 [173], and InceptionNet [171], that were pre-trained on ImageNet and downloaded from torchvision[5]. Our studies showed an interesting difference between modern vision transformers (ViT) and convolutional neural networks (CNNs). In our tests, only the ViT offered enough separating features in null space, whereas the CNNs were very hard to train in several classes and did not reach a satisfying performance level. For instance, we achieved zero accuracies on novel MVTec::Leather defects using R-NSCL with InceptionNet as the feature space.

For practical applications, inference times of the features are of particular interest. Therefore we conducted a runtime test across the different extractors. On standard hardware with an i7-7800 and an NVIDIA RTX 3080, we measured 35 frames/s for InceptionNet, 32 frames/s for EfficientNet, and 30 frames/s for ViT using $3 \times 224 \times 224$ sized images as input and PyTorch 2.0 [132]. Because of the excellent tradeoff between inference time and accuracy, we focus on Vision Transformer (ViT-B16) throughout the rest of the chapter.

5.4.2 Industrial defect detection

The main focus of this chapter is an industrial defect scenario where the task is to detect novel defect types. We use the industrial defects dataset MVTec [14] to analyze that scenario. The MVTec dataset consists of 15 classes with at least four defect types for each class. We pre-train a base classifier with two defect types and the defect-free class. On top of that, we incrementally learn a randomly sampled single novel defect type chosen from the remaining defect types. For pre-training, we use stochastic ADAM optimization with a learning rate $\eta = 10^{-6}$ for the feature extractor and $\eta = 10^{-3}$ for the classifier, which has $L = 2$ fully connected hidden layers. The mini-batch size is set to 20. We use early stopping based on a 10 % held-out validation set to ensure convergence of the base classifier training. As the defect classes of the datasets are highly imbalanced, we use a balanced sampler that ensures that the mini-batches have uniform class distribution by respecting the number of available examples per class through class-weighted sampling [23].

We noticed simple cluster and challenging diffuse scenarios during experiments in the dataset. While the pill, grid, capsule, screw, cable, zipper, and transistor defects are diffuse, the defects of the scenarios bottle, carpet, leather, tile, wood, hazelnut, and metal nut are already clustered in feature space and therefore easier to learn. To illustrate the difference between the cluster and diffuse scenario, we show the ViT-feature space in 2D using T-SNE [179] with perplexity $PP = 10$ in Fig. 5.1. This is important as we will see that the methods behave differently in both cases. The toothbrush class is excluded from the dataset as it contains no labeled defect types. Further, some classes have only a few examples per defect. Therefore using $k > 5$ across all classes is problematic.

As a second industrial dataset, we tested the AITEX fabric defect dataset [167]. This dataset consists of seven fabrics with 12 different defect types. Not all defect types occur on any fabric, so we only use the following eight classes: No defect, Nep, Warp ball, Crease, Cut selvage, Fuzzyball, Broken yarn, and Broken end. Again, we pre-train the 3-class base classifier with the first three classes and incrementally learn one of the novel defects.

[5] `https://pytorch.org/vision/0.11/models.html`

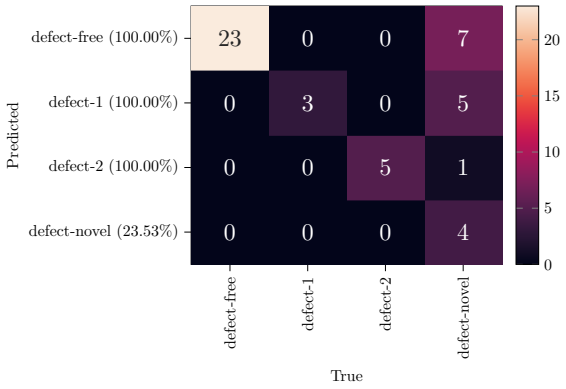
5.4.3 Cluster scenarios

We evaluated the accuracies of the novel class and the previously learned classes to keep track of both performances. Tab. 5.1 shows the results for $k = 5$ for seven simple scenarios of the MVTec dataset and the average performance. When we look at the accuracies of the base classes, we see that across methods, those are little affected by adding the novel defect type. On the other hand, when we look at the accuracies of the novel defect type, the performance differences are significant, and there is no clear best technique across all defect types. Generally, the R-NSCL loses a bit of accuracy for the novel class but improves the loss on the basis classes relative to the other methods in return, which is the main focus of this work.

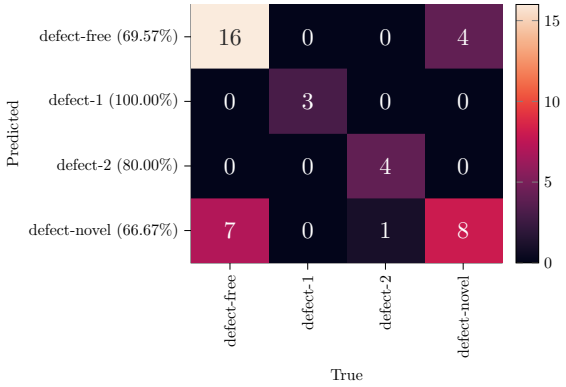
5.4.4 Diffuse scenarios

When analyzing the diffuse scenarios of the MVTec dataset, the advantage of the applied regularization becomes more apparent. Tab. 5.2 summarizes the results for $k = 5$ for seven diffuse scenarios of the MVTec dataset alongside the total averages. Especially for ProtoNet, iCaRL, AdNS, and Finetune, the performance on the base classes decreases significantly for some classes, with more than 10%, which is enormous for most relevant industrial applications. The critical observation is that the classification performance heavily depends on the particular scenario. Besides that, we see that every tested method decreases the performance on the base classes. Third, by using R-NSCL, the performance loss can be effectively reduced. Interestingly, we confirm the literature that ProtoNet [73] is a surprisingly effective method if targeting the average performance of the unified classifier, although, for MVTec, the NSCL method performs better on average. Fig. 5.4 shows confusion matrices for the capsule defects. Here, the effect of the regularization scheme becomes evident as it successfully prevents the misclassification of the defect-free examples. To investigate the behavior for different numbers of k available training examples, we plot the accuracies for the novel class and the base classes in Fig. 5.3. The significant performance losses of iCaRL and ProtoNet are especially remarkable. We noticed that the initial orthogonalization is responsible for that behavior and could reproduce the behavior when we omit the orthogonal initialization in R-NSCL (cf. Eq. 5.15).

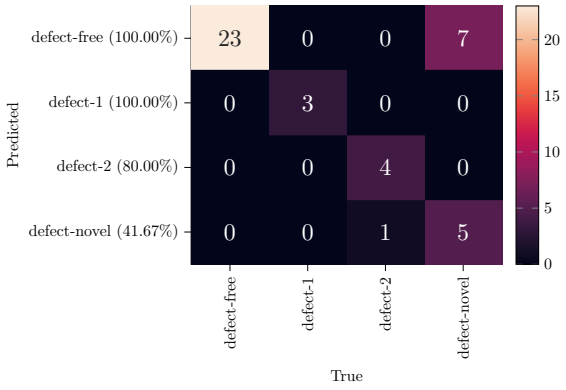
The fabric defects of the AITEX dataset have mostly a diffuse character, and the decrease in performance is significant, especially for NSCL and iCaRL. We summarize the results in Tab. 5.3. Interestingly the ProtoNet method performs quite well here, sometimes offering almost zero performance loss on the base class. For iCaRL, the variance is quite large, indicating the method's overall lack of robustness in the few-shot setting. This instability could possibly arise from the special training procedure, consisting of classifying novel examples and regressing old features simultaneously, combined with an imbalanced dataset.



(a) ProtoNet



(b) NSCL

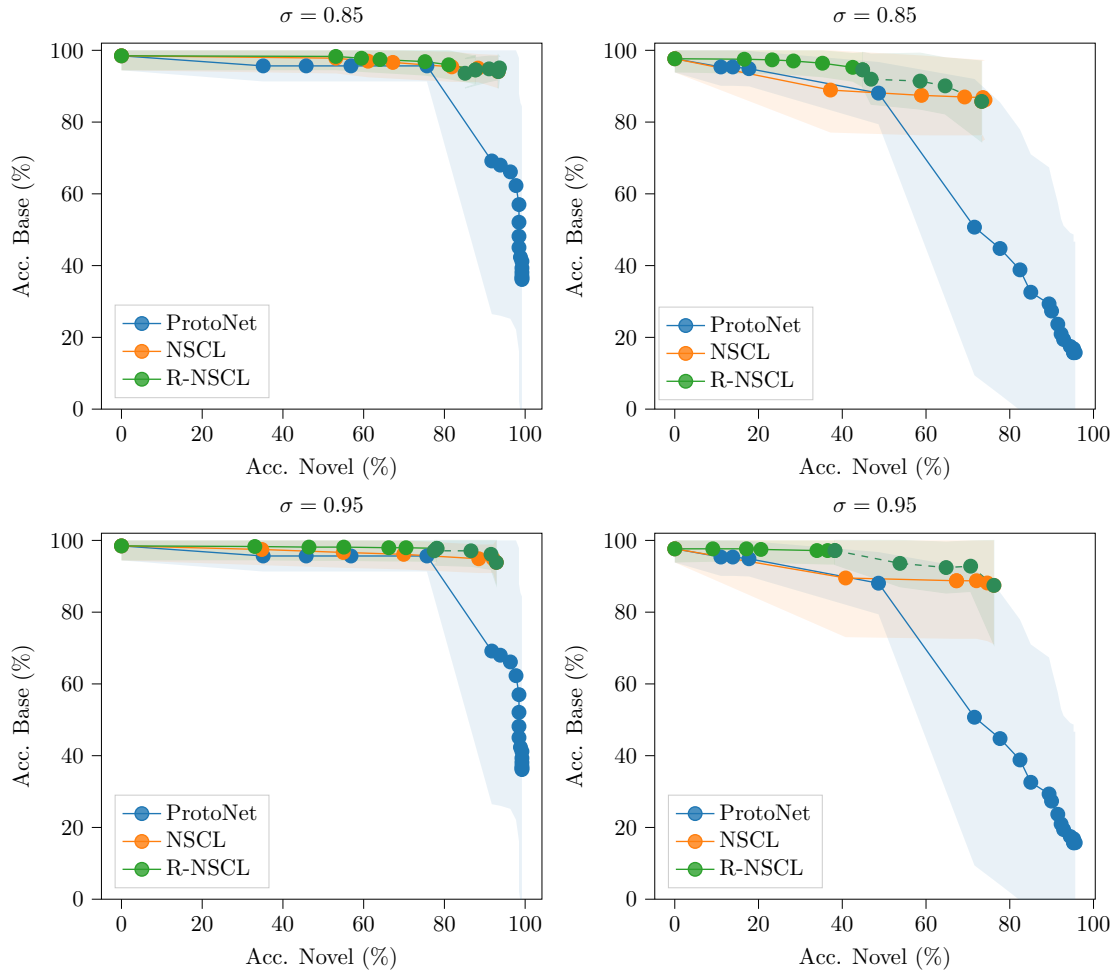


(c) R-NSCL

Figure 5.4: Confusion matrices for the MVTec::Capsule scenario [14]. While ProtoNet cannot classify the novel class, the NSCL amplifies the defect-free class so that it gets misclassified as a novel defect. Our regularization term effectively counteracts that behavior.

5.4.5 Trade-off between loss and gain

When analyzing the cluster and diffuse scenarios, it becomes evident that there is a trade-off between the performance on the base classes and on the novel class. We investigated this effect using trade-off curves in Fig. 5.5. For NSCL and R-NSCL we modulated the stop parameter τ and σ. The complete hyperparameter vector for the experiment is given by $\theta = \{ep = 150, \tau = [0\%, \infty], \sigma = \{0.85, 0.95\}, \lambda = [0, 10^{32}]\}$. For ProtoNet, we multiplied a scaling factor onto the novel prototype in the range $[0, 2]$ for smoothly phasing in the novel class. While in the cluster scenario, the NSCL method dominates by reaching almost perfect accuracy, it overfits in the diffuse scenario. Here, the regularization term effectively prevents the model from overfitting to the novel examples. We also tested different values for λ in this scenario. As expected, the performances interpolate between the two extremes of NSCL and R-NSCL as visualized by the dotted line in Fig. 5.5.



