Recognition of Objects and Their Defects

DOI:10.18136/PE.2022.826

A Thesis Submitted for the Degree of Doctor

of Philosophy in Computer Science

Amr Mohamed Abdelhameed Nagy Abdo

Supervisor: Dr. László Czúni

Department of Electrical Engineering and Information Systems

Doctoral School of Information Science and Technology

University of Pannonia

Veszprém, Hungary

2022

RECOGNITION of OBJECTS and THEIR DEFECTS

Thesis for obtaining a PhD degree in the Doctoral School of Information Science and Technology of the University of Pannonia

in the research field of Computer Sciences

Written by:
Amr Mohamed Abdelhameed Nagy Abdo

Supervisor(s): **Dr. László Czúni**

propose acceptance (yes / no)

.....
(supervisor/s)

As reviewer, I propose acceptance of the thesis:

Name of Reviewer: ..... (yes / no)

.....
(reviewer)

Name of Reviewer: ..... (yes / no)

.....
(reviewer)

The PhD-candidate has achieved ..... % at the public discussion.

Veszprém,

.....
(Chairman of the Committee)

The grade of the PhD Diploma .....(.....%)

.....
(Chairman of UDHC)

Acknowledgments

First of all, all gratitude and thankfulness to ALLAH for guiding and aiding me to bring this work out to light. I am deeply indebted to my supervisor Dr. László Czúni for his kindness and scientific support throughout the period it took to prepare this work in its final form. I would like to thank the Director of the Doctoral School Prof. Ferenc Hartung and the former Director of the Doctoral School Prof. Katalin Hangos for their help and support, also I wish to express my deep gratitude to the staff of the Department of Electrical Engineering and Information Systems, Faculty of Information Technology, University of Pannonia for their guidance and moral support during the making of this work.

My deep thanks to the Egyptian Ministry of Higher Education and Scientific Research and to the Hungarian Ministry of Higher Education for their cooperation with Egypt to have my study in Hungary.

This work has been partly implemented by the TKP2020-NKA-10 project with the support provided by the Ministry for Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the 2020 Thematic Excellence Programme funding scheme. We acknowledge the financial support of the Hungarian Scientific Research Fund grant OTKA K 135729 and OTKA K 120369.

We are grateful to the NVIDIA corporation for supporting our research with GPUs obtained by the NVIDIA GPU Grant Program.

Many thanks to all my friends and colleagues in Veszprém, they played a very important role in my life. The last, but the most important thanks to my family: my father, my mother, my sisters, my wife, my sons for their love, also for keeping me in their prayers and special thanks to my brother Dr. Abdelhameed Mohamed Abdelhameed Nagy Abdo for his support.

Amr M.Nagy 2022

Kivonat

Az ipar 4.0 által megkövetelt technológia, a kültéri környezet nagy felbontású térképei a járműipar számára, a gyárak digitális ikrei, vagy egyszerűen csak a minőségbiztosítási rendszerek a gyártási folyamatokban megbízható, pontos és gyors objektum felismerést és vizuális ellenőrző technológiákat igényelnek, amelyek gyakran az optikai információk folyamatos megfigyelésére támaszkodnak. Az ilyen megoldások kialakításához hagyományos gépi tanulási módszereket és mély neurális hálózatokat (DNN) alkalmaznak, de a típusaik, felépítésük, paramétereik és tanítási folyamataik több éves kutatásokat igényelnek, hogy általános, széles körben alkalmazható modelleket kapjunk. Ez a disszertáció objektumfelismerő és vizuális ellenőrző megközelítéseket mutat be a tárgyak és hibáik felismerésére különösen akkor, ha kevés felvétel áll rendelkezésünkre a tanításhoz. A disszertáció különböző megoldásokat mutat be az objektumfelismerésre és ezen objektumok hibáinak a felismerésére. Első javaslatom egy keretrendszer annak kimutatására, hogy a rejtett Markov modellek és az inerciális mérőegységek adatai kombinálhatók a neurális hálózatokkal, annak érdekében, hogy javítsák a tárgyfelismerés és a pózbecslés teljesítményét. Ezenkívül javasoltam egy keretrendszert, amivel az aktív látást és a rejtett Markov modellt kombináljuk, abból a célból, hogy javítsuk a felismerési hatékonyságot (különösen, ha a tárgyak el vannak takarva). Ezután javasoltam egy új konvolúciós sziámi neurális hálózati architektúrát a hibák felismerésére nagy osztályszámú közlekedési táblák halmazára. Továbbá, javasoltam egy új mecha-

nizmust is a több képen lefuttatott sziámi hálózatok konfidencia értékeinek SVM-ekkel való kombinálására. Végül bemutattam egy új architektúrát, amellyel az EfficientNet és a randomizált osztályozókat kombináltam few-shots és inkrementális tanuláshoz. Emellett bemutattam, hogy a sziámi hálózatok hogyan használhatók a zero-shot tanuláshoz. A különböző megközelítések teljesítményét több adathalmazon elemeztem. A kiértékelések eredményei azt mutatják, hogy a bemutatott megközelítések általában nagy pontosságot érnek el, versenyképesek a konkurrens módszerekkel. Számos esetben a memória- vagy időigény is jobb mint a vizsgált többi megközelítéseké.

Abstract

The technology required by industry 4.0, high-definition maps of outdoor environments for the vehicle industry, digital twins of factories, or simply quality insurance systems in manufacturing processes require reliable, accurate, and fast object recognition and visual inspection technologies, often relying on continuous monitoring of optical information. To construct such solutions, traditional machine learning methods and deep neural networks are used, but the type, structure, parameters, and training processes of them requires years of research to obtain general, widely applicable models. This dissertation presents object recognition and visual inspection approaches to recognize different objects and to detect their defects, especially when only few shots are available for training.

First, I proposed a framework to show that Hidden Markov Models and inertial measurement units' data can be combined with neural networks to improve the performance for object recognition and pose estimation. Moreover, I proposed a framework to combine active vision with the Hidden Markov Model to enhance the recognition performance (especially when the objects are occluded). Next, I proposed a new fusing convolutional siamese neural network architecture to recognize the defects in case of large number of classes of traffic signs. I also proposed a new mechanism to combine the confidence values of siamese networks, ran on several images, with support vector machines. Finally, I presented a new architecture to combine EfficientNet with randomized classifiers for few-shots and incremental

learning. Additionally, I showed how siamese networks can be utilized for zero-shot learning.

I analyzed the performance of the different approaches on several datasets. The evaluation results show that the presented approaches competitive and achieve high accuracy. In several cases the memory or time requirements are also better than in case of other existing approaches.

Contents

Acknowledgments **iii**

Kivonat **v**

Abstract **vii**

List of Abbreviations **xiv**

List of Figures **xxi**

List of Tables **xxiii**

1 Introduction **1**

1.1 Basic Concepts 2

1.1.1 Object Recognition 2

1.1.2 Visual Inspection 3

1.2 Motivation 3

1.3 Overview of Recognition Approaches 4

1.3.1 Traditional Approaches 4

1.3.2 Deep Learning Approaches 6

1.4 Overview of Visual Inspection Approaches 12

1.4.1 Low-Level Image Processing Approaches 13

1.4.2 High-Level Image Processing Approaches 14

1.5 Research Questions 15
1.6 Author's Contributions 19

2 Object Recognition Techniques using Deep Neural Networks and HMM 21

2.1 Introduction 21
2.2 View-Centered Approach 23
2.3 Hidden Markov Model Explanation 24
2.4 Datasets 25
2.4.1 COIL-100 Dataset 25
2.4.2 COIL-40 Dataset 26
2.4.3 ALOI-1000 Dataset 27
2.5 Improving Object Recognition of CNNs with Multiple Queries and HMMs 28
2.5.1 HMM Object Models 29
2.5.2 Object Views as States in a Markov Model 30
2.5.3 State Transitions 30
2.5.4 Recognition of Single Objects from Multiple Views 32
2.5.5 Proposed Method 32
2.5.6 Experimental Results 33
2.6 Hidden Markov Models Based on Convolutional Neural Network for Pose Estimation 34
2.6.1 Proposed Method 36
2.6.2 Tests and Evaluations 37
2.7 Active Multiview Recognition with Hidden Markov Temporal Support 40
2.7.1 Recognition of Objects from Multiple Views by Weak Global Classifiers 41

2.7.2 Active Recognition with HMM 43
2.7.3 Experiments and Evaluations 45
2.7.4 About Space and Time Complexity 48
2.7.5 An Alternative: ConvLSTM 48
2.7.6 Comparison to LSTM 50
2.7.7 Comparison to LSTM with Explicit Orientation 51
2.8 Summary 52

3 Detecting Traffic Sign Defects 54
3.1 Datasets 56
3.1.1 Traffic Signs Distortion Dataset (TSD Version I) 56
3.1.2 Traffic Signs Distortion Dataset (TSD Version II) 59
3.2 The Proposed Fusioning Convolutional Siamese Neural Network Architecture (FCSNN) 60
3.2.1 Training 62
3.2.2 N-way One-Shot Classification 62
3.2.3 Experimental Results 63
3.3 Recognition of Traffic Signs Defects with SVM based on CNN Confidence Values 65
3.3.1 Experiments and Discussion 66
3.3.2 Comparing FCSNN-SVM with Different DNNs Models 68
3.4 Summary 70

4 Classification, Zero and Fast Few-Shot Learning of Steel Surface Defects 72
4.1 Introduction 72
4.2 Related Works 77

4.2.1 Detection and Classification of Steel Surface Defects 78
4.2.2 Zero-Shot Learning 80
4.2.3 Few-Shot Learning 81
4.3 The Benchmark Datasets 83
4.3.1 Northeastern University Surface Defect Database (NEU) . . . 84
4.3.2 Xsteel Surface Defect Dataset (X-SSD) 84
4.4 Zero-Shot Learning and Classification with Siamese Neural Network . 85
4.4.1 The Proposed Methods 86
4.4.2 Experiments and Discussion 88
4.4.3 Further Discussion 89
4.5 Proposed Methods for Classification and Few-Shot Learning 91
4.5.1 Classification of Defects with EfficientNet-B7 91
4.5.2 Fast Few-Shot Learning of Steel Surface Defects with Randomized Network 93
4.6 Experimental Results of Classification and Discussion 99
4.6.1 Classification Results on the NEU Dataset 99
4.6.2 Classification Results on the X-SSD Dataset 100
4.7 Testing Few-Shot Learning with EffNet+RC 101
4.7.1 Time Complexity 107
4.8 Summary 108

5 Conclusion **111**

5.1 New Scientific Results 113
5.2 Publications 116
5.2.1 Publications Related to this Thesis 116
5.2.2 Publications not Related to this Thesis 117
5.2.3 Other Presentations 118