(a) Cluster scenarios

(b) Diffuse scenarios

Figure 5.5: Trade-off curves as averages over the simple cluster and diffuse MVTec scenarios for $k = 5$ novel examples and $\sigma \in \{0.85, 0.95\}$. All experiments were repeated using three different seeds. Our regularization scheme successfully reduces overfitting in the case of diffuse and overlapping classes and consistently improves standard NSCL. The dotted line highlights the reduction of the regularization strength $\lambda \in [10^{32}, 10^{20}, 10^{3}, 10^{2}, 0]$. For $\lambda = 0$, the R-NSCL method is equal to NSCL.

5.5 Conclusion

In this chapter, we analyzed the potential of null-space class-incremental learning (NSCL) [184] for one-class learning. Theoretically, the technique allows for learning novel features while retaining the learned representation. However, we found that the projection method is flawed as it does not consider the scale of the newly learned features. This is particularly an issue when the novel class is not well separated in feature space (diffuse scenario), causing severe performance degradation on the base classes.

To mitigate this issue, we proposed a regularization term that penalizes the norm of the newly learned features. As we showed in our experiments, regularized null-space class-incremental learning (R-NSCL) not only mitigates overfitting, e.g., in MVTec::Capsule, but can even improve accuracy, e.g., in MVTec::Pill. Our regularization can be used where NSCL is implemented and works without access to examples from the previously learned classes. Compared to existing methods, the main drawback is that discriminating from existing classes in null space is only possible if sufficient features for separation are left. Notably, the Vision Transformer [46] has advantages here. As this may not be the case in all practical use cases, methods that operate on the entire feature space, such as ProtoNet [169], have advantages in this respect. However, operating on data space potentially leads to performance loss on the base classes, which is the main problem for which R-NSCL was designed.

The original NSCL method [184] only updates fully-connected layers. In principle, as almost any commonly used layer in a neural network can be represented as matrix multiplication, this method is not limited to fully-connected layers. Hence, it could be extended to other layer types, such as convolution layers, which we leave for future work. Second, the combination of R-NSCL and finetuning is promising as this could learn novel discriminating features in case of overlapping diffuse scenarios and improve performance on the novel class significantly.

CHAPTER 6

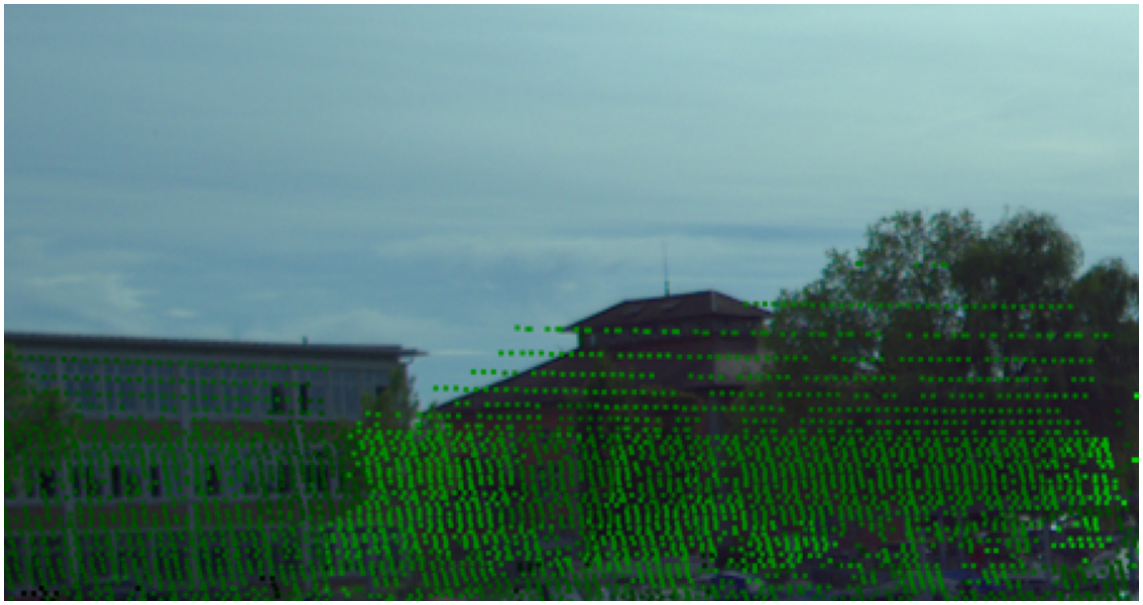
Multi-sensor fusion by optimizing mutual information

In the previous chapters, the weakly labeled data for one-class learning was captured by cameras, and the industrial defect detection solely relied on images. In this chapter, we extend this unimodal system to multi-modal sensor configurations. For this, a Lidar sensor is integrated to incorporate range data. Unfortunately, during the time of the dissertation project, the available optical inspection system could not be extended by a range sensor, and there are no usable public available datasets in the area of industrial surface inspection. Therefore, the sensor fusion approach is developed using the institute's own multi-sensor system consisting of a camera and a Lidar sensor. Here, we align the temporally aligned image and point cloud signals. The goal of the fusion process is as an additional depth channel for the images that contains the depth information from the Lidar system. Such data representation is commonly known as RGBD, where RGB comprises the normal color channels, and D stands for depth. Having such a fused data representation, the application of novelty detection using Eq. 3.1, estimating ICA using Eq. 4.17, or incrementally learning using Eq. 5.1 is straightforward as the depth can be interpreted as just another input channel. The following chapter describes our sensor fusion technique LMI and is based on the associated publication [64].

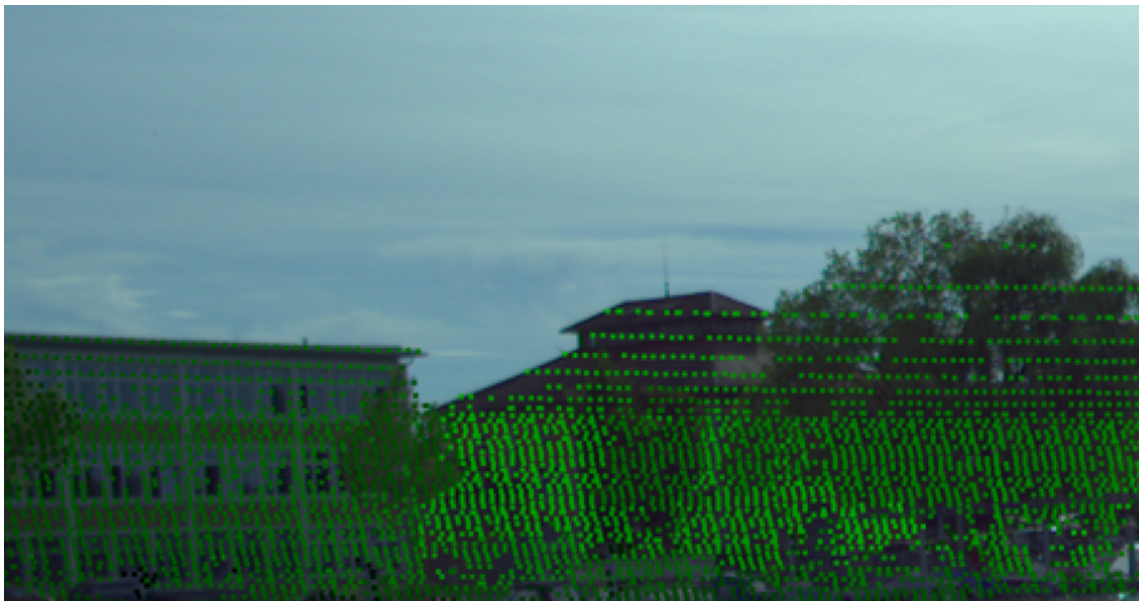
6.1 Introduction

For sensor fusion tasks such as point cloud coloring or acquiring depth information for downstream tasks (e.g., scene reconstruction or object recognition), estimating the registration transformation between the camera and Lidar coordinate systems is essential. Such a transformation is denoted by $\mathbf{E}$ and consists of a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$. Together, they describe the relative position of one sensor with respect to another. Compared to a manual process, automatic registration algorithms try to estimate the parameters $\mathbf{E}$ based on data automatically. In this chapter, we are addressing the targetless case [193, 33], which does not require a known pattern [119] such as a checkerboard. In the literature, often mutual information (e.g., [37]) has been applied [130, 84], as well as learning-based schemes [159, 198, 112], that learn the calibration, given a labeled training dataset. In that line of research, we propose a hybrid algorithm that uses a patch-based mutual information maximization scheme.

Contributions. Our information fusion method is hybrid in the sense that it uses registered training data for pre-training but involves an optimization scheme for unregistered test data. The



(a) Initial calibration $\mathbf{E}_{init}$



(b) LMI-optimized calibration $\mathbf{E}_{mi}$

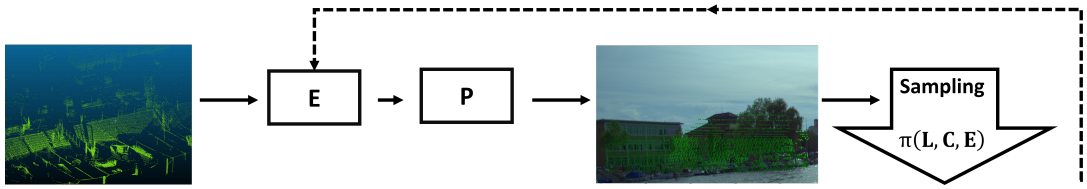
Figure 6.1: Visual registration results on the unseen maritime scenario. The accuracy can be seen best around the roof edge or the tree. The model has been pre-trained on KITTI [50], indicating good generalization capabilities.

method is based on projecting the Lidar point cloud onto the common image plane and extracting local patches by differentiable sampling [82]. As a prerequisite for the common image plane, the intrinsic camera parameters, i.e., the camera matrix, need to be available beforehand — extrinsic parameters can be chosen freely. The major advantage of such a local scheme is that general local shape patterns are more transferable to unseen scenarios. This allows the method to produce both accurate solutions and improved generalization compared to existing approaches, which makes the method suitable for situations where calibration is required (e.g., caused by weather conditions or mechanical changes), but no labeled training data are available. This is especially important for online or re-calibration scenarios. We show the effectiveness of our method on the KITTI dataset and test its generalization performance within a real-world maritime scenario. Our code is published on GitHub. [1]

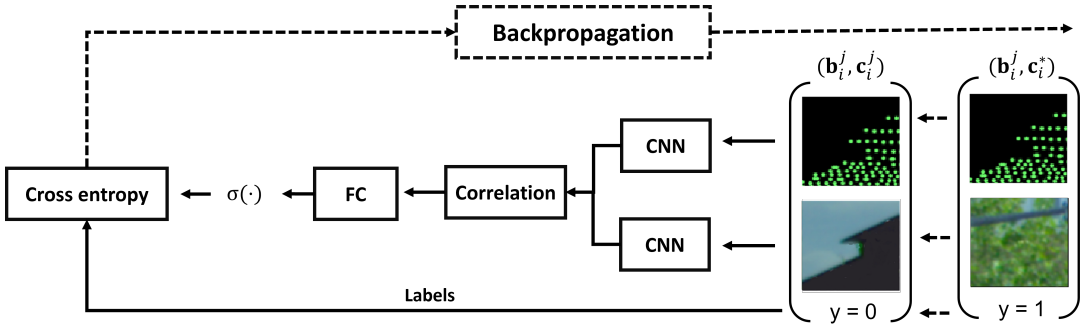
6.2 Related work

There are two major approaches to targetless registration of camera and Lidar streams. The first approach is based on optimization and works by optimizing two unregistered streams with respect to some performance metrics, such as the alignment of detected and projected 3d edges with the corresponding 2d image edges [28]. Alternatively, measuring and optimizing the mutual information between the intensities of projected Lidar points and the corresponding image pixel intensities directly has been proposed [130]. Besides, techniques from robotics incorporate a visual-odometry pipeline for estimating the extrinsic parameters by matching odometry trajectories obtained from both sensor signals [123]. Recent methods based on deep learning architectures, e.g., RegNet [159], learn to regress the 6 parameters of the extrinsic calibration directly based on a labeled dataset consisting of examples with known calibration parameters. Here, CMRNet [30] extends RegNet with an additional refinement procedure that consists of multiple models trained on different scales in a coarse-to-fine procedure. LCCNet [112] is similar in architecture, but adds an additional correlation layer for feature matching and introduces a cost volume over matched features between Lidar and camera. The combination of feature matching and hierarchical aggregation improves overall accuracy. Recently proposed hybrid methods for the calibration task use available pixel-wise labels such as object boundaries and object affinity to register the camera and point cloud data [185]. However, even though the registration quality is considerable, pixel-wise annotations are expensive to obtain. Our approach is similar in that we also combine deep learning with an optimization procedure. However, our approach differs as we are using small patches from raw images instead of relying on labeling or pixel-wise features that are often not available in practical situations.