Bibliography 119

List of Abbreviations

6-DoF Six Degrees of Freedom

ACNN Adaptive Learning Rate of the Convolutional Neural Networks

ANN Artificial Neural Network

AV Active Vision

BB8 A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth

BSR Batch Spectrum Regularization

CAD Computer-Aided Design

CD-FSL Cross-Domain Few-Shot Learning

CEDD Color and Edge Directivity Descriptor

CIFAR Canadian Institute for Advanced Research

CNN Convolutional Neural Network

CPN Classification Priority Network

ConvLSTM Convolutional Long Short-term Memory

DA Domain Adaptation

DDN Defect Detection Network

DL Deep Learning

DNN Deep Neural Network

DeVGG19 Decode VGG19

EHD Edge Histogram Descriptor

ELM Extreme Learning Machine

EffNet EfficientNet

EffNet+FtC Frozen EfficientNet Backbone with Back-Propagation Fine-Tuned Weights

EffNet+RC EfficientNet-B7 Backbone and the Randomized Classifier (RC)

ExtVGG16 Extended VGG16

FCSNN Fusing Convolution Siamese Neural Network

Ft EffNet-Unbal Fine-Tuned EfficientNet-B7 with Unbalanced Data

Ft EffNet Fine-Tuned EfficienNet-B7

GCN Graph Convolution Network

GPU Graphical Processing Unit

GT Ground Truth

HMM Hidden Markov Model

HOG Histogram of Oriented Gradients

HSV (Hue, Saturation, Value) Color Space

IMUs Inertial Measurement Units

IoT Internet of Things

LSTM Long Short-term Memory Network

MB Motion Blur

MFN Multi-level Feature Fusion Network

MG-CNN Multiple Group Convolutional Neural Network

MMGCN Multiple Micrographs Graph Convolutional Network

MNIST Modified National Institute of Standards and Technology Database

NEU Northeastern University Surface Defect Database

PoseCNN A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes

R-CNN Region-Based Convolutional Neural Network

RC Randomized Classifier

RECOS REctified-COrrelations on a Sphere

RGB Red, Green and Blue

RN Relation Network

ROI Region of Interest

RVFL Random Vector Functional Links Network

RestNet Residual Neural Network

SIFT Scale Invariant Feature Transform

SLFN Single Layer Feed Forward-Neural Network

SNN Siamese Neural Networks

SSD-6D Making RGB-based 3D Detection and 6D Pose Estimation Great Again

SSIM Structural Similarity Index Measure

SVM Support Vector Machine

UAV Unmanned Aerial Vehicle

VGG16 Visual Geometry Group from Oxford 16

VGG19 Visual Geometry Group from Oxford 19

VSD VSD network

X-SSD Xsteel Surface Defect Data-set

YOLOv4 You Look Only Once Network Version 4

Yolo You Look Only Once Network

K-NN K-Nearest Neighbor

mAP Mean Average Precision

List of Figures

1.1 Neural network architecture with three layers. 7

1.2 Feed forward neural networks architecture. 8

1.3 General architecture of convolutional neural networks. 9

1.4 Recurrent neural network architecture. 10

1.5 Example for siamese neural network architecture. 11

2.1 Model generation setup with target object in the centre. 23

2.2 First two lines: Clear samples from COIL-100. 3^{rd} line: Example queries loaded with Gaussian additive noise. 4^{th} line: Example queries loaded with motion blur. 5^{th} and 6^{th} lines: Occluded examples. 26

2.3 Top line: example objects from the COIL-100 dataset. Bottom line: objects with different backgrounds. 27

2.4 First two lines: Clear samples from ALOI-1000. 3^{rd} line: Example queries loaded with Gaussian additive noise. 4^{th} line: Example queries loaded with motion blur. 27

2.5 Geometrical explanation of transition probabilities. 31

2.6 Proposed architecture to combine CNNs with HMM model. 32

2.7 The average Hit-Rate for 40 objects with different number of queries. 34

2.8 Average orientation error at different number of queries for VGG16 only (orange) and VGG+HMM (blue). 38

2.9 Average orientation error for each object, in case of two queries, for VGG16 only (orange), VGG+HMM (blue), and VGG+mHMM with constant transition probabilities (green dotted). 38
2.10 Mostly rotational invariant objects. 39
2.11 Not rotational invariant objects. 39
2.12 Average orientation error at different number of queries for VGG+HMM with rotational invariant objects (blue) and VGG+HMM without rotational invariant objects (Green). 40
2.13 Not completely rotational invariant objects with rotations resulting in very similar views. 40
2.14 Object is recognized continuously in a sequence of queries. New queries (Q_2 and Q_3) are planned by the analysis of previous shot(s) (Q_1). 42
2.15 Overview of the proposed multiview method. 43
2.16 Comparison of non-active and active recognition when all queries are occluded. Top graph: COIL-100, bottom graph: ALOI-1000 datasets. Continuous curves show the GT being in the top 10 items of the retrieval list. 45
2.17 Comparison of active and non-active recognition on the distorted COIL dataset. First graph: motion blur, second graph: Gaussian noise. 46
2.18 Comparison of active and non-active recognition on the distorted ALOI dataset. First graph: images with motion blur, second graph: images with Gaussian noise. 47
2.19 Overview of the tested ConvLSTM framework in case of four sequential queries. 49

2.20 Comparison of active HMM and ConvLSTM on occluded queries. . . 51
2.21 LSTM architecture with explicit pose. 51
2.22 LSTM with explicit orientation results. 52

3.1 Researchers at McAfee placed a two-inch long piece of electrical tape horizontally across the middle of the '3' on a 35 mph (left) speed limit sign, causing the car's camera system to misread it as 85 mph (right) [62]. 55
3.2 Traffic sings, as examples for the 7 distortion classes, under investigation: faded, covered, scribbled, correct, covered and faded, covered and scribbled, faded and scribbled. 56
3.3 Traffic signs in the training dataset with their class codes. 57
3.4 Percentage of images for each defect class in the training dataset. . . 57
3.5 Distribution of traffic signs of the training data with each error class. 57
3.6 Examples images for the untrained traffic sign types. 58
3.7 Distribution of the untrained traffic signs with each error class. 58
3.8 Examples for the 4 defect classes under investigation: (**a**) faded, (**b**) covered, (**c**) correct,(**d**) scribbled. 59
3.9 The proposed fusioning convolutional siamese neural network architecture. 60
3.10 Examples of training pairs with their labels. 61
3.11 An example for 5-way one-shot testing. Since the pair with the same defect has the highest confidence, the network made a correct classification. 63
3.12 Combination of FCSNN with SVM in the training process. SVM is trained on the confidence values and true labels of several comparisons of training images and the elements of the support set. 65

4.1 Examples for the six kinds of defect classes of the Northeastern University surface defect database (NEU) [73]: (**a**) crazing (Cr), (**b**) inclusion (In), (**c**) patches (Pa), (**d**) pitted surface (Ps), (**e**) rolled-in scale (Rs), (**f**) scratches (Sc). 73

4.2 Examples for the seven kinds of defect classes of the Xsteel surface defect dataset (X-SSD) [72]: (**a**) inclusion (Si), (**b**) red iron sheet (Ri) , (**c**) iron sheet ash (Is), (**d**) scratches (Ss), (**e**) oxide scale of plate system (Op), (**f**) finishing roll printing (Fr), (**g**) oxide scale of temperature system (Ot). 74

4.3 Proposed siamese architecture based on the fusing convolutional siamese neural network (FCSNN). 87

4.4 Accuracy and Loss value curves when training classification on X-SSD. First row: trainable layers of VGG16 backbone. Second row: freezed VGG16 backbone. 90

4.5 The proposed architecture (EffNet+RC) for few-shot learning. In phase 0, we train the backbone through a large number of samples of base classes. In phase 1 (and further phases), we use features extracted by the previously trained backbone. Here, only the weights W are computed with the help of a few samples, while weights R are random and fixed. 95

4.6 (**Left**) Confusion matrix of EfficientNet-B7 classification of the seven error types of the X-SSD dataset. (**Right**) An example image of the Si class and an Is defect wrongly classified as Si. 101

4.7 The illustration of the **Effnet+RC** and the **EffNet+FtC** networks. While the structures are similar, there is a big difference as the former should not be trained with backpropagation. 103

4.8 The accuracy of the different classification models for all classes (base and new). Ft EffNet: fine-tuned EfficienNet-B7; EffNet+RC: fixed EfficienNet-B7 backbone plus randomized classifier; EffNet+FtC: fixed EfficienNet-B7 backbone plus fine-tuned classifier; Ft EffNet Unbal: fine-tuned EfficienNet-B7 with unbalanced dataset. 104

4.9 The accuracy of the different classification models for base classes. Ft EffNet: fine-tuned EfficienNet-B7; EffNet+RC: fixed EfficienNet-B7 backbone plus randomized classifier; EffNet+FtC: fixed EfficienNet-B7 backbone plus fine-tuned classifier; Ft EffNet Unbal: fine-tuned EfficienNet-B7 with unbalanced dataset. 105

4.10 The accuracy of the different classification models for new classes. Ft EffNet: fine-tuned EfficienNet-B7; EffNet+RC: fixed EfficienNet-B7 backbone plus randomized classifier; EffNet+FtC: fixed EfficienNet-B7 backbone plus fine-tuned classifier; Ft EffNet Unbal: fine-tuned EfficienNet-B7 with unbalanced dataset. 105

4.11 Elapsed training time of the different models under investigation. . . 108

List of Tables

1.1 Summarizing research questions and datasets information of Thesis I. 17

1.2 Summarizing research questions and datasets information of Thesis II. 18

1.3 Summarizing research questions and datasets information of Thesis III. 18

2.1 Average Hit-Rates (%) in case of different query distortions applying 8 sequential queries. 47

2.2 Memory and running time requirements of the HMM and LSTM models. 48

3.1 Accuracy of three siamese networks on untrained traffic signs (TSD version I) in three independent tests. 64

3.2 FCSNN-ResNet50-SVM evaluation on TSD version I. The accuracy on the 490 test images of untrained classes of traffic signs was 77%. . 67

3.3 FCSNN-ResNet50-SVM evaluation on TSD version II. The accuracy on the 669 test images of TSD was 88%. 68

3.4 Evaluation of SNNs and feature based classification approaches. In case of four methods (first four lines) we could increase performance by choosing the defect class with the highest average confidence on the support support set. 69

4.1 Comparison of accuracy values depending whether feature extraction was using ImageNet weights (Freeze) or were optimized (Unfreeze). . 90

4.2 Running and testing environment. 99

4.3 Comparison of the classification accuracy of different models on the NEU dataset. If not specified then information is based on [98]. Training/testing ratio is 80/20 in general. Best values are highlighted in bold. 100

4.4 Comparison of EfficientNet-B7 with other models on X-SSD (all data are based on paper [72] except for ExtVGG16 [78] and EfficientNet-B7). Best values are highlighted in bold. 101

4.5 The distribution of original X-SSD images in the training and testing datasets for few-shot learning. K is the number of shots in the experiments. The real number of images fed to the network during training is larger due to augmentation. 102

4.6 The accuracy and rank of the different classification models evaluated on all classes (base and new). Best values are highlighted in bold . . 106

4.7 The weighted F_1 score and rank of the different classification models evaluated on all classes (base and new). Best values are highlighted in bold 107

4.8 Time spent for training at different number of k-shots. 108

Chapter 1

Introduction

In recent years, object recognition, as one of the most fundamental and demanding topics in computer vision, has received great attention. It is considered as one of the most important tasks that brings significant impacts to the society since many applications are built upon object recognition techniques, such as object recognition of traffic signs, shopping applications, human recognition in surveillance, road object recognition in autonomous driving and daily object recognition in handheld devices and robotics. Unfortunately, 3D object recognition remains one of the core problems in computer vision. 3D object recognition in case of real-life environments using handheld devices or robots is a difficult task due to changing viewpoints, varying 3D to 2D projections, possible different noises (e.g. motion blur, color distortion), and the limited computational resources and memory. Also, often video-stream acquisition is crucial for many computer vision applications such as unmanned aerial vehicle (UAV) applications, manufacturing industry or video surveillance. The objective of object detection is to develop computational models and techniques that provide one of the most basic pieces of information needed by computer vision applications: What objects and where are they?