[1] `https://github.com/matherm/Patch-MI-registration`.



(a) Differential patch sampling



(b) Mutual information maximization

Figure 6.2: The Lidar point cloud gets transformed by $\mathbf{E}$ and projected by the known and fixed camera matrix $\mathbf{P}$. Afterward, image patches $\mathbf{c}_j$ are sampled centered around the Lidar projections $\mathbf{u}_j$. For estimating mutual information, the registered patch tuples $(\mathbf{b}_i^j, \mathbf{c}_i^j)$ are labeled with label $y = 0$. The contrast examples consist of randomly picked camera patches $(\mathbf{b}_i^j, \mathbf{c}_i^*)$ and are labeled $y = 1$. The extrinsic calibration matrix $\mathbf{E}$ is optimized by solving the classification problem in Eq. 6.7, where the gradient of the cross entropy loss is computed by backpropagation.

6.3 Patchwise LMI registration

The goal of our method is to compute the extrinsic calibration parameters

$$\mathbf{E} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{6.1}$$

of two non-registered but temporally synchronized Lidar and camera datasets with N frames. The raw RGB camera images are stored as $\mathbf{C}_i \in [0,1]^{3\times H\times W}$. The Lidar point clouds are matrices $\mathbf{L}_i \in \mathbb{R}^{4\times K_i}$ with K_i 3d points in homogeneous coordinates. Throughout the chapter, we use i to reference a single frame $(\mathbf{L}_i, \mathbf{C}_i)$ and use j to reference a single Lidar point. Our goal is to optimize mutual information $\mathcal{I}$ between patch ensembles extracted from Lidar and camera frames with respect to the calibration parameters $\mathbf{E}$. Optimizing globally is problematic, as the projected Lidar signal is very sparse. Moreover, transferable local image and shape features are underrepresented compared to the empty parts of the projections (cf. Fig. 6.6), i.e., most of the pixels are empty. Therefore, we use a local approach and first project the Lidar points with the known projection matrix $\mathbf{P}$ onto the common image plane and then sample local patches from both sensors. Note that for a common image plane, at least the intrinsic camera parameters need to be available or obtained by camera calibration. The full method is a two-step procedure with an estimation, possibly offline, and an optimization step. In the estimation step (cf. Sec. 6.3.1), a neural network is pre-trained to estimate the mutual information between patch ensembles based on registered training data $\mathcal{D}_{train} = (\mathbf{L}_{train}, \mathbf{C}_{train}, \mathbf{E}_{true})$. In the optimization step (cf. Sec. 6.3.2), the mutual information between patch ensembles extracted from miscalibrated sensors is maximized with respect to the

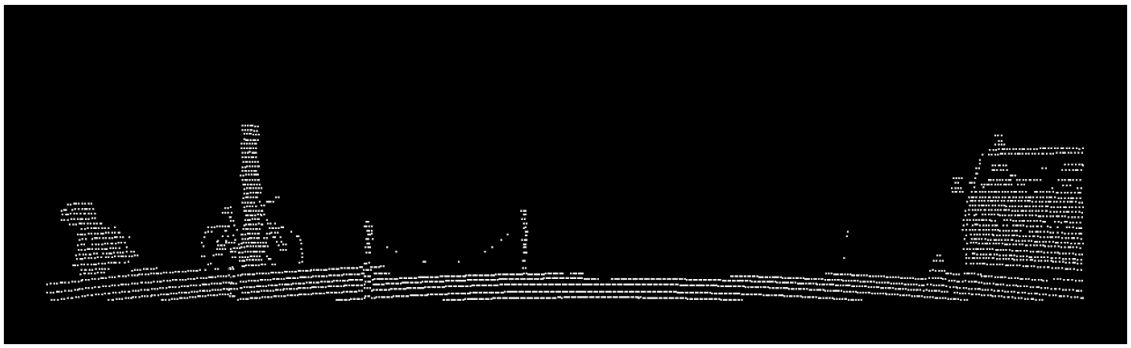


Figure 6.3: Visualization of a projected Lidar point cloud $\mathbf{B}_i$. Note the sparsity of the projected point cloud.

unknown calibration parameters $\mathbf{E}$ based on an unregistered test set $\mathcal{D}_{test} = (\mathbf{L}_{test}, \mathbf{C}_{test}, \mathbf{E}_{init})$. Fig. 6.2 illustrates the overall architecture. Fig. 6.3 shows a projected example Lidar point cloud.

6.3.1 Estimating mutual information

The following section introduces mutual information neural estimation (MINE) for estimating the shared information between projected Lidar points and camera images.

Mutual information neural estimation

The mutual information $\mathcal{I}$ is a non-linear dependency measure based on shared information between two variables. In the given calibration scenario, we use the measure to maximize the registration between projected Lidar points $\mathbf{B}$ and camera images $\mathbf{C}$ in the common image plane.

However, instead of optimizing entire images and point clouds, we follow a local scheme and optimize the mutual information of extracted local patches $\mathbf{b}$ and $\mathbf{c}$ instead:

$$\mathcal{I}(\mathbf{b}, \mathbf{c}) = \mathcal{H}[\mathbf{b}] - \mathcal{H}[\mathbf{b}|\mathbf{c}]. \tag{6.2}$$

Existing mutual information estimation techniques rely either on binning [37] or k-nearest neighbors statistics [94]. Both techniques are unreliable in high dimensions [11]. Also, gradient optimization using these estimators is difficult, as the derivatives generally do not exist [130]. However, recently a new class of mutual information neural estimators (MINE) was proposed [11]. These techniques transform the estimation problem into a binary classification problem, which can be solved efficiently using deep neural networks. The network is trained to separate registered $(\mathbf{b}, \mathbf{c})$ from non-registered $(\mathbf{b}, \mathbf{c}^*)$ tuples of patches and hereby implicitly estimates the mutual information between the two modalities. As neural networks are differentiable by design, gradient propagation with respect to the calibration $\mathbf{E}$ parameters is straightforward using backpropagation.

The needed training data $(\tilde{\mathbf{X}}, \mathbf{y})$ for the binary classification problem consists of two different M-sized sets of patch tuples

$$\tilde{\mathbf{X}} = \{\mathbb{J}, \mathbb{M}\}, \quad \mathbf{y} = \{\mathbf{0}, \mathbf{1}\}. \tag{6.3}$$

In our case, the tuples $(\mathbf{b}, \mathbf{c}) \in \mathbb{J}$ are registered Lidar-camera patches, whereas the tuples $(\mathbf{b}, \mathbf{c}^*) \in \mathbb{M}$ are random and therefore non-registered Lidar-camera patches. The class indicator function

$$p(\tilde{\mathbf{x}} = 0) = f_\theta(\tilde{\mathbf{x}}) \tag{6.4}$$

is parameterized by a neural network f_θ with trainable parameters θ. As an optimization criterion, the well-known cross-entropy

$$\arg\min_\theta CE(f_\theta(\tilde{\mathbf{X}}), \mathbf{y}) = \frac{1}{2M} \sum_i^{2M} \mathbf{y}_i \log \sigma(f_\theta(\tilde{\mathbf{x}}_i)) \tag{6.5}$$

is used, where $\sigma(\cdot)$ is the sigmoid function.

Formally, the relationship between the mutual information and the surrogate classification problem is given in terms of the Donsker-Varadhan representation (DV) [45] of the Kullback–Leibler divergence, i.e.,

$$\mathcal{I}(\mathbf{b}, \mathbf{c}) = KL(\mathbb{J}|\mathbb{M}) \geq \sup_f \mathbb{E}_\mathbb{J}[f] - \log(\mathbb{E}_\mathbb{M}[e^f]), \tag{6.6}$$

where f is the class of functions for which the expectation exists. MINE-like estimators [11] restrict the function class f to the class of neural network functions f_θ using the universal approximation theorem. Experiments showed that the numerically more stable cross entropy could be used as a loose lower-bound instead (e.g., [70]):

$$\mathcal{I}(\mathbf{b}, \mathbf{c}) \geq \mathcal{I}_{f_\theta}(\mathbf{b}, \mathbf{c}) = -CE(f_\theta(\tilde{\mathbf{X}}), \mathbf{y}) + \text{const.} \tag{6.7}$$

We describe the generation of the registered and non-registered sets of patches in Sec. 6.3.1. The architecture of the neural network f_θ for classification is introduced in Sec. 6.3.1.

Projection

We follow [50] and transform the point cloud $\mathbf{L}_i$ by the extrinsic transformation $\mathbf{E} \in \text{SE}(3)$. Here, $\mathbf{E}$ represents an element from the special Euclidean group and is composed of a rotation matrix $\mathbf{R} \in \text{SO}(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$. Afterward, the transformed point cloud is projected by the camera matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$. The projected coordinates are given by

$$\mathbf{U}_i \sim \widehat{\mathbf{U}}_i = \mathbf{PEL}_i, \tag{6.8}$$

where $\mathbf{L}_i \in \mathbb{R}^{4\times K_i}$ is the 3d input point cloud in homogeneous coordinates, $\widehat{\mathbf{U}}_i \in \mathbb{R}^{3\times K_i}$ is the projected 2d point cloud in homogeneous coordinates, $\mathbf{U}_i \in \mathbb{R}^{2\times K_i}$ are image coordinates normalized by dividing by the z-coordinate, and K_i is the total number of points in the point cloud. Next, we compute a depth image $\mathbf{B}_i \in \mathbb{R}^{1\times H\times W}$ of the point cloud with the same spatial size as the RGB camera image $\mathbf{B}_i \in \mathbb{R}^{3\times H\times W}$. $\mathbf{B}_i$ is rendered by setting the Euclidean distance of the transformed Lidar point to the camera origin $d(\mathbf{EL}_i^j, \mathbf{0})$ as pixel intensity, given by

$$\mathbf{B}_i(x, y) = \begin{cases} d(\mathbf{EL}_i^j)/s & (x, y) \in \tilde{\mathbf{U}}_i \\ 0 & \text{otherwise} \end{cases}, \tag{6.9}$$

where $\tilde{\mathbf{U}}_\mathbf{i} \in \mathbb{Z}^{2\times K_i}$ is computed by rounding all values of $\mathbf{U}_i$ to the next integer pixel position, and $s = \text{std}(\{d(\mathbf{EL}_i^j)\}_j^{K_i})$ is the standard deviation of all euclidean distances of the transformed Lidar points in the i-th point cloud. Here, we use the Euclidean distance instead of Lidar intensity values because the needed shape features, such as sharp edges, are better retained, and the distances are less noisy [174]. The scaling factor is needed because we expect close and very distant Lidar points across different frames.

Differentiable patch sampling

One significant difficulty in our approach is computing the gradient w.r.t. $\mathbf{E}$ as the patch sampling procedure is usually not differentiable. Hence, a differentiable patch sampling function $\pi(\cdot)$ is required for generating the $2M$ patch tuples

$$\tilde{\mathbf{X}} = \pi(\mathbf{L}, \mathbf{C}; \mathbf{E}), \tag{6.10}$$

with corresponding labels

$$\mathbf{y} = \{0^M, 1^M\} \tag{6.11}$$

for the surrogate classification problem. For compatibility with Eq. 6.7, we define the patch sample function $\pi(\mathbf{L}, \mathbf{C}; \mathbf{E})$ on the entire dataset. Next, we crop square patches with size $S \times S$ from the two images $\mathbf{B}_i$ and $\mathbf{C}_i$ (see green dots on black background in Fig. 6.6). To back-propagate the gradient of the neural network f_θ to the calibration parameters $\mathbf{E}$, the cropping procedure needs to be differentiable. To this end, we use a localized separable interpolation kernel [82]

$$k(x) = e^{-(x-\frac{S}{2})^2}, \tag{6.12}$$

with offset S for centering the patches, and choose the projected Lidar points $\mathbf{U}_i = \{\mathbf{u}_i^j\}_{j=1}^{K_i}$ as patch centers (cf. Eq. 6.12). Note that interpolation is mandatory at this stage as the projected Lidar points fall in-between pixels. Hence, the center pixel and the pixels of the extracted patches are interpolated subpixels. The patch tuples $\{(\mathbf{b}_i^j, \mathbf{c}_i^j)\}_{j=1}^{K_i}$ corresponding to $\mathbf{U}_i$ are given by

$$\mathbf{b}_i^j(x, y) = \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{B}_i(h, w)\ k(\mathbf{u}_i^j(y) - h)\ k(\mathbf{u}_i^j(x) - w) \tag{6.13}$$

and

$$\mathbf{c}_i^j(x, y) = \sum_{h=1}^{H} \sum_{w=1}^{W} \mathbf{B}_i(h, w)\ k(\mathbf{u}_i^j(y) - h)\ k(\mathbf{u}_i^j(x) - w). \tag{6.14}$$

Because the interpolation kernel is differentiable, the gradient can be back-propagated through the patch cropping procedure by applying the chain rule. Note that the patch ensemble depends on the projected Lidar points $\mathbf{U}_i$ and, therefore, on the current calibration configuration during optimization. However, as the ensemble is kept constant during the gradient update, the differentiability of the optimization with respect to $\mathbf{E}$ is retained.

Sub-sampling

As the Lidar point clouds can contain thousands of points, there are also thousands of patch tuples that can be computationally prohibitive. To mitigate, we sub-sample a M-sized subset of the available K_i patch tuples. Therefore, we define a set of equidistant 2d grid points $\mathcal{G}$ with distance M_g in the $H \times W$ plane (cf. Fig. 6.4) and then select patch tuples that are nearest neighbors of the grid points. As a helpful side-effect, the 2d grid equalizes the sampling density of the projected points, as many spots are hugely oversampled, such as walls or the floor. Unfortunately, patch pairs may be the nearest neighbor of multiple grid points, which causes interfering imbalances. To eliminate such duplicate patch pairs, we iterate over the K_i patch tuples and test for each tuple $(\mathbf{b}_i^j, \mathbf{c}_i^j)$, whether its corresponding projected Lidar point $\mathbf{u}_i^j$ is the nearest neighbor of a grid point $\mathcal{G}_m$. Formally, the set of registered patches is given by

$$\mathbb{J} \sim \widehat{\mathbb{J}} = \{(\mathbf{b}_i^j, \mathbf{c}_i^j) \mid \exists \mathcal{G}_m :\ j = \arg\min_k ||\mathbf{u}_i^k - \mathcal{G}_m||_2\}_{j=1}^{K_i}. \tag{6.15}$$

The equally sized set of randomized patch pairs

$$\mathbb{M} \sim \widehat{\mathbb{M}} = \{(\mathbf{b}_i^j, \mathbf{c}_i^*)\}_{j=1}^{K_i} \tag{6.16}$$

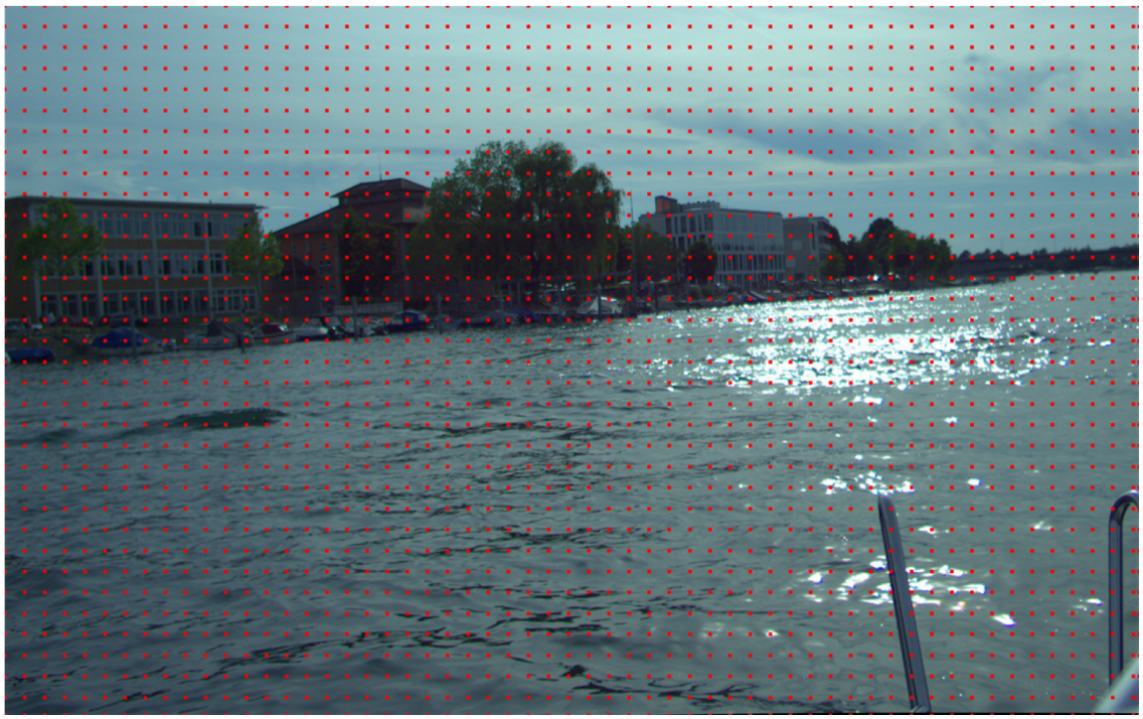


Figure 6.4: Visualization of the uniform sampling grid $\mathcal{G}$ for sub-sampling the projected Lidar points. The distance M_g between the grid points is given in pixels.

is generated by picking the camera patches $\mathbf{c}^*$ randomly. Finally, the differentiable patch sampling is given by

$$\pi(\mathbf{L}, \mathbf{C}; \mathbf{E}) = \{\mathbb{J}, \mathbb{M}\}. \tag{6.17}$$

Again, as the sampled set $\pi(\cdot)$ is kept constant during the gradient update, differentiability with respect to $\mathbf{E}$ is retained.

MINE architecture

The used modified MINE estimator requires a parametric function f_θ for classification and the subsequent mutual information estimation (cf. Eq. 6.7). Therefore, we define a convolutional neural network (CNN) f_θ with two input heads. One input head is for the Lidar patches $\mathbf{b}_j$, and a second for the camera patches $\mathbf{c}_j$. The schematic architecture of the neural network is shown in Fig. 6.2b. The CNN blocks consist of a standard batch normalization layer, followed by six convolutional layers with average pooling and ReLU-activation functions. We use 32 features in every CNN layer and a kernel size of five. The input layers differ in the number of channels for Lidar and image patches. The correlation layer does not have parameters and correlates the inputs by flattening and computing the dot product between the 32 feature maps of the two input paths by computing

$$\text{correlation}_z(\mathbf{f}_z, \mathbf{g}_z) = \langle \mathbf{f}_z \cdot \mathbf{g}_z \rangle \tag{6.18}$$

per feature map, where $\mathbf{f}_z \in \mathbb{R}^{uv}$ is the z^{th} feature map of the Lidar path, $\mathbf{g}_z \in \mathbb{R}^{uv}$ is the z^{th} feature map of the camera path, and uv is the dimension of the flattened feature map f_z. The fully connected block contains two fully connected (FC) layers with ReLU-activation and 32 features. The entire network has 259 205 parameters.

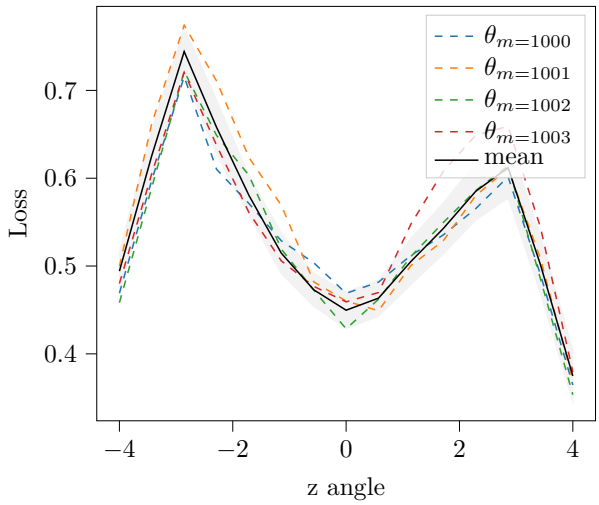
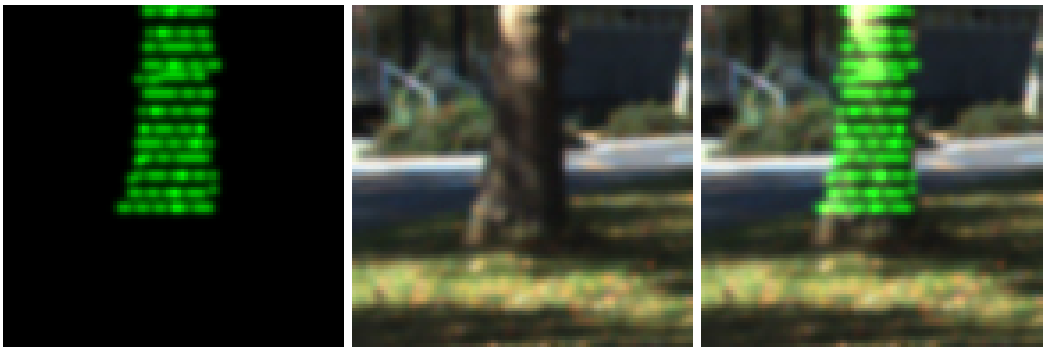


Figure 6.5: Visualizing the loss function $-\mathcal{I}_{f_\theta}(\pi(\mathbf{L}, \mathbf{C}; \mathbf{E}_{dist}))$ for small Euler angle distortions of $\mathbf{E}_{dist}(0, 0, \epsilon, \mathbf{0})$, with $\epsilon \in [-4, 4]$. The dotted lines show the loss function at different iterations θ_m of gradient descent. Notice the wrong local minimum (yellow).



(a) Registered tuple $(\mathbf{b}_i^j, \mathbf{c}_i^j) \in \mathbb{J}$

(b) Random non-registered tuple $(\mathbf{b}_i^j, \mathbf{c}_i^*) \in \mathbb{M}$

Figure 6.6: Example of registered and non-registered patch tuples. The left and middle columns show a patch from $\mathbf{B}$ and $\mathbf{C}$, respectively. For illustration purposes, we also show the combined patches in the right column.

Estimating $\mathcal{I}(\mathbf{Lidar}, \mathbf{Camera})$

The pre-training step is required to maximize the lower bound of the mutual information, such that it can be used as an optimization criterion in the registration phase. The pre-training is done using the registered training examples and optimizing the trainable parameters θ of the CNN f_θ as defined in Eq. 6.7. Plugging all steps of the pipeline (cf. Fig. 6.2) together results in the optimization problem

$$\theta^* = \arg\max_\theta \mathcal{I}_{f_\theta}(\pi(\mathbf{L}_{train}, \mathbf{C}_{train}; \mathbf{E}_{true}), \mathbf{y}), \tag{6.19}$$

where $\pi(\cdot)$ is the patch sampling function that transforms an input pair into tuples of aligned and non-aligned local patches (cf. Sec. 6.3.1). Again, the optimization is solved by solving the surrogate classification problem. The resulting θ^* is used for initializing the second step.

6.3.2 Optimizing registration

The following section describes the optimization procedure for estimating the unknown registration matrix $\mathbf{E}$ using gradient ascent.

Optimizing E

The second step is the actual registration of an unregistered test set $\mathcal{D}_{test} = (\mathbf{L}_{test}, \mathbf{C}_{test}, \mathbf{E}_{init})$ with N input frames with respect to the extrinsic calibration parameters $\mathbf{E}_{mi}$. The initialization is given by $\mathbf{E} = \mathbf{E}_{init}$. Because of the pre-trained $\mathcal{I}_{f_\theta}$ estimator, the propagated gradients are stable and can be directly used for improving the initial estimate of $\mathbf{E}$

$$\mathbf{E}_{mi} = \arg\max_{\mathbf{E}} \mathcal{I}_{f^*_\theta}(\pi(\mathbf{L}_{test}, \mathbf{C}_{test}; \mathbf{E})). \tag{6.20}$$

During experiments, we noticed that the optimization sometimes gets stuck in a local minimum near the global optimum. Therefore, we add the CNN parameters θ to the optimization

$$\mathbf{E}_{mi}, \theta' = \arg\max_{\mathbf{E},\theta} \mathcal{I}_{f_\theta}(\pi(\mathbf{L}_{test}, \mathbf{C}_{test}; \mathbf{E})) \tag{6.21}$$

once it reaches its first plateau. We show the loss function for a single rotation angle at different iteration steps θ_m in (cf. Fig. 6.5). Alg. 3 shows the entire algorithm in pseudocode.