As one of the fundamental problems of computer vision, object recognition and

detection forms the basis of many other computer vision tasks, such as visual inspection [1], object tracking [2], instance segmentation [3], image captioning [4], etc.

The next important step after recognizing the objects by machine is to detect anomalies on it. Visual inspection technology allows differentiating anomalies in objects mimicking human visual inspection. While it suggests monitoring with a minimum amount of human activity, applying the same solution to a wide variety of defect types is challenging. There is a wide field of such applications including traffic signs defects, steel surface defects, solar panels, automated product manufacturing, railway industry, casting or welding, and healthcare. In order to meet industrial expectations, there is a strong need to achieve high performance in automated visual inspection.

This work is concerned with research methods that can recognize objects with traditional and deep learning approaches, building reliable solutions to recognize different types of defects on recognized objects with high accuracy based on two major approaches (feature detection with classification and siamese DNNs).

1.1 Basic Concepts

1.1.1 Object Recognition

Object recognition is a general term to describe a collection of related computer vision tasks that involve identifying objects in digital photographs or motion pictures. Image classification involves predicting the class of objects in an image. Object localization refers to identifying the location of one or more objects in an image and drawing a bounding box around their extent. Object detection combines these two tasks and localizes and classifies one or more objects in an image.

When a user or practitioner refers to object recognition, they often mean object detection.

1.1.2 Visual Inspection

Visual inspection means observation of the same type of objects repeatedly to detect anomalies. Visual inspection systems have been purpose-built for the production environment and addresses a wide range of use cases across the automotive, electronics, semiconductor, and other industrial sectors.
One of the most challenging and time-consuming processes in the production process is product inspection, particularly visual inspection. The importance of the inspection process has been heightened by the demands of today's manufacturing environment [5]. These include:

- Quality levels are so high that sampling inspection is not applicable.
- Production rates are so high that manual inspection is not feasible.
- Tolerances are so tight that manual visual inspection is inadequate.

These are a few of the reasons why visual inspection will dominate quality control in the future manufacturing arena.

1.2 Motivation

Technology required by industry 4.0 in manufacturing processes requires reliable, accurate, and fast object recognition and visual inspection technologies, often relying on continuous monitoring of optical information. Additionally, lightweight methods for object recognition may be very useful for a variety of applications such as drones

or wearable computing. Moreover, as automation is wide-spreading in manufacturing processes, the need for automatic anomaly detection is growing. That is, if we trained our machine intelligence for a given task, there is always a non-zero probability that unseen events might happen that the system is not trained to handle. A part of this problem is few-shot learning, where new kinds of errors appear to be classified as soon as possible, typically with a very low number of training samples. We have to carry out incremental learning, since previously trained knowledge should not be forgotten. Moreover, the re-training of the whole architecture would be resource demanding: the large amount of time, memory, and processing power is typically not available on site or in time.

1.3 Overview of Recognition Approaches

The evolution of object recognition and detection has usually gone through two historical periods in the last two decades: Traditional object detection period (before 2014) and deep learning-based detection period (after 2014).

1.3.1 Traditional Approaches

Traditional object detection and recognition methods are built on handcrafted features, shallow trainable architectures and mainly start from extracting these features by traditional image processing methods. These feature descriptors are generally combined with traditional machine learning classification algorithms. Some of the used popular feature extraction techniques, includes:

- General Hough transform [6] (proposed by Ballard H. in 1981) can be regarded as a good approach to complete geometric feature extraction. The Hough transform is a technique for separating characteristics of a specific shape inside

an image. The classical Hough transform is most typically employed for the detection of regular curves such as lines, circles, ellipses, because it requires the desired features to be provided in some parametric form. The generalized Hough transform can be used to detect arbitrary shapes (i.e., shapes having no simple analytical form).

- Harris corner detector [7] proposed in 1988, extracts objects features by detecting corners. This method extracts corner features from two images and calculates the correlation degree between their points to detect objects. The above two methods are sensitive to the features transformation of the image, i.e the change of the image size, rotation and grey value affects the final results.

- The scale invariant feature transform (SIFT) [8] method was proposed by Lowe in 2004. The SIFT algorithm is used to detect and describe local features of images commonly known as the 'key-points' of the image. These keypoints are scale and rotation invariant and can be used for various computer vision applications, like image matching, object detection, scene detection, etc.

- Histogram of oriented gradients (HOG) [9] is basically a feature descriptor that is utilized to detect objects. The histogram of oriented gradients descriptor technique includes occurrences of gradient orientation in localized portions of an image, such as a detection window, the region of interest (ROI), among others. One advantage of HOG-like features is their simplicity, and the easy understanding of information they carry.

- The Viola-Jones algorithm [10], is named after two computer vision researchers who proposed the method in 2001: Paul Viola and Michael Jones. It is the first framework for object detection which gives viable results for real-time situations on many platforms. It aims to target the problem of face detection

but can be trained to detect different object classes. The Viola-Jones algorithm has 4 main steps: selecting Haar-like features, creating an integral image, running AdaBoost training, and creating classifier cascades.

- The color and edge directivity descriptor (CEDD) [11] was found to be one of the most robust, fast and compact among those. CEDD is a block-based approach where each image block is classified into one of 6 texture classes (non-edge, vertical, horizontal, 45-degree diagonal, 135-degree diagonal, and non-directional edges) with the help of the MPEG7 Edge Histogram Descriptor (EHD). Then for each texture class a 24 bin color histogram is generated where each bin represents colors obtained by the division of the HSV (Hue, Saturation, Value) color space. Values of the generated histogram of length 6×24 are then normalized and quantized to 8 bits.

Since features were manually designed in traditional object detection and recognition methods, which are not adaptive to various circumstances and patterns, deep learning methods have appeared.

1.3.2 Deep Learning Approaches

Artificial neural networks (ANNs) are biologically inspired computational networks. It is a group of multiple perceptrons/neurons at each layer. ANNs were mostly limited to three layers until the 1990s, with one input, one hidden layer, and one output layer, as you can see in Figure 1.1. The input layer accepts the inputs, the hidden layer processes the inputs, and the output layer produces the result. Essentially, each layer learns certain weights. ANNs are capable of learning any nonlinear function. Activation functions introduce nonlinear properties to the network. This helps the network learn the complex relationship between input and output. In

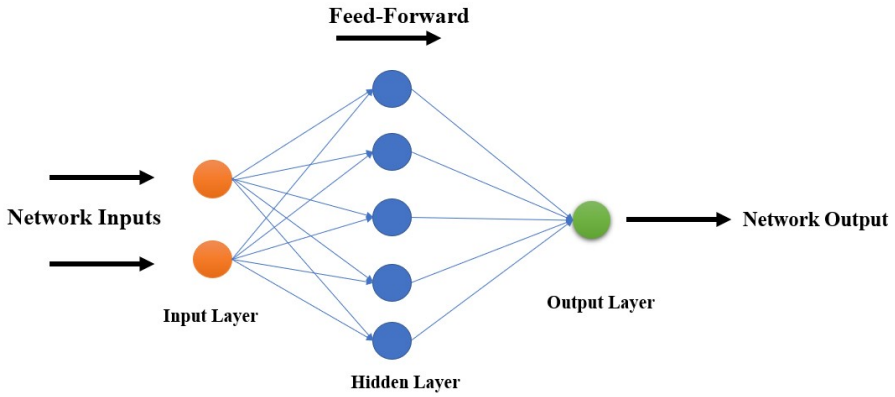

Figure 1.1: Neural network architecture with three layers.

2009 parallelization of ANNs training using Graphical Processing Units (GPUs) was demonstrated in [12]. Subsequently, ANNs have been successfully extended to so-called deep learning models, extending to 100s of hidden layers. It is useful to consider that each neuron in the network transforms the incoming data into a distinct output signal. As the depth of the ANNs is increased the network can transform the data in more complex manners, effectively adding variables to the learned relationship between inputs and outputs. Mathematics may be key to assessing how confident we can be about deep learning. Inspired by the structure of our brains, each "neuron" is a simple mathematical calculation, taking numbers as input and producing a single number as an output. Deep learning usefulness has been proven, but there still are a lot of unanswered questions about the theory of why such deep learning approaches work. There are different techniques such as (explainable AI and graph knowledge representation) which try to simplify the rules, understand the mathematical foundations of deep learning and visualize what is happening inside deep learning. [13] presents a summary of current research in the area and makes a plea for more interpretability in artificial intelligence. Furthermore, it presents two approaches to explain predictions of deep learning models, one method which

computes the sensitivity of the prediction with respect to changes in the input and one approach which meaningfully decomposes the decision in terms of the input variables. In [14], a mathematical model called the "REctified-COrrelations on a Sphere" (RECOS) is proposed to answer these two questions: (1) why a nonlinear activation function is essential at the filter output of all intermediate layers? (2) what is the advantage of the two-layer cascade system over the one-layer system?

With the rapid development in deep learning, more powerful tools, which are able to learn semantic, high-level, deeper features, are introduced to address the problems existing in traditional architectures. These models behave differently in network architecture, training strategy, and optimization function. Here is list of different common types of neural networks that exists:

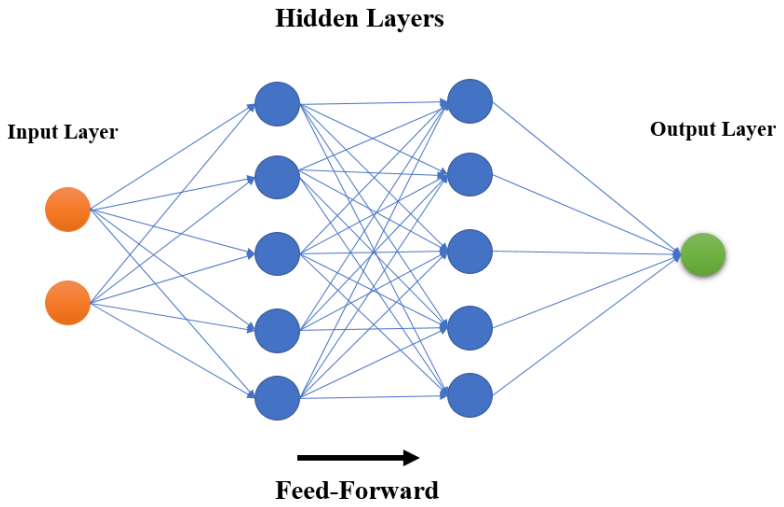

Figure 1.2: Feed forward neural networks architecture.