Gradient ascent implementation

We do not optimize in full-batch, but use a single frame at a time, i.e., stochastic gradient descent (SGD) [18]. This means we compute the back-propagated gradients

$$\nabla_{\mathbf{E}} \mathcal{I} = \nabla_{\mathbf{E}} \mathcal{I}_{f_{\theta^*_w}}(\pi(\mathbf{L}^i_{test}, \mathbf{C}^i_{test}; \mathbf{E})), \tag{6.22}$$

with

$$i \sim \mathcal{U}(0, N) \tag{6.23}$$

based on a single randomly sampled pair $(\mathbf{L}_i, \mathbf{C}_i)$ instead of the full data set. Next, we split the parametrization of $\mathbf{E}$ into their affine components $\mathbf{R}$ and $\mathbf{t}$ and treat the components separately. While optimizing $\mathbf{t}$ is trivial, optimizing the rotation matrix $\mathbf{R}$ is non-trivial as it is constrained to be orthonormal. We optimize inside the Lie group SO(3) and use the *geodesic flow* method as proposed in [138] for computing the derivatives. However, instead of solving the gradient w.r.t. $\mathbf{R}$ analytically, we back-propagate it by the chain rule.

Algorithm 3 LMI algorithm with *train*() and *register*() methods

$f_\theta \leftarrow$ MINE architecture with trainable parameters θ
$\tau \leftarrow$ Maximum number of epochs
$\eta \leftarrow$ Learning rate
$M \leftarrow$ Number of patch tuples
$\mathbf{y} \leftarrow \{0^M, 1^M\}$

procedure TRAIN($\mathbf{L}_{train}, \mathbf{C}_{train}, \mathbf{E}_{true}$)
    $\theta \leftarrow \theta_{init}$
    while not converged(max_epochs) **do**
        $\tilde{\mathbf{X}} = \pi(\mathbf{L}_{train}, \mathbf{C}_{train}; \mathbf{E}_{true})$
        $\theta = \theta - \eta \nabla_\theta CE(f_\theta(\tilde{\mathbf{X}}), \mathbf{y}))$
    $\theta^* \leftarrow \theta$
    return θ^* ▷ The trained MINE parameters

procedure REGISTER($\mathbf{L}_{test}, \mathbf{C}_{test}, \mathbf{E}_{init}, \theta^*$)
    $\mathbf{R}, \mathbf{t} \leftarrow \mathbf{E}_{init}$
    while not converged(τ) **do**
        $\tilde{\mathbf{X}} = \pi(\mathbf{L}_{test}, \mathbf{C}_{test}; \mathbf{E}_{mi})$
        $\nabla_{\mathbf{R}}\mathcal{I}, \nabla_{\mathbf{t}}\mathcal{I} \leftarrow \nabla_{\mathbf{E}} CE(f_{\theta^*}(\tilde{\mathbf{X}}), \mathbf{y}))$
        $\nabla_{\mathbf{\Theta}}\mathcal{I} = (\nabla_{\mathbf{R}}\mathcal{I})^T \mathbf{R} - \mathbf{R}^T (\nabla_{\mathbf{R}}\mathcal{I})$
        $\mathbf{R}^T = \exp(\eta\, \mathbf{\Theta_r})\, \mathbf{R}^T$
        $\mathbf{t} = \mathbf{t} - \eta \nabla_{\mathbf{t}}\mathcal{I}$
    $\mathbf{E}_{mi} \leftarrow (\mathbf{R}, \mathbf{t})$
    return $\mathbf{E}_{mi}$ ▷ The optimized calibration parameters

Technically, we use the orthogonal layer developed for Lie-Adam [65] as introduced in Chapter 4 and repeat its notation shortly for completeness. The back-propagated gradient of the loss function $\nabla_{\mathbf{R}}\mathcal{I}$ is a skew-symmetric matrix $\mathbf{\Theta}$, that represents an infinitesimal rotation and hence an element of the corresponding Lie algebra $\mathfrak{so}(3)$. The Lie algebra consists of all skew-symmetric matrices and becomes the space for gradient ascent. Every skew-symmetric matrix $\mathbf{\Theta}$ can be uniquely parameterized by a 3d vector $\mathbf{r}$ giving rise to a vector space. Further, every skew-symmetric matrix $\mathbf{\Theta}$ can be related to an orthogonal matrix $\mathbf{R}$ by

$$\mathbf{R} = \exp(\mathbf{\Theta}), \tag{6.24}$$

where $\exp(\cdot)$ is the matrix exponential. To compute a valid gradient step beyond the neighborhood of $\mathbf{R}$, the gradient direction must be expressed by the Lie bracket [138]. The relation between the two gradient expressions is given by the commutator

$$\nabla_{\mathbf{\Theta}}\mathcal{I} = (\nabla_{\mathbf{R}}\mathcal{I})^T\,\mathbf{R} - \mathbf{R}^T\,(\nabla_{\mathbf{R}}\mathcal{I})\,. \tag{6.25}$$

We compute the corresponding parameter vector $\mathbf{r}$ by taking the upper triangular matrix of $\nabla_{\mathbf{\Theta}}\mathcal{I}$. The geodesic flow update rule is then given by

$$\mathbf{R}_{m+1}^T = \exp(\eta\,\mathbf{\Theta}_{\mathbf{r}_m})\,\mathbf{R}_m^T, \tag{6.26}$$

with step size η and the skew-symmetric matrix $\mathbf{\Theta}_{\mathbf{r}_m}$ parameterized by $\mathbf{r}_m$ at the m-th iteration step. Unfortunately, rotation matrices above two dimensions are not commutative. Hence we cannot make additive steps of ascent in $\mathfrak{so}(3)$ and need to map between the Lie algebra and the manifold in every iteration by computing the matrix exponential in Eq. 6.24. We also use ADAM optimizer with $lr = 10^{-3}$ for controlling the step size (cf. Chapter 4) and run the optimization for 4000 iterations until convergence. The learning rate is reduced by 0.1 when reaching a plateau.

6.4 Experiments

For the experiments, we use the following techniques as baselines: LCCNet [112], RegNet [159], and CMRNet [30], which are neural network methods.[2] We also compare classic methods based on normalized mutual information and gradient information (gNMI [130])[3], respectively particle swarm optimization (pNMI [174]). For measuring performance, we follow [30] and also measure the translation error

$$\mathbf{t}_{\Delta}(\mathbf{t}_{true}, \mathbf{t}_{est}) = ||\mathbf{t}_{true} - \mathbf{t}_{est}||_2 \tag{6.27}$$

and the rotation error by using the quaternion angle

$$\mathbf{R}_{\Delta}(\mathbf{Q}_{true}, \mathbf{Q}_{est}) = 2 * \text{atan2}(||\,\mathbf{Q}_{\Delta}^{Im}\,||_2, \mathbf{Q}_{\Delta}^{Re}), \tag{6.28}$$

with

$$\mathbf{Q}_{\Delta} = \mathbf{Q}_{\mathbf{R}_{true}} * \mathbf{Q}_{\mathbf{R}_{est}}^{-1}, \tag{6.29}$$

during the optimization procedure. The last formula first computes the difference rotation $\mathbf{Q}_{\Delta}$ and then measures the shortest angle to identity. atan2 is the standard 2-argument arctangent extension, which distinguishes the four quadrants in the Euclidean plane and is available through numerous math libraries.

[2] https://github.com/IIPCVLAB/LCCNet

[3] https://github.com/xmba15/automatic_lidar_camera_calibration

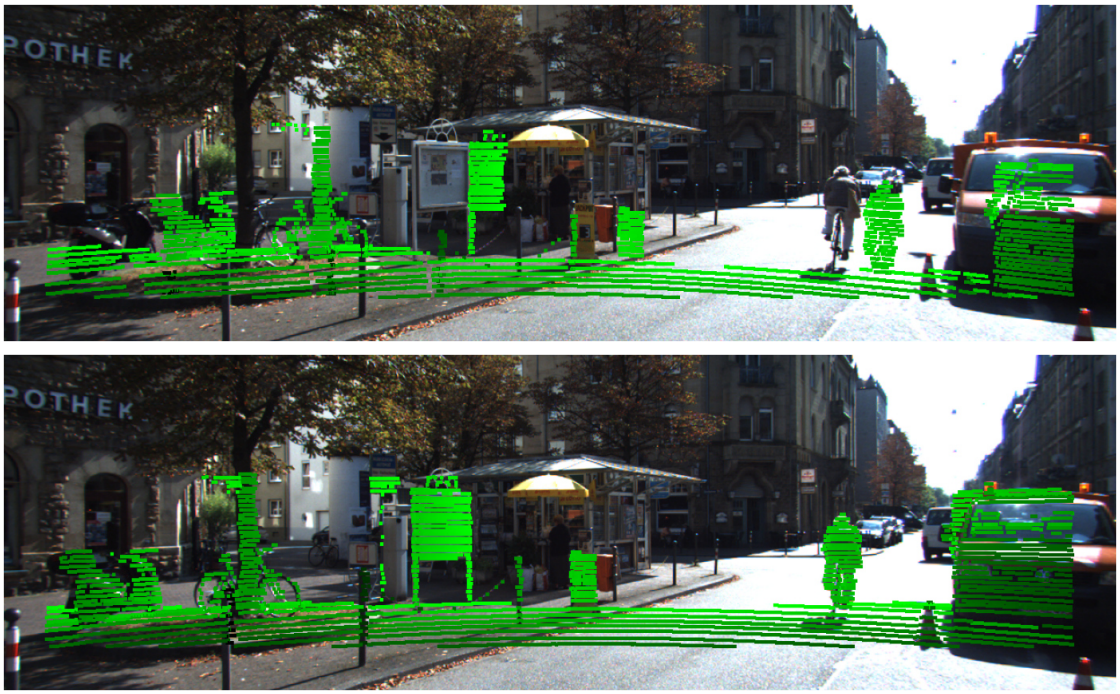


Figure 6.7: Visual registration results on the KITTI dataset. Note the accuracy at the bollard on the left-hand side of the image.

6.4.1 Pre-training

For pre-training, we use the KITTI dataset [50], which consists of 20 sequences with varying lengths. In total, there are 7125 frames available. The images are stored as 375×1242 RGB. The Lidar scans are 360° Velodyne-64 scans. We take the first ten sequences as training data and pre-train the mutual information estimator $\mathcal{I}_\theta(\mathbf{A}, \mathbf{B})$ until convergence by using left-out 10 % validation data, and ADAM optimizer with $\eta = 10^{-3}$ for all trainable parameters. We use $M_g = 5$ for sampling patches and a mini-batch size $M = 24$ for training the estimator CNN network. The cropped patch size S is set fixed to 96×96 pixels, which we found to be a good trade-off between runtime, generalization, and accuracy. In Tab. 6.2, we also report results for other patch sizes, i.e., $\{68, 96, 128\}$.

6.4.2 Registration performance

We take the left-out ten sequences for measuring performance and test the registration performance by adding minor random distortions to the available ground truth calibration parameters $(\mathbf{Rt})_{true}$. This is implemented by uniformly sampling random translations and rotations from the specified interval $\mathcal{U}(\cdot, \cdot)$. Because of the relatively large translation miscalibrations in the experiment, we found it helpful to increase the learning rate for parameter $\mathbf{t}$ to $\eta_{\mathbf{t}} = 10^{-2}$. Tab. 6.1 and Tab. 6.2 summarize the results. Fig. 6.7 show the registration results for a representative test set example.

While the gNMI method fails on the KITTI dataset, mainly because of the unreliable Lidar intensity values, LCCNet achieved a remarkable rotation error of 0.16° and a translation error of 0.02m. Note that the significant variance in Tab. 6.2 is mainly due to the per-frame calibration estimation. With at least 200 frames available, the LMI method is on par with the baselines. When more than

Error R	**Error t**	**Performance**	
Euler [deg]	xyz [m]	$\mathbf{R}_\Delta$ [deg]	$\mathbf{t}_\Delta$ [m]
$\mathcal{U}(-2, 2)$	$\mathcal{U}(-0.30, 0.30)$	$\mathbf{0.11 \pm 0.05}$	$\mathbf{0.02 \pm 0.01}$
$\mathcal{U}(-2, 2)$	$\mathcal{U}(-0.60, 0.60)$	0.13 ± 0.07	0.03 ± 0.01
$\mathcal{U}(-5, 5)$	$\mathcal{U}(-0.30, 0.30)$	0.13 ± 0.06	0.03 ± 0.01
$\mathcal{U}(-5, 5)$	$\mathcal{U}(-0.60, 0.60)$	0.14 ± 0.06	0.03 ± 0.01
$\mathcal{U}(-7, 7)$	$\mathcal{U}(-0.30, 0.30)$	0.12 ± 0.07	0.02 ± 0.01
$\mathcal{U}(-7, 7)$	$\mathcal{U}(-0.60, 0.60)$	0.89 ± 1.13	0.41 ± 0.58
$\mathcal{U}(-8, 8)$	$\mathcal{U}(-0.30, 0.30)$	0.13 ± 0.07	0.02 ± 0.01
$\mathcal{U}(-8, 8)$	$\mathcal{U}(-0.60, 0.60)$	3.41 ± 4.14	0.31 ± 0.42
$\mathcal{U}(-9, 9)$	$\mathcal{U}(-0.30, 0.30)$	1.12 ± 2.64	0.07 ± 0.13

Table 6.1: Registration results on the KITTI [50] challenge averaged over five uniform random sampled distortions in the specified interval, i.e., $r_x, r_y, r_z \sim \mathcal{U}(\cdot, \cdot)$ and $x, y, z \sim \mathcal{U}(\cdot, \cdot)$.

500 frames are available, the baselines could be surpassed. However, most of their experiments included large translation distortions of up to 2m, making their method more suited for rough or initial registration. In contrast, we are more focused on fine registration (see Tab. 6.1). We show the progress of a single registration optimization in Fig. 6.8.

Figure 6.8: Visualizing the optimization for 400 iterations, i.e., gradient updates.

Method	**Performance**	
	$\mathbf{R}_\Delta$ [deg]	$\mathbf{t}_\Delta$ [m]
gNMI[130]	8.17 ± 5.69	0.08 ± 0.05
pNMI[174]	1.08 ± 0.51	0.03 ± 0.01
CMRNet[30]	1.07 ± 0.77	0.33 ± 0.11
RegNet[159]	0.28 ± 0.20	0.06 ± 0.10
LCCNet[112]	0.16 ± 0.47	0.02 ± 0.02
LMI($N = 533, S = 128$)	0.08 ± 0.04	0.04 ± 0.01
LMI($N = 533, S = 96$)	$\mathbf{0.14 \pm 0.02}$	$\mathbf{0.02 \pm 0.01}$
LMI($N = 533, S = 68$)	0.13 ± 0.01	0.03 ± 0.01
LMI($N = 235, S = 128$)	0.16 ± 0.01	0.13 ± 0.00
LMI($N = 235, S = 96$)	0.23 ± 0.12	0.01 ± 0.01
LMI($N = 235, S = 68$)	0.39 ± 0.06	0.10 ± 0.01
LMI($N = 147, S = 128$)	0.23 ± 0.04	0.14 ± 0.01
LMI($N = 147, S = 96$)	1.80 ± 1.03	0.43 ± 0.35
LMI($N = 147, S = 68$)	0.55 ± 0.26	0.14 ± 0.02
LMI($N = 58, S = 128$)	0.50 ± 0.44	0.11 ± 0.04
LMI($N = 58, S = 96$)	1.25 ± 0.15	0.14 ± 0.03
LMI($N = 58, S = 68$)	1.33 ± 0.13	0.16 ± 0.02

Table 6.2: Registration results on the KITTI [50] challenge with a random angle distortion $r_x, r_y, r_z \sim \mathcal{U}(-2, 2)$ and random translation offset $x, y, z \sim \mathcal{U}(-0.6, 0.6)$. We used the first sequences of the test dataset corresponding to sequences 10 to 13 in the original KITTI dataset. Results are averaged over five runs. To evaluate the dependence on the number of test examples, we also report results for our Local Mutual Information method (LMI) for different numbers of available frames N and patch sizes S.

6.4.3 Generalization performance

For testing the generalization performance, we use an unseen scenario of a docking maneuver of a motor boat consisting of 472 1919×1199 sized RGB images and Velodyne-128 Lidar scans. The mutual information estimator is again pre-trained on KITTI. The initial rotation estimate is $\mathbf{R} = \mathbf{I}$ and the known true solution is roughly 2° off (see Tab. 6.3). We distorted the true translation by a small ϵ, as translation offset can be measured quite accurately in practice. In the given real-world scenario, the translation miscalibration is relatively small compared to the angular offset. Therefore, we found it helpful to decrease the learning rate for parameter $\mathbf{t}$ to $\eta_\mathbf{t} = 10^{-4}$. In Fig. 6.1, we show an example demonstrating the generalization capabilities. Surprisingly most methods perform better than the learning approach LCCNet in the generalization scenario, indicating that the LCCNet learned KITTI-specific features that do not generalize well to the docking scene. The LMI optimization of the unregistered test set took 3min on standard hardware using an Intel i7-7600 and an NVIDIA 1080Ti. Note that the runtime is generally slower compared to feedforward approaches, such as LCCNet, and more similar to the classic optimization approaches.

Method	**Error t** xyz [m]	**Performance** $\mathbf{R}_\Delta$ [deg]	$\mathbf{t}_\Delta$ [m]
LCCNet	$\mathcal{U}(-0.01, 0.01)$	1.80 ± 0.52 (1.84)	0.96 ± 0.27 (0.99)
	$\mathcal{U}(-0.02, 0.02)$	1.79 ± 0.52 (1.84)	4.80 ± 1.38 (4.91)
gNMI	$\mathcal{U}(-0.01, 0.01)$	1.17 ± 0.73 (0.67)	0.07 ± 0.03 (0.08)
	$\mathcal{U}(-0.02, 0.02)$	1.16 ± 0.39 (1.32)	0.06 ± 0.01 (0.05)
pNMI	$\mathcal{U}(-0.01, 0.01)$	0.96 ± 0.60 (0.74)	0.02 ± 0.01 (0.02)
	$\mathcal{U}(-0.02, 0.02)$	0.77 ± 0.48 (0.73)	0.02 ± 0.01 (0.02)
LMI	$\mathcal{U}(-0.01, 0.01)$	$\mathbf{0.62 \pm 0.60}$ (**0.28**)	$\mathbf{0.01 \pm 0.00}$ (**0.01**)
	$\mathcal{U}(-0.02, 0.02)$	0.72 ± 0.56 (0.67)	0.01 ± 0.01 (0.02)

Table 6.3: Measuring generalization performance (mean±std (median)) for the maritime scenario with $R_{init} = \mathbf{I}$. Results are averaged over five runs with random measurement distortions. The manually found true solution is $r_x, r_y, r_z = (0.49, 1.09, -0.10)$, and $x, y, z = (0.77, 0.07, -0.11)$.

6.5 Conclusion

In this chapter, we presented a hybrid algorithm for targetless registration of a Lidar-camera system using mutual information maximization on local patches. The algorithm is suitable for situations where calibration is required (e.g., caused by weather conditions or mechanical changes), but no labeled data is available. Technically, our method optimizes the alignment of local shape patterns between the camera and projected Lidar points over multiple frames. By using local patterns, the algorithm is able to generalize to completely new Lidar-camera configurations.

The main limitation of the algorithm comes from the chosen receptive field, i.e., the patch size, which constrains the maximum angular miscalibration to about $\pm 7^\circ$ and the translation miscalibration to approximately 0.5m in our experiments. As soon as the projected Lidar points lie outside of the corresponding RGB image patches, no valid error signal can be propagated. Hence, the real-world region of convergence heavily depends on the distance of surrounding objects, and the patch size needs to be increased for large distance problems. However, too large patch sizes would lead to decreased generalization performance because large patches contain dataset-specific patterns that do not generalize to new configurations. A second limitation is the number of available frames and, more important, prominent objects in the given sequence. When there are too few shape patterns (e.g., edges or occlusions) available for alignment, the method becomes unstable, similar to the existing methods. The output of

By using a hybrid optimization scheme for calibrating temporal synchronized sensors, several extensions are possible for future work. One direction is optimizing other parameters like the lens distortion, another is integrating other sensor modalities like radar or infrared.

CHAPTER 7

Concluding remarks and outlook

After the fundamentals chapter, the following chapters introduced the proposed algorithmic methods, namely (1) Deep μshift [67], (2) Lie-Adam [65], (3) R-NSCL [69], and (4) LMI [64], targeting a deep learning-based system for industrial defect detection. The following conclusion chapter summarizes the main contributions and gives a short outlook on identified potential future research directions.

7.1 Summary

The research presented in this dissertation built upon the fields of unsupervised learning [89], semi-supervised learning [200, 199], transfer learning [192], and component analysis in signal processing, which has a significant history in optical inspection [97]. In summary, this dissertation addressed the challenges of defect detection in an industrial manufacturing process where only weakly labeled data and limited supervision are available. The data stream captured from an optical system was categorized into four partitions: observed non-defects (reference), unobserved non-defects (variants), observed defects (outliers), and unobserved defects (novelties). The goal was to develop a data-efficient method that could effectively learn a data representation with suitable invariances against variants while having robust detection capabilities towards unknown defects.

Although current state-of-the-art models for defect detection employ deep convolutional autoencoders or modern autoregressive methods, they often fall short in performance compared to more straightforward methods like One-Class SVM [1, 21]. To overcome these limitations, the proposed approach leveraged information theory to analyze feature representations. Here, the entropy, or negentropy, of a representation is an exciting measure, as it sets an upper bound on the size of its typical set, thus influencing its chances of intersecting with unknown distributions. The hypothesis inspiring this thesis was that the representation with the smallest typical set would yield the best model for separating the data. In the context of real-world signals, independent component analysis (ICA) [35] is known to produce compact, low-entropy, or sparse features, making it a suitable starting point for this research. Based on that, this dissertation introduced an efficient novelty detection method and a novel hyperparameter selection method for finding the optimal patch size. Furthermore, a novel one-class incremental learning approach and a data fusion algorithm that both reduce the need for supervision in optical inspection scenarios were proposed:

Deep μshift [67] is a non-expensive algorithm for image novelty detection. For this algorithm, we combined the Hotelling T^2 test with a transferred pre-trained neural network, such as EfficientNet [173] or Vision Transformer [46], to induce a rich feature space. Unlike previous approaches, we used a regularized full-rank covariance matrix of the feature space instead of a compressed low-rank approximation. This maximized information and significantly improved performance for small datasets. Additionally, we generalized the developed patch ensemble approach to novelty localization and introduced a hyperparameter for controlling the expected spatial size of anomalies. Our experiments demonstrated that our approach achieved comparable results to state-of-the-art methods while being applicable to large-scale industrial inspection scenarios. Moreover, due to its simple architecture, our model has significantly faster prediction times than comparable methods, and we showed data efficiency by achieving 90 % AUC with only a few non-defective examples. It turned out that optimizing sparsity allows effectively tuning the patch size hyperparameter, which is crucial for the model to reach a detection performance similar to deep learning models.

This finding motivated another aspect of my research, where I focused on computing sparsity efficiently on a large scale. The requirement came from the fact that feature spaces in computer vision often have hundreds of dimensions, and thousands of unlabeled data examples must be processed. For this task, I analyzed independent component analysis (ICA) for large data matrices and highlighted the importance of orthogonal constraints. Besides, we emphasized the necessity of the pre-whitening requirement for stability. The introduced Lie-Adam algorithm [65] improves the geodesic flow update rule [138] using the ADAM optimizer and uses the efficient Caley approximation for computing the matrix exponential. We effectively managed the space complexity of the pipeline using mini-batches, resulting in an efficient and scalable approach. We showed its effectiveness by computing the first 484 independent components of the ImageNet dataset [42].

In the second part, higher-order features were incorporated to enhance detection capabilities incrementally by observing a limited number of labeled examples. In our investigation of the null-space learning framework (NSCL) [184] for one-class incremental learning (one-class-IL), I identified a problem where the framework overfits novel classes in the one-class setting, leading to a degradation in performance on previously learned classes. To address this issue, I proposed a regularization term that penalizes the scale of newly learned features using ReLU-regularization. Our regularized null-space learning framework (R-NSCL) [69] stabilized null-space training and prevented excessive growth of novel decision regions. We evaluated our methods in few-shot protocols using industrial datasets, explicitly focusing on defect characteristics in industrial defect detection scenarios.

Furthermore, the proposed system was extended using a novel information fusion algorithm to register multi-sensor systems consisting of Lidar and cameras. Here, we introduced a targetless information fusion method for point cloud and image data called LMI [64]. The algorithm's output is the extrinsic calibration matrix and, consequently, an image channel with range data (RGBD). Alternatively, computing a colored point cloud is possible. Our method relies on registered training data for pre-training but incorporates an optimization scheme for unregistered test data. Technically, we project the Lidar point cloud onto the common image plane and optimize the mutual information between local patches. This approach proved beneficial in scenarios where calibration is required due to changing environmental conditions or mechanical changes, but no targets are available for registration. Our method offers accurate solutions and improved generalization compared to existing approaches, making it suitable for online or re-calibration scenarios where labeled training data is unavailable.