- Feed Forward Neural Networks
  - A feed forward neural network is an artificial neural network in which the connections between the nodes do not form a cycle, (see Figure 1.2). It is the basic type of neural networks where input data travels in one direction

only, passing through artificial neural nodes and exiting through output nodes. While feed forward neural networks are fairly straightforward, their simplified architecture can be used as an advantage in particular machine learning applications. But there are challenges with feed forward neural networks, while solving an image classification problem: the first step is to convert a 2-dimensional image into a 1-dimensional vector prior to training the model. This has two drawbacks: The number of trainable parameters increases drastically with an increase in the size of the image and it loses the spatial features of an image. Spatial features refer to the arrangement of the pixels in an image and the relationship between them.

- Convolutional Neural Networks (CNNs)

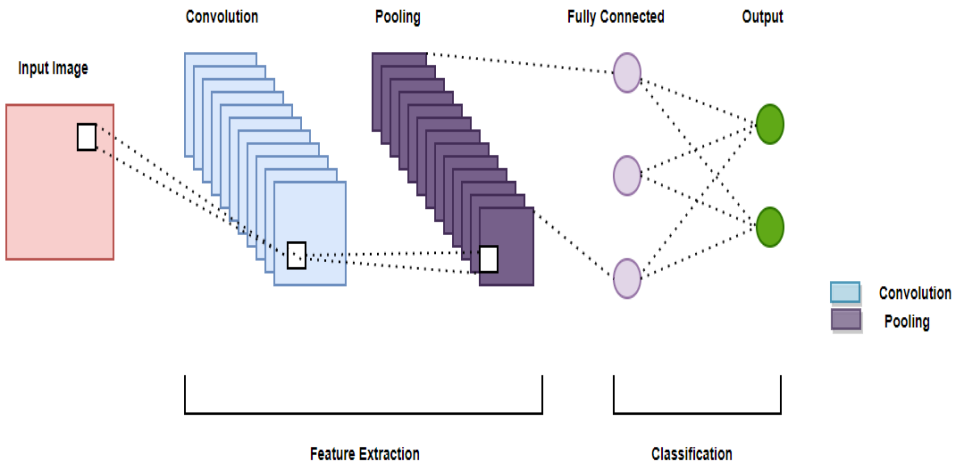

Figure 1.3: General architecture of convolutional neural networks.

  - CNNs are very popular in the deep learning community right now. These CNNs models are being used across different applications and domains, and they're especially prevalent in image and video processing projects.

CNNs capture the spatial features from an image. They help us in identifying the object accurately, the location of an object, as well as its relationship with other objects in an image. CNNs consist of two parts, feature extraction and classification. Consequently, there are three main types of layers to build CNNs architectures, convolutional layer, pooling layer and fully connected layer, (see Figure 1.3). The convolutional layer parameters consist of a set of K learnable filters (i.e., kernels), where each filter has a width and a height, and are nearly always square. The advantages of CNNs is that it learns the filters automatically without mentioning it explicitly. These filters help in extracting the right and relevant features from the input data. Additionally, CNNs follow the concept of parameter sharing. A single filter is applied across different parts of an input to produce a feature map.

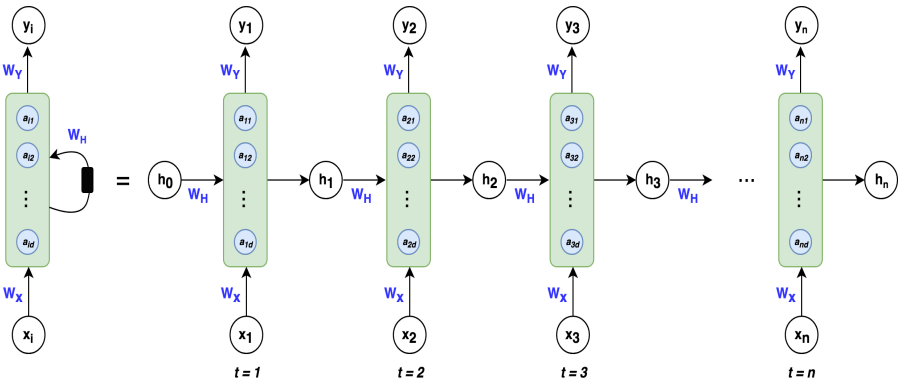

Figure 1.4: Recurrent neural network architecture.

- Recurrent Neural Networks (RNNs)
  - RNNs were created because there were a few issues in the feed-forward neural network: cannot handle sequential data, considers only the current input, and cannot memorize previous inputs. RNNs work on the principle

of saving the output of a layer and feeding this back to the input to help in predicting the outcome of the layer, (see Figure 1.4). The advantages of RNNs is capturing the sequential information present in the input data.

- Siamese Neural Networks (SNNs)

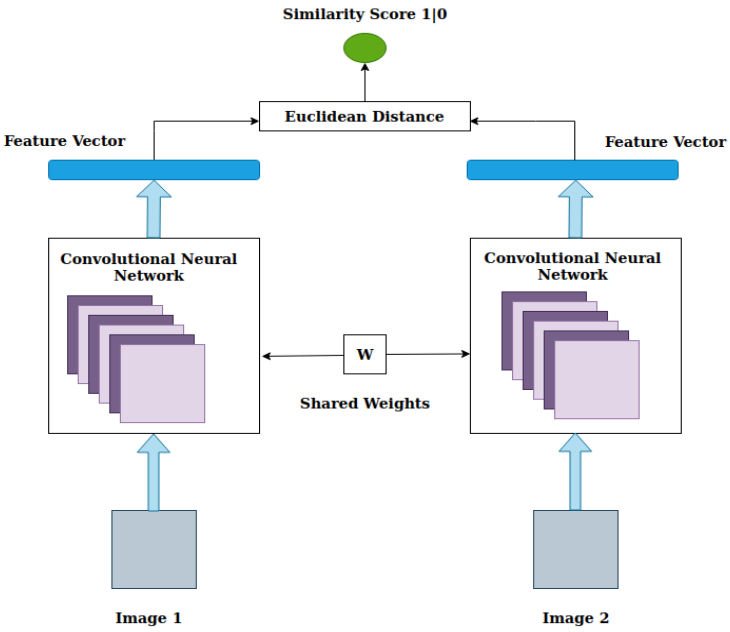

Figure 1.5: Example for siamese neural network architecture.

  - SNNs [15] are neural networks containing two or more sub-networks that are connected by a layer which is typically responsible for the comparison of the features of the branches. The sub-networks are identical: they have the same parameters and weights trained simultaneously. The main idea behind siamese networks is to learn the proper similarity function needed for the efficient comparison of input images in a specific task, (see Figure 1.5).

1.4 Overview of Visual Inspection Approaches

Visual inspection is the oldest and most basic method of inspection. In its simplest form, visual inspection is the process of examining a component or piece of equipment using one's naked eye to look for defects or anomalies. In order to meet industrial expectations, there is a strong need to achieve high performance in automated visual inspection. Image processing and computer vision techniques can be used to achieve these goals and enhance the capabilities of visual inspection. There is a wide field of such applications including automated product manufacturing, railway industry, casting, welding, and healthcare. A general taxonomy of the different defects was presented in [16]: those detectable by only visual methods (e.g. contamination, color or shape errors) and palpable (detectable by touch and vision, e.g. cracks, bumps). The defect detection process can be formulated as either an object detection or a segmentation task. In the object detection approach the goal is to detect each defect in the image and classify it into one of the predefined classes. In the image segmentation approach the problem is essentially solved by pixel wise classification, where the goal is to classify each image pixel as part of a defect or not. In general, object detection and instance segmentation are difficult tasks, as the number of instances in a particular image is unknown and often unbounded. Additionally, due to a wide range of products to be assembled, sensors cannot easily adapt to different materials and shapes of the products to be inspected, variations in the object's position, lighting, and background cause additional challenges.

It is possible to classify the visual inspection approaches into low-level image processing approaches such as statistical [17], structural [18], filter-based [19], model-based [20], and high-level image processing approaches such as supervised [21] unsupervised or semi-supervised classifiers [22].

1.4.1 Low-Level Image Processing Approaches

1.4.1.1 Statistical Methods

Statistical methods concentrate on analyzing the spatial distribution of pixel values in an image. In [17], authors proposed a new algorithm by combining the autocorrelation function with the grey level co-occurrence matrix. First, an autocorrelation function is used to determine the pattern period then the size of the detection window can be obtained thus co-occurrence can be computed. In order to distinguish defective and defect-free images, Euclidean distance is computed between templates and queries.

1.4.1.2 Structural Approaches

Structural approaches primarily concentrate on finding texture primitives of texture images and they are especially suitable for textures with obvious structural attributes. Such elements can be extracted from the texture and defined as texture primitives. Simple grey-scale areas, line segments are often the texture primitives. [18] deals with fabric defect inspection: they propose a prior knowledge guided least squares regression to combine the global structure of texture feature space and the prior from local similarity. This combination helps to generate a more clear irregularity map and to identify various defects accurately and robustly.

1.4.1.3 Filter-Based Approaches

Filter based methods aim to describe textures in a transformed domain using spatial transformations, filters, or filter banks. They are the most widely used approaches for texture analysis, description and inspection. In [19] a banknote defect detection algorithm is presented to detect cracks and scratches on banknote images using a

quaternion wavelet transform and edge intensity. The banknote image is first registered using the least squares method under the quaternion wavelet decomposition framework. The defective features are extracted using the edge difference between the reference image and the test image.

1.4.1.4 Model-Based Approaches

Model-based algorithms describe texture patterns by modelling special distributions or other attributes with certain models. In [20], a differential filter is used to distinguish the defect among the textures, but here a quantitative model characterizing the impact of illumination on the image is developed, based on which the non-uniform brightness in the image can be effectively removed. By comparing the model output against the captured image, the illumination effect can be successfully removed.

1.4.2 High-Level Image Processing Approaches

Although good results may be achieved with low-level methods on the description of defect features and the detection of defects, most of them are application dependent. Recently, with the development of deep learning technology, methods that use deep neural networks are gradually rising in the industrial defect inspection field. We can classify deep learning algorithms into supervised and unsupervised classifiers.

1.4.2.1 Supervised Classifiers

In [21], a framework called classification priority network (CPN) was described for the detection and classification of defects. In CPN, the image is first classified by a multi-group CNN, training different groups of convolution kernels separately to extract the feature map groups of different types of defects. Then, according to the classification result, the feature map groups (named multiple group CNN, MG-

CNN) that may contain defects are separately input into another neural network, to regress the bounding boxes of the corresponding defects.

1.4.2.2 Unsupervised Classifiers or Semi-Supervised Classifiers

Automated defect inspection has long been a challenging task especially in industrial applications, where collecting and labeling large amounts of defective samples are usually harsh and impracticable. In [22], they proposed an approach to detect and localize defects with only defect-free samples for model training. This approach is carried out by reconstructing image patches with convolutional denoising autoencoder networks at different Gaussian pyramid levels, and synthesizing detection results from these different resolution channels. Reconstruction residuals of the training patches are used as the indicator for direct pixel-wise defect prediction, and the reconstruction residual map generated in each channel is combined to generate the final inspection result. This method has two prominent characteristics, which benefit the implementation of automatic defect inspection in practice. First, it is absolutely unsupervised that no human intervention is needed throughout the training process. Second, a multi-modal strategy is utilized in this method to synthesize results from multiple pyramid levels. This approach is capable of improving the robustness and accuracy of the method.

1.5 Research Questions

Deep learning (DL) is used in the domain of digital image processing to solve difficult problems (e.g. image colourization, classification, segmentation and detection). DL methods such as CNNs mostly improve prediction performance using big data and plentiful computing resources and have pushed the boundaries of what was

possible. Problems that were assumed to be unsolvable are now being solved with super-human accuracy. Image classification is a prime example of this. Since appearing by [23] in 2012, DL has dominated the domain due to substantially better performance compared to traditional methods. Additionally, DL is not going to solve all CV problems [24]. There are some problems where traditional techniques with global features are a better solution. [24], authors analyzed the benefits and drawbacks of both approaches. [25], analyzed the performance between several classic hand-crafted and deep keypoint detector and descriptor methods such as LF-Net and SuperPoint. According to the results, some classic detector-descriptor combinations can outperform pretrained deep models while other classic and deep techniques are still equivalent. [26], compared the differences of traditional and deep learning algorithms to learn more about which is better suited for a certain application. The two difficult ill-posed problems that were investigated are multispectral image registration and faint edge recognition.

Main questions, this thesis is dealing with, are summarized in the following tables:

Table 1.1 shows the problems, datasets and results related to improving multi-view object recognition.

Table 1.2 shows the problems, datasets and results related to detection of defects of traffic signs.

Table 1.3 shows the problems, datasets and results related to steel surface defect detection.

Table 1.1: Summarizing research questions and datasets information of Thesis I.

Thesis	**Task to be solved**	**Proposed solution**	**Dataset**	**Advantages**
1.1	- How can we improve the performance of single shot CNN detectors in multiple view tasks?	- HMMs can be used to combine the CNNs and IMUs data for object recognition and roughly pose estimation. - We use pre-trained CNNs.	**COIL-40 Dataset** - 40 objects - Each object divided into 8 subclasses (8 poses) by sectors of 45° - 200 random images for augmentation - 14400 images, 80% with changing backgrounds - 75 % for training and 25% for testing	- Improved the performance of object recognition and roughly pose estimation compared to CNN such as VGG16
1.2	- How can we improve the performance of the HMM for object recognition? - What happens if we trained the model on a clear dataset and tested on noisy dataset such as occlusion?	- HMM can be improved by integrating it with the active vision (AV-HMM-CEDD). - We use pretrained CEDD descriptor	**COIL-100 Dataset** - 100 objects - 72 images (5° at the same elevation) - LSTM was trained on the whole COIL dataset and tested its recognition performance on the partially COIL occluded dataset **ALOI-1000 Dataset** - 1000 different small objects - 72 images (5° at the same elevation)	- Lightweight solution - Improved the performance of HMM - Handle noisy testing images

Table 1.2: Summarizing research questions and datasets information of Thesis II.

Thesis	Task to be solved	Proposed solution	Dataset	Advantages
2.1	- How can we recognize the defects of traffic signs with high accuracy?	- A new siamese neural network architecture called fusioning convolutional siamese neural networks (FCSNN), was proposed to recognize the defects of a large number of classes of traffic signs.	**Traffic Signs with Defects (TSD Version I)** - Collected real world images captured from dash car - 21 different traffic signs - 7 different defect classes - Dataset contains 6016 images used for training - For testing, a dataset containing 490 images of untrained new traffic signs was used.	- Good recognition rate for faded defect class - Generalize the performance for the new traffic signs never seen by the network
2.2	- How can we improve the performance of siamese neural networks?	- A new mechanism to combine the confidence values of siamese networks with SVM with the help of support set images.	**Traffic Signs with Defects (TSD Version II)** - Collected real world images - 66 different traffic signs - 4 different defect classes - 3792 for training and 669 for testing	- Better performance compared to FCSNN.

Table 1.3: Summarizing research questions and datasets information of Thesis III.

Thesis	Task to be solved	Proposed solution	Dataset	Advantages
3.1	- Visual inspection is a key component in automated production industry for recognizing surface defects. - In case of very few-shots for the new classes, incremental learning becomes even more difficult. - Catastrophic forgetting problem - Time and memory complexity requirements	- A new architecture to combine EfficientNet with Randomized Networks (EffNet+RC) for classification and few-shot learning approaches, as well as the issue of continuous learning for steel surface defects	**NEU surface defect database** - 6 different surface defect classes - Images size 200x200 pixels - Collected from hot-rolled steel strips - 1800 images and 300 samples per class - 80% for training and 20% for testing **Xsteel surface defect dataset (X-SDD)** - Data-set contains 1360 images - 7 different defect classes - Image size 128x128 pixels - 70% for training and 30% for testing	- One model to handle the problems of the low number of available shots of new classes, the catastrophic forgetting, and the long training time required for re-training. - In case of classification of steel surface defects outperforms all other known approaches, with an accuracy 100%
3.2	- How can we utilize the siamese neural network for zero shot learning in case of steel defects?	- A deeper architecture of FCSNN proposed and utilized for zero-shot learning.	**Zero shot in case of NEU dataset** - 3 classes used for training and 3 classes used for testing **Zero shot in case of X-SSD dataset** - 4 classes used for training and 3 classes used for testing	- Could be applied to new classes never seen for the network

1.6 Author's Contributions

This dissertation focuses on introducing different approaches to solve the problem of recognizing objects and their defects. The main contributions and their associated scientific results are presented below:

- Presented a novel idea to integrate neural networks with hidden Markov models (HMM) and inertial measurement units (IMUs) data to improve the performance for multi-view object recognition and pose estimation, (**see Chapter 2**).

- Improved the performance of the HMM based solutions by integrating active vision (AV) and information fusion from sequential multiple shots in a framework called AV-HMM-CEDD. This helps the recognition of 3D objects even if they are partially occluded. Additionally, the proposed AV-HMM-CEDD framework is computationally lightweight, requires limited memory and can incorporate other classifiers, not only the presented CEDD, (**see Chapter 2**).

- Proposed a new siamese neural network architecture called fusing convolution siamese neural network FCSNN, to recognize the defects of traffic signs. Moreover, it is possible to use it to recognize defects in untrained traffic signs classes, (**see Chapter 3**).

- Introduced a new mechanism to improve the performance of FCSNN by combining the output confidence values of siamese networks with a support vector machine (SVM) with the help of support set images. The advantage of our approach, compared to the concept of an ensemble of networks, is that only one network is to be trained and maintained, (**see Chapter 3**).

- Proposed a new architecture to combine EfficientNet deep neural networks

with randomized classifiers for the solution of the following problems: few shots learning especially for new classes, catastrophic forgetting of known information when tuning for new artifacts and the long training time required for re-training or fine-tuning existing models. Additionally, to deal with zero shot learning, I proposed deeper architecture of the siamese network and utilized it for zero-shot learning, (**see Chapter 4**).

Chapter 2

Object Recognition Techniques using Deep Neural Networks and HMM

2.1 Introduction

While there is a long history of optical object recognition only in the last few years, we have seen significant improvements with the evolution of neural networks. Recent developments in deep learning frameworks based on deep neural networks (DNN) significantly increased the accuracy of machine learning algorithms. However, the benefits of DNN-based DL are significantly diminished when there are severe data limitations and/or when there are no relevant problems for transfer learning. In [27] as an example, pure data-driven auto-encoders dealing with high-dimensional raw input data would require significant amount of data for effective operation even when stacked shallow auto-encoders are employed.

Real-world data are usually collected from diverse domains or obtained from various feature extractors. This data can be utilized to improve the performance

of recognition. Multi-view object recognition focuses on using several images from different views for performance improvement. But, the question is, how can we integrate these data effectively? Statistical models such as HMMs are one of the techniques that could handle this problem. HMMs are frequently utilized in a variety of recognition issues, such as speech, musical sound, human activity, or object recognition. The barrier to use HMMs in object recognition was always the real lack of ordered sequential information. It is a natural assumption that multiple shots can decrease ambiguity and if those shots are from different directions, the amount of information gathered from the object also increases. In this later case, we soon arrive at relative pose estimation where the appearance of the 3D object corresponds to its relative pose. Due to natural ambiguities, such as noise, occlusion and geometrical distortions, this estimation can be ill-posed.

Current multi-view approaches comprise conventional computer vision (CV) or DNNs methods such as [28, 29]. LSTM (long short-term memory) networks are popular techniques to include temporal domains in the deep neural network frameworks. Additionally, many hybrid approaches try to combine the best features of traditional computer vision techniques with statistical models and DNNs rather than choosing just one of them. In [30], two alternative vision techniques (traditional machine vision and modern deep learning techniques) were utilized for the problem of object recognition for a mobile robot in an indoor environment. The first approach uses HOG descriptor with SVM classifier as a traditional machine vision model while the second approach uses Tiny-YOLOv3 as a modern deep learning model.

Unfortunately, when considering lightweight solutions, which can be crucial for mobile wearable technology, the temporal combination of multiple views is much less investigated.

In this chapter, we present a new approach to combine the output of the CNN (confidence values) with HMM. HMM and IMUs data are used to improve the performance of the convolutional neural networks for object recognition and roughly pose estimation. Additionally, we propose a lightweight variational approach, which can be combined with any single shot detection technique, including DNNs. The proposed models also implement information fusion since besides 2D images they use IMUs sensors for the estimation of change in the relative orientation of the camera and active vision to improve the performance of HMMs.