Each work addressed specific challenges in their respective fields and presented novel solutions to improve performance, efficiency, or generalization capabilities. The main contributions of my work

were manifold. Firstly, applying mean-shift-based novelty detection using deep learning, explicitly targeting optical inspection tasks, improves over traditional methods in accuracy and efficiency. Secondly, ADAM optimization accelerated the geodesic flow update rule for independent component analysis (ICA). Thirdly, regularized one-class incremental learning (one-class-IL) mitigates overfitting in the null-space framework when incrementally learning novel defect concepts using a limited number of labeled examples. Finally, the fusion of multi-modal sensor data, here images and Lidar range data, through patchwise mutual information maximization shows promising generalization capabilities.

Overall, my research contributed to the field of defect detection, where only limited supervision is available, and showed that reducing needed supervision is possible. The proposed approaches demonstrated promising results and opened perspectives for further advancements in optical inspection and related domains.

7.2 Future work

While this dissertation contributed to the area of defect detection and its closely associated domains, there exist multiple directions for potential future research and enhancement. The following list summarizes possible directions for future work to improve optical inspection further under limited supervision:

Exploration of recent deep learning techniques: While the proposed approaches in this dissertation yielded promising results, there is still room for exploring more advanced deep learning techniques. Future research could investigate the effectiveness of state-of-the-art generative models, such as deep diffusion models [38] or vector-quantized autoencoders (VQ-VAE) [4], in the context of weakly labeled data and defect detection. Additionally, incorporating more sources of label information could be explored to enhance the quality of learned representations.

Incorporation of self-supervised learning: Self-supervised learning has gained significant attention in recent years, showing promise in leveraging unlabeled data to learn useful representations. Future work could explore integrating self-supervised learning techniques, such as contrastive learning [58] or predictive coding [61], to improve the quality and robustness of learned representations. This could mitigate reliance on weakly labeled data and provide more effective defect detection models.

Investigation of ensemble learning techniques: Ensemble learning has been shown to improve the performance and robustness of machine learning models by combining multiple individual models [98]. Future research could explore applying ensemble learning techniques in the context of defect detection with weakly labeled data. This could involve combining multiple one-class learning models, each trained on different subsets of the available data, to achieve better generalization and detection accuracy. Furthermore, such an approach adds uncertainty estimates to predictions, which is relevant to practical applications.

Real-time implementation and deployment: Beside research prototypes, future work could focus on the practical implementation and deployment of the proposed defect detection models in real-time industrial settings. This would involve addressing challenges related to computational efficiency, scalability, and integration with existing production systems. The performance of the models under real-world operational conditions and their impact on production efficiency could be evaluated.

Evaluation on more extensive and diverse datasets: The proposed approaches in this dissertation were evaluated on specific datasets related to optical industrial inspection. Future research could involve evaluating the models on more extensive and diverse datasets to assess their generalization capabilities. This could include datasets from different manufacturing domains or datasets with a broader range of defect types. Furthermore, benchmarking the proposed approaches against existing state-of-the-art methods on these more extensive and diverse datasets would provide a more comprehensive evaluation.

Integration of domain knowledge: Exploiting domain knowledge about the manufacturing process and the specific defects of interest can significantly enhance defect detection performance. Future work could focus on incorporating domain knowledge into the learning process through explicit feature engineering or the design of specialized loss functions that capture domain-specific constraints. This could lead to more interpretable and practical defect detection models.

In conclusion, there are several exciting opportunities for future research in the field of machine vision and computer vision in general. Exploring advanced deep learning techniques, incorporating self-supervised learning, investigating ensemble learning, integrating domain knowledge, and focusing on real-time implementations are just a few of the potential directions that may further reduce supervision requirements in industrial computer systems and enhance their practical applicability.

Bibliography

[1] Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Latent space autoregression for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[2] Pierre Ablin, Jean-François Cardoso, and Alexandre Gramfort. Faster ICA under orthogonal constraint. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[3] Pierre Ablin, Alexandre Gramfort, Jean-François Cardoso, and Francis Bach. Stochastic algorithms with descent guarantees for ICA. *Proceedings of Machine Learning Research*, 2019.

[4] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. *Advances in neural information processing systems*, 30, 2017.

[5] Dror Aiger and Hugues Talbot. The phase only transform for unsupervised surface defect detection. In *Emerging Topics In Computer Vision And Its Applications*. World Scientific, 2012.

[6] Awad H Al-Mohy and Nicholas J Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31(3), 2010.

[7] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1), 2015.

[8] Mazin Aouf and Laurence AF Park. Approximate document outlier detection using random spectral projection. In *Australasian Joint Conference on Artificial Intelligence*. Springer, 2012.

[9] André Araujo, Wade Norris, and Jack Sim. Computing receptive fields of convolutional neural networks. *Distill*, 2019. https://distill.pub/2019/computing-receptive-fields.

[10] Raman Arora, Andrew Cotter, Karen Livescu, and Nathan Srebro. Stochastic optimization for PCA and PLS. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2012.

[11] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. D. Hjelm. Mutual information neural estimation. In *35th International Conference on Machine Learning*, ICML 2018, vol. 2, 2018.

[12] Johan G Belinfante, Bernard Kolman, and Harvey A Smith. An introduction to Lie groups and Lie algebras, with applications. *SIAM Review*, 8(1), 1966.

[13] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6), 1995.

[14] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. MVTec AD - a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[15] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[16] Adriana Bodnarova, Mohammed Bennamoun, and KK Kubik. Defect detection in textile materials based on aspects of the hvs. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, 1998.

[17] Giacomo Boracchi, Diego Carrera, and Brendt Wohlberg. Novelty detection in images by sparse representations. In *2014 IEEE Symposium on Intelligent Embedded Systems (IES)*. IEEE, 2014.

[18] Léon Bottou. Online algorithms and stochastic approximations. *Online learning and neural networks*, 1998.

[19] Paula Branco, Luís Torgo, and Rita P Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM computing surveys (CSUR)*, 49(2), 2016.

[20] Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. In *European Conference on Computer Vision*. Springer, 2002.

[21] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000.

[22] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *ArXiv*, 2022.

[23] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks*, 106, 2018.

[24] Shengyu Cao, Simai He, Zhening Li, and Zhen Wang. Extreme ratio between spectral and frobenius norms of nonnegative tensors. *SIAM Journal on Matrix Analysis and Applications*, 44(2), 2023.

[25] J. . Cardoso. Infomax and maximum likelihood for blind source separation. *IEEE Signal Processing Letters*, 4(4), 1997.

[26] J-F Cardoso and Beate H Laheld. Equivariant adaptive source separation. *IEEE Transactions on signal processing*, 44(12), 1996.

[27] Jean-François Cardoso and Antoine Souloumiac. Blind beamforming for non-gaussian signals. In *IEE proceedings F (radar and signal processing)*, volume 140. IET, 1993.

[28] Juan Castorena, Ulugbek S. Kamilov, and Petros T. Boufounos. Autocalibration of Lidar and optical cameras via edge alignment. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016. doi: 10.1109/ICASSP.2016.7472200.

[29] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, 2018.

[30] Daniele Cattaneo, Matteo Vaghi, Augusto Luis Ballardini, Simone Fontana, Domenico G Sorrenti, and Wolfram Burgard. Cmrnet: Camera to Lidar-map registration. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019.

[31] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[32] Yongsheng Cheng, Jiang Zhu, and Xiaokang Lin. An enhanced incremental svd algorithm for change point detection in dynamic networks. *IEEE Access*, 2018.

[33] H. J. Chien, R. Klette, N. Schneider, and U. Franke. Visual odometry driven online calibration for monocular Lidar-camera systems. In *Proceedings - International Conference on Pattern Recognition*, jan, 2016. vol. 0. Institute of Electrical and Electronics Engineers Inc.

[34] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011.

[35] Pierre Comon. Independent component analysis, a new concept? *Signal processing*, 36(3), 1994.

[36] Jonathan Cook and Vikram Ramadas. When to consult precision-recall curves. *The Stata Journal*, 20(1), 2020.

[37] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[38] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[39] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1. Ieee, 2005.

[40] Yann Dauphin, Harm De Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in neural information processing systems*, 2015.

[41] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: a patch distribution modeling framework for anomaly detection and localization. In *International Conference on Pattern Recognition*. Springer, 2021.

[42] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[43] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *ICLR (Poster)*, 2016.

[44] Rémi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74, 2018.

[45] Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2), 1983.

[46] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[47] César Ferri, José Hernández-Orallo, and R Modroiu. An experimental comparison of performance measures for classification. *Pattern recognition letters*, 30(1), 2009.

[48] Simone Fiori. A theory for learning by weight flow on stiefel-grassman manifold. *Neural Computation*, 13(7), 2001.

[49] Simone Fiori and Yoshua Bengio. Quasi-geodesic neural learning algorithms over the orthogonal group: A tutorial. *Journal of Machine Learning Research*, 6(5), 2005.

[50] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11), 2013.

[51] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv:1803.07728*, 2018.

[52] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. *Advances in neural information processing systems*, 31, 2018.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[54] Sachin Goyal, Aditi Raghunathan, Moksh Jain, Harsha Vardhan Simhadri, and Prateek Jain. Drocc: Deep robust one-class classification. In *International Conference on Machine Learning*. PMLR, 2020.

[55] Alex Graves. Sequence transduction with recurrent neural networks. *ArXiv*, abs/1211.3711, 2012.

[56] Michael Grunwald, Matthias Hermann, Fabian Freiberg, Pascal Laube, and Matthias O Franz. Optical surface inspection: A novelty detection approach based on CNN-encoded texture features. In *SPIE Applications of Digital Image Processing*. SPIE, 2018.

[57] Michael Grunwald, Matthias Hermann, Fabian Freiberg, and Matthias O Franz. Biologically-vision-inspired vs. CNN texture representations in novelty detection. In *SPIE Applications of Machine Learning*, 2021.

[58] Markus Hafner, Maria Katsantoni, Tino Köster, James Marks, Joyita Mukherjee, Dorothee Staiger, Jernej Ule, and Mihaela Zavolan. Clip and complementary methods. *Nature Reviews Methods Primers*, 1(1), 2021.

[59] Frederic Hake, Matthias Hermann, Hamza Alkhatib, Christian Hesse, Karsten Holste, Georg Umlauf, Gaël Kermarrec, and Ingo Neumann. Damage detection for port infrastructure by means of machine-learning-algorithms. In *Proceedings of FIG Working Week*, 2020.

[60] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. *Technical Report No. 2009-05*, 2009.

[61] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[62] J Hérault and B ANS. Circuits neuronaux á synapses modifiables: Décodage de messages composites par apprentissage non supervisé [neuronal circuits with modifiable synapses: De-

coding composite messages by unsupervised learning]. *Comptes Rendus de l'Académie des Sciences*, 299, 1984.

[63] Jeanny Hérault, Christian Jutten, and Bernard Ans. Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé. In *10 Colloque sur le traitement du signal et des images, FRA, 1985*. GRETSI, Groupe d'Etudes du Traitement du Signal et des Images, 1985.

[64] M. Hermann, Dennis Grießer, Bernhard Gundel, Daniel Dold, Georg Umlauf, and M. O. Franz. Targetless Lidar-camera registration using patch-wise mutual information. In *25th International Conference Information Fusion (FUSION)*, 2022.

[65] M. Hermann, G. Umlauf, and M. Franz. Large-scale independent component analysis by speeding up Lie group techniques. In *47th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[66] M. Hermann, G. Umlauf, B. Goldlücke, and M.O. Franz. Fast and efficient image novelty detection based on mean-shifts. *21th International Conference on Image Analysis and Processing (ICIAP)*, 2022.

[67] M. Hermann, G. Umlauf, B. Goldlücke, and M.O. Franz. Image novelty detection based on mean-shift and typical set size. *Sensors — Unusual Behavior Detection Based on Machine Learning*, 2022.

[68] M. Hermann, Georg Umlauf, and M. O. Franz. Fast and memory-efficient independent component analysis using Lie group techniques. In *10th International Conference on Curves and Surfaces*, 2022.

[69] M. Hermann, G. Umlauf, B. Goldlücke, and M. O. Franz. Incremental one-class learning using regularized null-space training for industrial defect detection. *Submitted to ICMV*, 2023.

[70] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2018.

[71] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9), 2021.

[72] Harold Hotelling. The generalization of student's ratio. In *Breakthroughs in statistics*. Springer, 1992.

[73] Shell Xu Hu, Da Li, Jan Stühmer, Minyoung Kim, and Timothy M Hospedales. Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2022.