2.2 View-Centered Approach

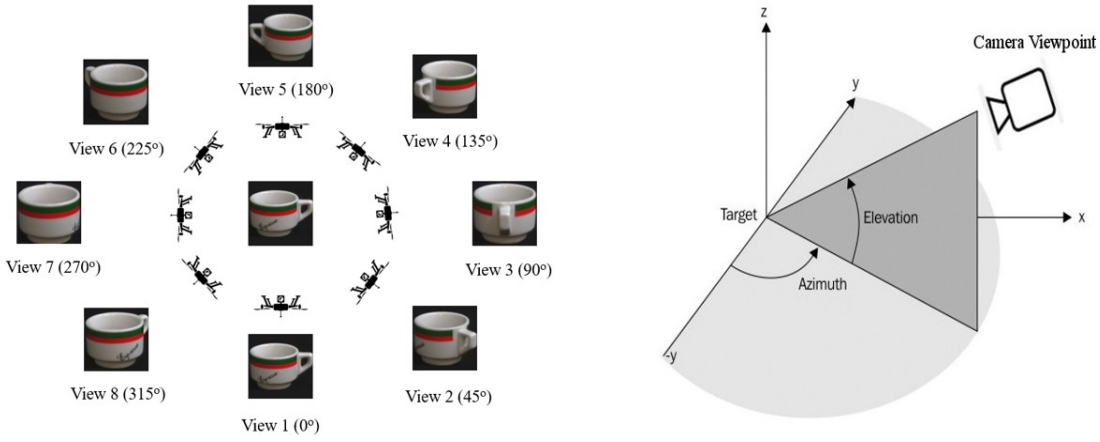

Figure 2.1: Model generation setup with target object in the centre.

View-centered representations model the outlook of objects as captured from different viewpoints. Figure 2.1 illustrates the view-centered approach where multi-view object models are built up from 2D images taken from different orientations, being the object of interest in the focus of the camera. To get a complete object model a larger number of different azimuth and elevation angles are required. How-

ever, for average applications the elevation can be limited (in our tests we used only one elevation angle typical for an object placed on the table).

2.3 Hidden Markov Model Explanation

Hidden Markov Models are statistical frameworks in which the system being modeled is assumed to behave as a Markov process with directly unobservable (hidden) states. HMM can be considered as the simplest dynamic Bayesian network. The logic behind HMMs was already introduced in the late 1960s and early 1970s in the works of [31, 32]. HMM is a probabilistic model, which originates from discrete first-order Markov processes.

In this section, we briefly overview the theoretical background of HMMs and some related problems. Let $S = \{S_1, \cdots, S_N\}$ denote the set of N hidden states of the model. In each t index step this model is described as being in one $q_t \in S$ state, where $t = 1, \cdots, T$. Between two steps the model undergoes a change of state according to a set of transition probabilities associated with each state. The transition probabilities have first-order Markov property, *i.e.*

$$P(q_t = S_i | q_1, \cdots, q_{t-1}) = P(q_t = S_i | q_{t-1}) \tag{2.1}$$

Furthermore, we only consider the processes, where the transitions of equation 2.1 is independent of time. Thus, we can define the set of transition probabilities in the form

$$a_{ij} = P(q_t = S_i | q_{t-1} = S_j) \tag{2.2}$$

where i and j indices refer to states of HMM, $a_{ij} \geq 0$, and for a given state $\sum_{j=1}^{N} a_{ij} = 1$ holds. The transition probability matrix is denoted by $\mathbf{A} =$

$\{a_{ij}\}_{1\leq i,j\leq N}$. We also define the initial state probabilities:

$$\pi_i = P(q_1 = S_i) \tag{2.3}$$

and $\pi = \{\pi_i\}_{1\leq i\leq N}$. Now we extend this model to include the case, where the observation is a probabilistic function of each state. Let $O = \{o_1, o_2, \cdots, o_T\}$ denote the set of observations. The emission probability of a particular o_t observation for state S_i is defined as

$$b_i(o_t) = P(o_t|q_t = S_i) \tag{2.4}$$

$\mathbf{B} = b_i(o_t)$, $1 \leq i \leq N$ is a sequence of observation likelihoods, also called emission probabilities, each expressing the probability of an observation o_t being generated from a state i. The complete set of parameters of a given HMM is described by $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$. A more comprehensive tutorial on HMMs can be found in [33].

2.4 Datasets

The following datasets were used to train convolutional neural networks, generate the HMM models and run the different tests.

2.4.1 COIL-100 Dataset

The COIL-100 dataset [14] includes 100 different objects each with 72 images taken by 5° at the same elevation. We evaluated retrieval with clear and heavily distorted queries using Gaussian noise and motion blur. The imnoise function of Matlab, with standard deviation $sd = 0.012$, was used to generate additive Gaussian noise (GN) while motion blur (MB) was made by fspecial with parameters $len = 15$, and

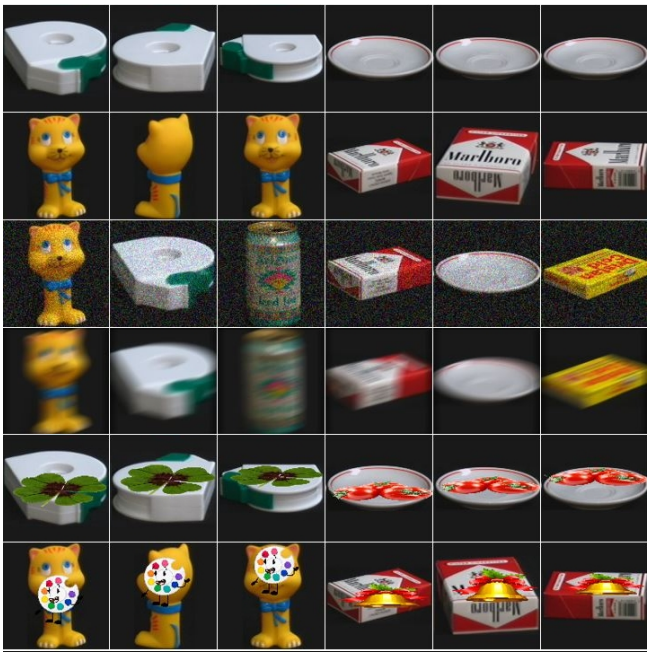

Figure 2.2: First two lines: Clear samples from COIL-100. 3^{rd} line: Example queries loaded with Gaussian additive noise. 4^{th} line: Example queries loaded with motion blur. 5^{th} and 6^{th} lines: Occluded examples.

angle $\theta = 20°$. Some examples of the queries are shown in Figure 2.2. To simulate real-life scenarios, we created the occluded COIL-100 dataset containing the same 100 objects, but with partial occlusion over the object areas (for illustration see Figure 2.2).

2.4.2 COIL-40 Dataset

We have chosen 40 objects from the COIL-100 dataset for one of the experiments, see Figure 2.3, for sample images. Each object was represented with 8 poses by sectors of 45°. Images are originally with black background but to be more realistic we have given different backgrounds, selected from 200 random images, so the original COIL image covers around 25% of the area of 128 × 128 pixels, (see Figure 2.3 bottom line). We believe that a small adjacent black area around the objects does not distort the results since it appears in all objects and gives no advantage to any

Figure 2.3: Top line: example objects from the COIL-100 dataset. Bottom line: objects with different backgrounds.

of the methods. Thus, we got 2880 images (40×72) directly from COIL-100 and 11520 from augmentation. The dataset was cut into training and testing parts so no queries of the experiments could exactly match those images used to train the CNN. Dataset was divided into 75 % for training and 25% for testing.

2.4.3 ALOI-1000 Dataset

Figure 2.4: First two lines: Clear samples from ALOI-1000. 3^{rd} line: Example queries loaded with Gaussian additive noise. 4^{th} line: Example queries loaded with motion blur.

The ALOI-1000 dataset includes 1000 different small objects recorded against a black background. Each object was recorded by rotation in the plane at 5° steps. For

evaluation under different conditions, we used the same distortion settings, including occlusions, as described for the COIL-100 dataset. Please note, that while the resolution of images in COIL is 128×128 it is 384×288 for ALOI. (This explains the less visible Gaussian noise and motion blur in Figure 2.4).

2.5 Improving Object Recognition of CNNs with Multiple Queries and HMMs

There are clear trade-offs between traditional CV and deep learning-based approaches. Classic CV algorithms are well-established, transparent, and optimized for performance and power efficiency, while DL offers greater accuracy and versatility at the cost of large amounts of computing resources. Hybrid approaches combine traditional CV and deep learning approaches and offer the advantages of both methodologies. Also, the fusion of machine learning algorithms and deep neural networks have become very popular.

In this section, we combine neural networks with hidden Markov models for multiview object recognition. While convolutional neural networks are very efficient in object recognition there is still a need for improvements in many practical cases. For example, if the performance from single images is not satisfactory or the object localization is not solved with the neural network then information fusion from several images and from inertial sensors can still help a lot to improve the recognition rate. In our use case, we are to recognize objects from several directions with the VGG16 network. We assume that no localization of objects is possible on the images due to the lack of bounding box annotations, we have to recognize the objects even if they occupy only about 25% of the field of view. To overcome this problem, we propose to use a HMM approach where the consecutive queries, shots taken

from different viewing directions, are first evaluated with VGG16 and then with the Viterbi algorithm. The role of the later is to estimate the most probable sequence of poses of candidates (from the predefined 8 horizontal views in our experiments), thus we can select the most probable object. The approach, as evaluated with different numbers of queries over a set of 40 objects from the COIL-100 dataset, can result in a significant increase of hit rate compared to one-shot recognition or to combining individual shots without the HMM model.

2.5.1 HMM Object Models

An HMM is defined by:

- the set of N possible hidden states $S = \{S_1, ..., S_N\}$,
- transition probabilities between states S_i and S_j, (see Eq. 2.2),
- emission probabilities based on observations, $P(o)$, (see Eq. 2.4),
- initial state probabilities π_i.

The observation of objects with multiple views is a process where in each t^{th} step this model is in one $q_t \in S$ state, where $t = 1, ..., T$. To achieve object retrieval will need to build HMM models for all elements of the set of objects (M) where the states correspond to different poses. Then, based on the sequence of observations, we find the most probable state sequence for all object models. The state sequence among these objects, which has the maximum probability, will belong to the object being retrieved.

2.5.2 Object Views as States in a Markov Model

[34] showed that the states can be considered as the 2D views (poses) of a given object model. Observations of these (hidden states) can be easily imagined as the camera is targeting towards an object from a given elevation and azimuth. In the experiments, they used static subdivision of the circle of 360° into 8 uniform sectors 45° each at the same elevation. The initial state probabilities was defined by $\pi = \{\pi_i\}_{1 \leq i \leq N}$ based on the opening angle of the views:

$$\pi_i = P(q_1 = S_i) = \frac{\psi(S_i)}{360} \tag{2.5}$$

where $\psi(S_i)$ is the angle interval (given in degree) of the aperture of state S_i.

2.5.3 State Transitions

Between two steps the model can undergo a change of states according to a set of transition probabilities associated with each state pair. In general, the transition probabilities are defined in Eq. 2.2.

The transition probability matrix is denoted by $\mathbf{A} = \{a_{ij}\}_{1 \leq i,j \leq N}$, where $a_{ij} \geq 0$, and for a given state $\sum_{j=1}^{N} a_{ij} = 1$ holds.

Building a Hidden Markov Model means the definition of hidden states and learning its parameters (π, $\mathbf{A}$, and emission probabilities introduced later) by examining typical examples. However, the problem doesn't allow such a training process: the probability of going from one state to another severely depends on the user's behavior. Contrary, [34] computed transition probabilities directly based on geometric probability as follows.

First define $\Delta_{t-1,t}$ as the orientation difference between two successive observations

(o_t and o_{t-1}) where α defines the orientation of the current observation:

$$\Delta_{t-1,t} = \alpha(o_t) - \alpha(o_{t-1}). \tag{2.6}$$

Now define R_i as the aperture interval angle belonging to state S_i by borderlines:

$$R_i = [S_i^{min}, S_i^{max}[. \tag{2.7}$$

where S_i^{min} and S_i^{max} denote the two (left and right) terminal positions of state S_i. The back-projected aperture interval angle is the range of orientation from where the previous observation could originate:

$$L_j = [S_j^{min} - \Delta_{t-1,t}, S_j^{max} - \Delta_{t-1,t}). \tag{2.8}$$

Now, to define the transition probability of coming from state S_i, they computed the ratio of opening angles of the intersection L_j and R_j and of the opening of L_j:

$$a_{ij} = P(q_t = S_j | q_{t-1} = S_i) = \frac{\alpha(L_j \cap R_i)}{\alpha(L_j)}. \tag{2.9}$$

For illustration, see Figure 2.5.

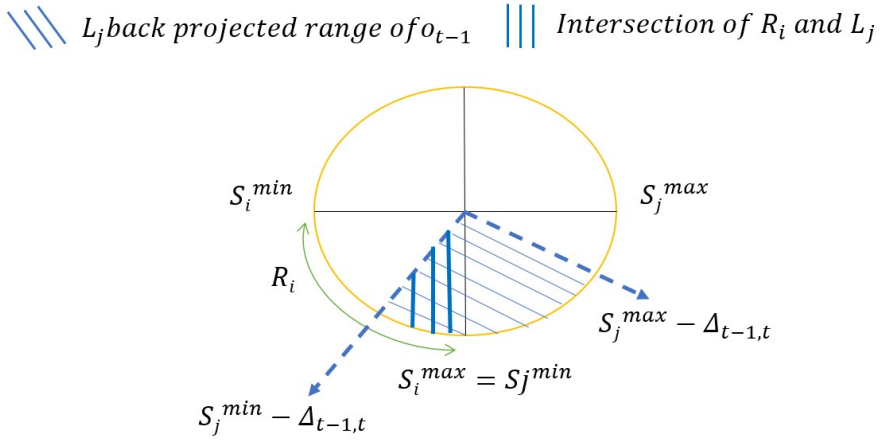

Figure 2.5: Geometrical explanation of transition probabilities.

2.5.4 Recognition of Single Objects from Multiple Views

Let $O = \{o_1, o_2, \cdots, o_T\}$ denote the set of observations. The emission probability of a particular observation o_t for state S_i is defined in Eq. 2.4.

[35] showed that the CEDD (Color and Edge Directivity Descriptor) [36] is a robust low dimensional descriptor for object recognition, where the Tanimoto coefficient can be used to generate simultaneously values interpreted as probabilities. In our use case, we utilized VGG16 as a global object classifier instead of CEDD's and used confidence values extracted from VGG16 as probabilities.

2.5.5 Proposed Method

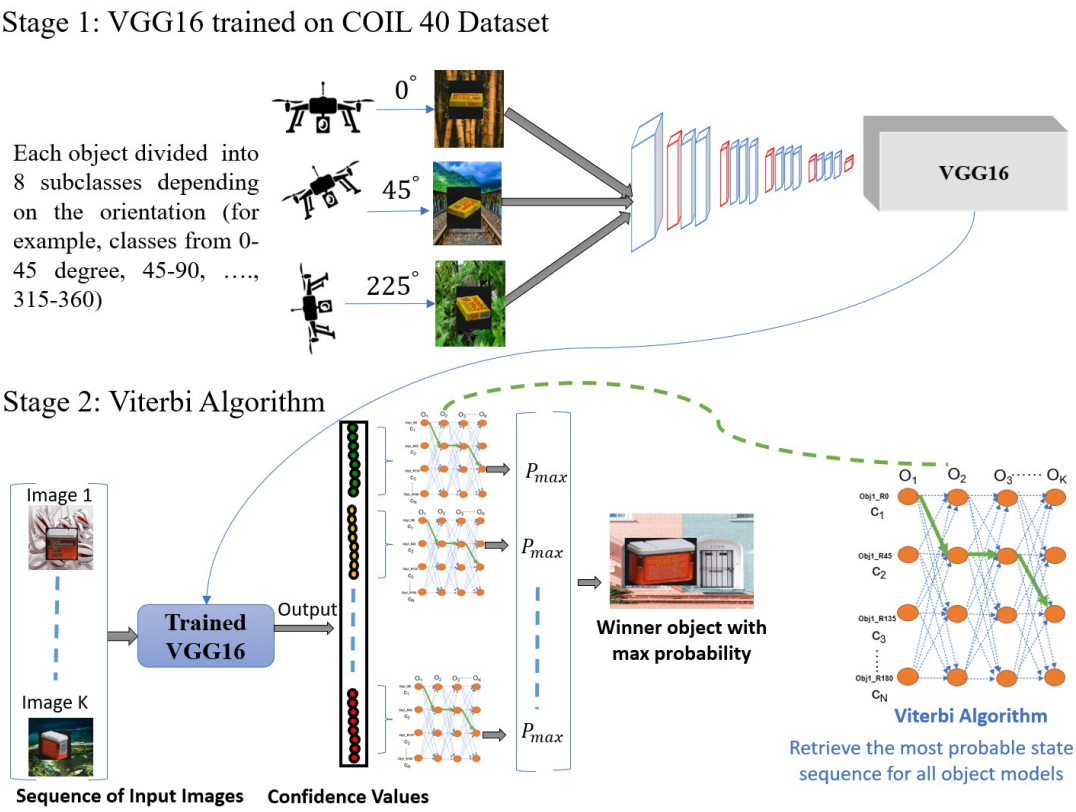

Figure 2.6: Proposed architecture to combine CNNs with HMM model.

Our proposed architecture consists of two stages, (see Figure 2.6). In stage one, we used the COIL-40 dataset for training VGG16. Dataset was divided into 75 % for training and 25% for testing. In stage two, the states can be considered as the 2D views (poses) of a given object model. Figure 2.6 shows that, for each view, we compute the confidence values from the pre-trained VGG16 model on the COIL-40 dataset. These confidence values will be integrated in HMM. The initial probabilities of these states were computed by Eq. 2.5 and the transition probabilities computed by Eq. 2.9. The emission probability of a particular observation o_t for state S_i is defined by Eq. 2.4.

Now, we use VGG16 as a global object classifier. We can run the same CNNs for all queries generating confidence values where we can run the Viterbi algorithm to combine the values of 2.4, 2.5 and 2.9 to get the most probable state sequences. Then we choose the object with the highest probability value as the winner.

2.5.6 Experimental Results

A single VGG16 network was used to recognize the 320 (40×8) classes: the network was pre-trained with ImageNet [17] images. We did not refine the feature extraction layers of the network, only the 4 end layers, responsible for classification, were replaced and re-trained. During training image rotation, shift, shear, zoom, and horizontal flip were applied as further augmentation.

To illustrate the gain achieved with the HMM models we also computed the hit rate for the method where the multiple queries were analyzed independently, and the results were ranked according to the average probabilities by objects (averaging the highest probability of each object for the given number of queries). This method is referred to as the average of multiple CNN detections. We evaluated the hit rate for 1, 2, 3, and 4 queries; the average results for the 40 objects are shown in Figure 2.7

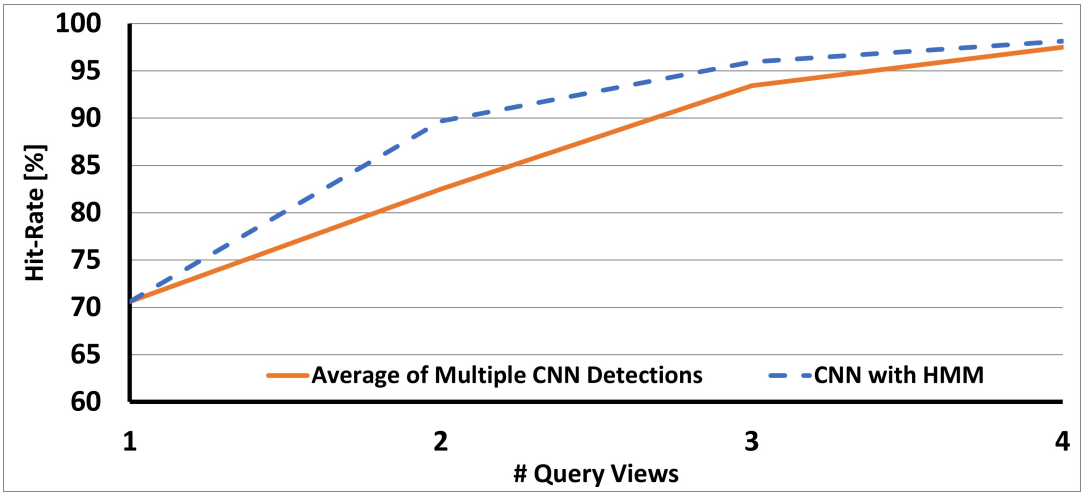

Figure 2.7: The average Hit-Rate for 40 objects with different number of queries.

based on several random tests with each object as a query. It is not surprising that as we increase the number of queries the Hit-Rate increases monotonically. It is clearly visible that for multiple queries the proposed method outperforms the other, although neither of them could reach 100%.

2.6 Hidden Markov Models Based on Convolutional Neural Network for Pose Estimation

The recognition of 3D objects is an elementary problem in many application fields such as robotics, autonomous vehicles, or augmented reality. However, to interact with the objects of the environment, not only specific or generic object recognition is inevitable, but the determination of their pose is also essential. Pose estimation is also a fundamental problem in computer vision and a large number of algorithms have been proposed for the various conditions and applications. In recent years, the state-of-the-art of convolutional neural networks, like Regional CNNs [37], Yolo [38], Mask R-CNN [39] and Single Shot Detectors [40], have been proven to be very effi-

cient for object detection and recognition in RGB and depth images, however these CNNs do not provide us with straightforward object pose estimation. The problem of the estimation of the 6-DoF object pose was recently attacked by different CNN approaches. Classical approaches can be grouped [41], as direct linear transformation, Perspective n-Point, and a priori information estimators; they all suffer from the problem of efficient feature selection, correspondence generation and outlier filtering. Contrary, CNNs based methods have the great advantage to learn the combination of the best possible features and classifiers or regressors. Partially as a result of the Amazon Picking Challenge [42], the interest in object manipulation has increased recently leading to the development of several 6-DoF object estimation methods. Many of these methods, such as PoseCNN [43], SSD-6D [44], Real-Time Seamless Single Shot 6D Object Pose Prediction [45], BB8 [46], use CNNs to estimate pose with high accuracy of known objects in cluttered environments. It is well-known that the general disadvantage of neural network-based methods is the dependency on the training data and the utilized training methods. For example, in paper [47], the performance drop caused by missing object labels is analysed. Unfortunately, the generation of training data is typically costly whether it is based on real or synthetic data, especially if the pose is to be represented [48]. We have previously shown that HMMs can improve the recognition of objects from a sequence of images when global classifiers are utilized [49]. Since our proposal utilized orientation sensors it is straightforward to investigate whether we can improve object recognizers (such as CNNs) in pose estimation. In this section, we show that using a general object classification network (namely VGG16), the temporal inference generated by the HMM can significantly increase the roughly pose estimation possibilities.