[74] Aapo Hyvarinen. One-unit contrast functions for independent component analysis: A statistical analysis. In *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*. IEEE, 1997.

[75] Aapo Hyvärinen. Independent component analysis: recent advances. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984), 2013.

[76] Aapo Hyvarinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *Advances in Neural Information Processing Systems*, 29, 2016.

[77] Aapo Hyvarinen and Hiroshi Morioka. Nonlinear ICA of temporally dependent stationary sources. In *Artificial Intelligence and Statistics*. PMLR, 2017.

[78] Aapo Hyvärinen and Erkki Oja. A fast fixed-point algorithm for independent component analysis. *Neural computation*, 9(7), 1997.

[79] Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural networks*, 12(3), 1999.

[80] Aapo Hyvärinen, Jarmo Hurri, and Patrick O Hoyer. *Natural image statistics: A probabilistic approach to early computational vision.*, volume 39. Springer Science & Business Media, 2009.

[81] Aapo Hyvarinen, Hiroaki Sasaki, and Richard Turner. Nonlinear ICA using auxiliary variables and generalized contrastive learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019.

[82] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. *Advances in Neural Information Processing Systems*, 2015, January 2015.

[83] Dinesh Jayaraman and Kristen Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[84] Peng Jiang, Philip Osteen, and Srikanth Saripalli. Calibrating Lidar and camera using semantic mutual information. *arXiv:2104.12023*, 2021.

[85] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873), 2021.

[86] Juha Karhunen, Erkki Oja, Liuyue Wang, Ricardo Vigario, and Jyrki Joutsensalo. A class of neural networks for independent component analysis. *IEEE Transactions on neural networks*, 8(3), 1997.

[87] Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4), 2018.

[88] Ilyes Khemakhem, Diederik P Kingma, and Aapo Hyvärinen. Variational autoencoders and nonlinear ICA: A unifying framework. *arXiv:1907.04809*, 2019.

[89] Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[90] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[91] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

[92] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, volume 19, 2014.

[93] Yajing Kong, Liu Liu, Zhen Wang, and Dacheng Tao. Balancing stability and plasticity through advanced null space in continual learning. In *European Conference on Computer Vision*, 2022.

[94] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6), 2004.

[95] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical reports*, 2009.

[96] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[97] Ajay Kumar. Computer-vision-based fabric defect detection: A survey. *IEEE transactions on industrial electronics*, 55(1), 2008.

[98] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, 2017.

[99] Quoc V Le, Alexandre Karpenko, Jiquan Ngiam, and Andrew Y Ng. ICA with reconstruction cost for efficient overcomplete feature learning. In *Advances in neural information processing systems*, 2011.

[100] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR 2011*. IEEE, 2011.

[101] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

[102] Olivier Ledoit and Michael Wolf. Honey, i shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4), 2004.

[103] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[104] Te-Won Lee and Terrence J Sejnowski. Independent component analysis for mixed subgaussian and super-gaussian sources. In *Technical report, Computational Neurobiology Lab, The Salk Institute, La Jolla*. Citeseer, 1998.

[105] Avraham Levy and Michael Lindenbaum. Sequential karhunen-loeve basis extraction and its application to images. In *International Conference on Image Processing. ICIP*. IEEE, 1998.

[106] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. Cutpaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2021.

[107] Xiaocan Li, Shuo Wang, and Yinghao Cai. Tutorial: Complexity analysis of singular value decomposition and its variants. *arXiv:1906.12085*, 2019.

[108] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *13th European Conference on Computer Vision*. Springer, 2014.

[109] DC Liu and J Nocedal. On the limited memory method for large scale optimization. *Mathematical Programming: Series A and B*, 1989.

[110] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 2008.

[111] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[112] Xudong Lv, Boya Wang, Ziwen Dou, Dong Ye, and Shuo Wang. Lccnet: Lidar and camera self-calibration using cost volume network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[113] Prasanta C. Mahalanobis. On the generalised distance in statistics. *Proceedings of the National Institute of Science, India*, 2(1), 1936.

[114] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[115] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24. Elsevier, 1989.

[116] Larry Medsker and Lakhmi C Jain. *Recurrent neural networks: design and applications*. CRC press, 1999.

[117] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

[118] Dubravko Miljković. Review of novelty detection methods. In *The 33rd International Convention MIPRO*. IEEE, 2010.

[119] Subodh Mishra, Gaurav Pandey, and Srikanth Saripalli. Extrinsic calibration of a 3d-Lidar and a camera. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020. doi: 10.1109/IV47402.2020.9304750.

[120] Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. Essentials for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[121] Jair Montoya-Martínez, Jean-François Cardoso, and Alexandre Gramfort. Caveats with stochastic gradient and maximum likelihood based ICA for EEG. In *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2017.

[122] Eric Moreau and Odile Macchi. Self-adaptive source separation. II. comparison of the direct, feedback, and mixed linear network. *IEEE transactions on signal processing*, 46(1), 1998.

[123] B. Nagy, L. Kovacs, and C. Benedek. Sfm and semantic information based online targetless camera-Lidar self-calibration. In *Proceedings - International Conference on Image Processing*, vol. 2019-Septe. IEEE Computer Society, sep, 2019. ICIP.

[124] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using typicality. *arXiv:1906.02994*, 5, 2019.

[125] Loris Nanni, Stefano Ghidoni, and Sheryl Brahnam. Handcrafted vs. non-handcrafted features for computer vision classification. *Pattern Recognition*, 71, 2017.

[126] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng. Tiled convolutional neural networks. In *Advances in neural information processing systems*, 2010.

[127] Yasunori Nishimori. Learning algorithm for independent component analysis by geodesic flows on orthogonal group. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1999.

[128] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3), 1982.

[129] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv:2304.07193*, 2023.

[130] Gaurav Pandey, James R McBride, Silvio Savarese, and Ryan M Eustice. Automatic extrinsic calibration of vision and Lidar by maximizing mutual information. *Journal of Field Robotics*, 32(5), 2015.

[131] Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In *Proceedings of the 24th ACM SIGKDD*, 2018.

[132] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[133] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 2011.

[134] Pramuditha Perera and Vishal M Patel. Learning deep features for one-class classification. *IEEE Transactions on Image Processing*, 28(11), 2019.

[135] Pramuditha Perera, Poojan Oza, and Vishal M Patel. One-class classification: A survey. *arXiv:2101.03064*, 2021.

[136] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99, 2014.

[137] Robert J Plemmons. Matrix analysis (roger a. horn and charles r. johnson), 1988.

[138] Mark D Plumbley. Geometrical methods for non-negative ICA: Manifolds, Lie groups and toral subalgebras. *Neurocomputing*, 67, 2005.

[139] Mark D Plumbley. Geometry and manifolds for independent component analysis. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4. IEEE, 2007.

[140] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv:2010.16061*, 2020.

[141] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*. Springer, 2020.

[142] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[143] Liqun Qi. Some simple estimates for singular values of a matrix. *Linear algebra and its applications*, 56, 1984.

[144] Domen Racki, Dejan Tomazevic, and Danijel Skocaj. A compact convolutional neural network for textured surface anomaly detection. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018.

[145] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCarRL: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017.

[146] Oliver Rippel, Patrick Mertens, and Dorit Merhof. Modeling the distribution of normal data in pre-trained deep features for anomaly detection. In *International Conference on Pattern Recognition (ICPR)*. IEEE, 2021.

[147] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 2008.

[148] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. *CVPR*, 2022.

[149] Peter J Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3), 1999.

[150] Marco Rudolph, Bastian Wandt, and Bodo Rosenhahn. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

[151] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International conference on machine learning*. PMLR, 2018.

[152] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 2021.

[153] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088), 1986.

[154] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3), 2015.

[155] Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6), 1989.

[156] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[157] Kripasindhu Sarkar, Basavaraj Hampiholi, Kiran Varanasi, and Didier Stricker. Learning 3d shapes as multi-layered height-maps using 2d convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[158] Michele Scarpiniti, Simone Scardapane, Danilo Comminiello, Raffaele Parisi, and Aurelio Uncini. Effective blind source separation based on the ADAM algorithm. In *Multidisciplinary Approaches to Neural Computing*. Springer, 2018.

[159] N. Schneider, F. Piewak, C. Stiller, and U. Franke. Regnet: Multimodal sensor registration using deep neural networks. *in IEEE Intelligent Vehicles Symposium, Proceedings*, 7995, 2017.

[160] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12, 1999.

[161] Michael E Schuckers and Michael E Schuckers. Receiver operating characteristic curve and equal error rate. *Computational Methods in Biometric Authentication: Statistical Methods for Performance Evaluation*, 2010.

[162] S Easter Selvan, Alexandru Mustatea, C Cecil Xavier, and Jean Sequeira. Accurate estimation of ICA weight matrix by implicit constraint imposition using Lie group. *IEEE transactions on neural networks*, 20(10), 2009.

[163] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv:1708.00489*, 2017.

[164] Shelly Sheynin, Sagie Benaim, and Lior Wolf. A hierarchical transformation-discriminating generative model for few shot anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[165] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv:1703.00810*, 2017.

[166] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676), 2017.

[167] Javier Silvestre-Blanes, Teresa Albero-Albero, Ignacio Miralles, Rubén Pérez-Llorens, and Jorge Moreno. A public fabric database for defect detection methods and results. *Autex Research Journal*, 19(4), 2019.

[168] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[169] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[170] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 2014.

[171] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[172] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[173] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*. PMLR, 2019.

[174] Zachary Taylor and Juan Nieto. Automatic calibration of Lidar and camera images using normalized mutual information. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. Citeseer, 2013.

[175] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*. Springer, 2020.

[176] Asher Trockman and J Zico Kolter. Patches are all you need? In *ICLR 2022*, 2022.

[177] Gido M van de Ven and Andreas S Tolias. Three continual learning scenarios. In *NeurIPS Continual Learning Workshop*, volume 1, 2018.

[178] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alexander Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *Arxiv*, 2016.

[179] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

[180] Tamás Varga and Horst Bunke. Generation of synthetic training data for an hmm-based handwriting recognition system. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* IEEE, 2003.

[181] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[182] Aseem Wadhwa and Upamanyu Madhow. Bottom-up deep learning using the hebbian principle, 2016.

[183] Xiang Wan, Lilan Liu, Sen Wang, and Yi Wang. A transfer learning strip steel surface defect recognition network based on vgg19. In *International Workshop of Advanced Manufacturing and Automation*. Springer, 2019.

[184] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021.

[185] Weimin Wang, Shohei Nobuhara, Ryosuke Nakamura, and Ken Sakurada. Soic: Semantic online initialization and calibration for Lidar and camera. *arXiv:2003.04260*, 2020.

[186] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3), 2020.

[187] Satosi Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development*, 4(1), 1960.

[188] Peter H Westfall. Kurtosis as peakedness, 1905–2014. RIP. *The American Statistician*, 68 (3), 2014.

[189] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4), 2002.

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

[191] Jihun Yi and Sungroh Yoon. Patch svdd: Patch-level svdd for anomaly detection and segmentation. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[192] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[193] K. Yuan, Z. Guo, and Z. Jane Wang. Regnet: Tolerance aware Lidar-camera online calibration with geometric deep learning and generative model. *IEEE Robotics and Automation Letters*, 2020.

[194] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

[195] Vicente Zarzoso and Pierre Comon. Robust independent component analysis by iterative maximization of the kurtosis contrast with algebraic optimal step size. *IEEE Transactions on neural networks*, 21(2), 2009.

[196] Hui Zhang, Jason E Fritts, and Sally A Goldman. Image segmentation evaluation: A survey of unsupervised methods. *computer vision and image understanding*, 110(2), 2008.

[197] Lily Zhang, Mark Goldstein, and Rajesh Ranganath. Understanding failures in out-of-distribution detection with deep generative models. In *International Conference on Machine Learning*. PMLR, 2021.

[198] Ganning Zhao, Jiesi Hu, Suya You, and C-C Jay Kuo. Calibdnn: multimodal sensor calibration for perception using deep neural networks. In *Signal Processing, Sensor/Information Fusion, and Target Recognition*, volume 11756. SPIE, 2021.

[199] Hong-Yu Zhou, Avital Oliver, Jianxin Wu, and Yefeng Zheng. When semi-supervised learning meets transfer learning: Training strategies, models and datasets. *arXiv:1812.05313*, 2018.

[200] Huiwei Zhou, Yan Zhang, Degen Huang, and Lishuang Li. Semi-supervised learning with transfer learning. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, 2013.

[201] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 2018.