2.6.1 Proposed Method

We followed the approach in Section 2.5.5, A single CNN is to recognize an object and its pose through several observations then a statistical framework is applied to evaluate the result of inferences and to make the final object recognition and pose estimation [50]. We have chosen a well-known neural network, often used as a backbone of more complex architectures, namely VGG16 [51]. We don't deal with the localization of the object within the image frame. I.e., it gives no big stress for the annotation procedure to generate training data but makes it hard work for the processing framework to achieve good pose estimation.

During the recognition of consecutive queries, shots taken from different viewing directions are first evaluated by VGG16 inference resulting in confidence values. We assume that the relative pose changes between the shots are recorded by easily available IMUs sensors (such as those built into most mobile phones).

Using the image shots, the pre-built object models, the trained VGG16 networks, and the change in orientation between shots we use an HMM framework to evaluate the image sequences and to determine the most probable object and its pose series generating the observations. Since the order of sequential poses (the actual changes of relative viewing directions) is determined by the behavior of the camera (or with other words by the user) it cannot be generally modelled in the model to determine the actual transition probabilities. What we can do is to measure the real change in relative poses query by query, with the help of IMUs sensors, and use geometric probabilities to evaluate the chance of going from one state to another. For this resolution of the problem of computing state transitions, please see subsection 2.5.3. To compute the orientation error, we summarized the steps in Algorithm 1.

Algorithm 1: Pose estimation based on CNN with HMM and IMUs Data

Input: Sequence of observations

Output: Pose estimation computed based on most probable sequence obtained from viterbi algorithm

```
Each object is divided into 8 subclasses by sector of 45°
Trained CNN model (VGG16) on 8 subclasses for each object
Run Viterbi algorithm for each object based on the confidence values
  extracted from VGG16
If object not recognized go to END
while Object is recognized do
    Compute orientation for each view in the most probable state sequence
    Compute the smallest orientation difference of sectors for each view
    Pose estimation (P_e) ≡ Average the smallest orientation difference of
     the most probable state sequence
    return P_e
END
```

2.6.2 Tests and Evaluations

A single VGG16 network was used to recognize all 320 (40 objects $\times$ 8 poses) classes. We follow the same setting and architecture as in Section 2.5.3. To get a general overview of the performance we computed the pose error by averaging the orientation error for each object and each pose estimated in 8 independent random experiments. As one could expect the error may depend on the number of observations (i.e. the number of queries). As Figure 2.8 shows, increasing the number of queries results in the decrease of average pose error from 67.18° to 44.78°.

As a reference, we computed the average error of the VGG16 network illustrated by orange in Figure 2.8. These values are ranging from 67.18^{o} to 63.59^{o} significantly higher than the VGG+HMM technique.

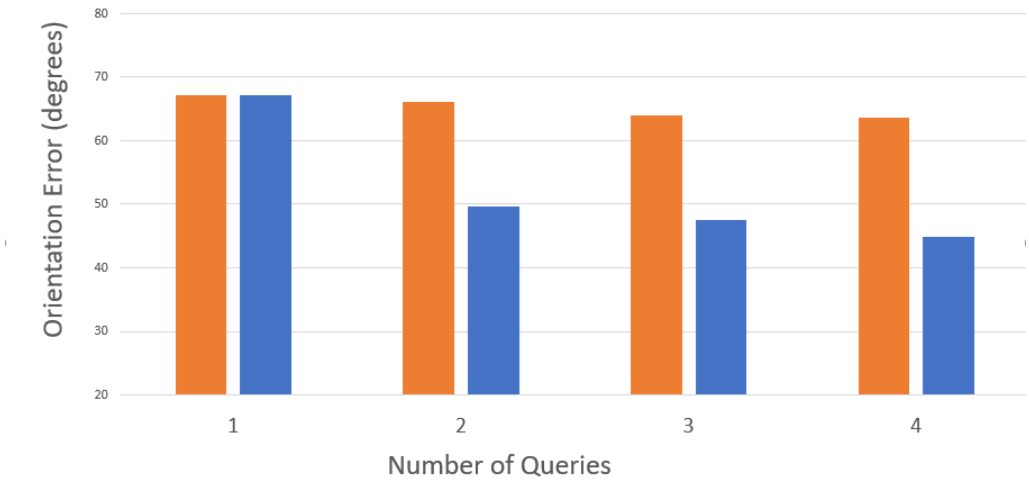

Figure 2.8: Average orientation error at different number of queries for VGG16 only (orange) and VGG+HMM (blue).

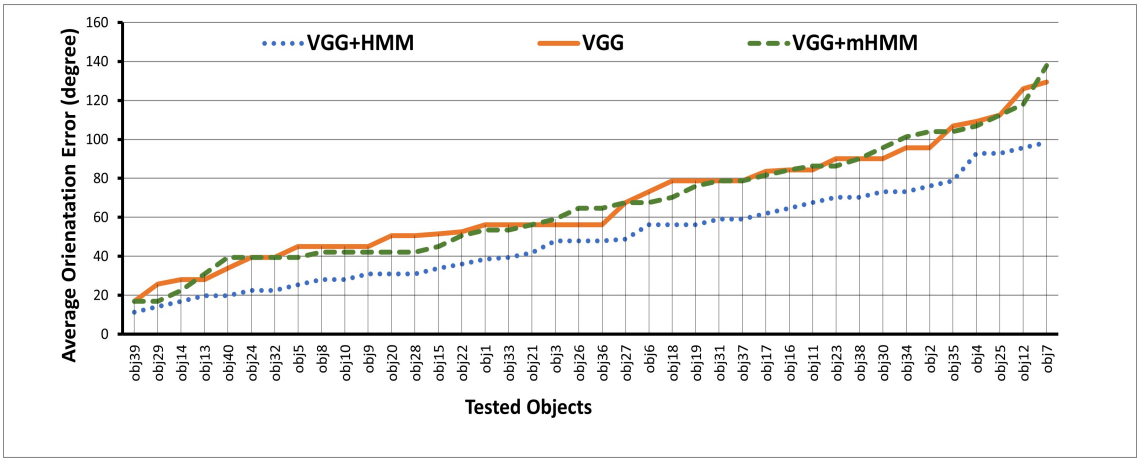

Figure 2.9: Average orientation error for each object, in case of two queries, for VGG16 only (orange), VGG+HMM (blue), and VGG+mHMM with constant transition probabilities (green dotted).

To highlight the information added by the orientation sensor we made tests where the transition probabilities were set constant and computed the orientation error for each object, in case of two queries. This is named VGG+mHMM and is shown by green dotted lines in Figure 2.9. There is no significant difference between VGG16 and VGG+mHMM as expected.

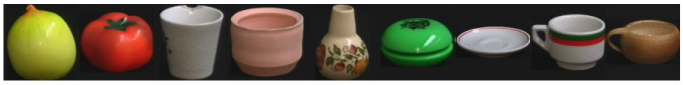

Figure 2.10: Mostly rotational invariant objects.

Figure 2.11: Not rotational invariant objects.

2.6.2.1 Orientation Error without Rotational Invariant Objects

The COIL-40 dataset contains rotational invariant and not rotational invariant objects as appeared in Figures 2.10 and 2.11, respectively.

When we computed the rough estimation for the pose estimation without the rotational invariant objects, the error decreased from 45^{o} to 40^{o}, (see Figure 2.12). Additionally, Figure 2.13, shows that for some not rotational invariant objects, there

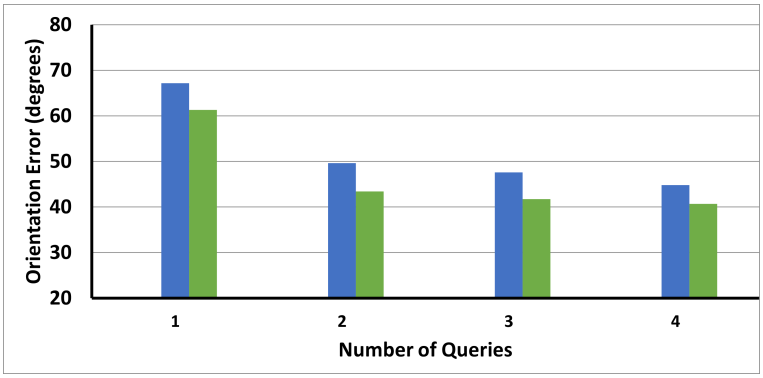

Figure 2.12: Average orientation error at different number of queries for VGG+HMM with rotational invariant objects (blue) and VGG+HMM without rotational invariant objects (Green).

are still some views from different sectors that seem to be very similar affecting the estimation of the pose.

Figure 2.13: Not completely rotational invariant objects with rotations resulting in very similar views.

2.7 Active Multiview Recognition with Hidden Markov Temporal Support

To improve the performance of HMM, active multiview object recognition focusing on the directional support of sequential multiple shots will be utilized. Inertial sensors were used to estimate the orientation change of the camera and thus to

estimate the probability of relative poses similar to Section 2.5.3. With the help of relative orientation change, we can compute transition probabilities between possible poses and can use a hidden Markov model to evaluate state (pose) sequences.

Furthermore, we can plan our next viewing position to minimize the risk of misclassification, resulting in higher overall recognition rates. Besides giving the theoretical details, we use two datasets to illustrate the performance of our model through several tests including occlusion, blur, Gaussian noise, and to compare to a solution with a long short-term memory network (LSTM).

2.7.1 Recognition of Objects from Multiple Views by Weak Global Classifiers

To create the HMM models for all elements of the set of objects (M) and evaluate the state transition between the states we follow the same strategy in Section 2.5.3. The emission probability of a particular observation o_t for state S_i is defined by Eq. 2.4.

But now to show the universally of the framework, we use the combination of the CEDD descriptor [36] and Tanimoto coefficient to approximate the emission probabilities of states. CEDD's advantage is that it uses only a short vector (length of 144) as a descriptor, but it is global and less robust considering its recognition abilities under various circumstances. More sophisticated (but also computationally expensive) single shot recognition techniques can also be used within our frameworks such as VGG [51], SSD [40], or Yolo [52].

Emission probability of Eq. 2.4 can be given as:

$$b_i(o_t) = \frac{\mathcal{T}(\mathcal{C}(S_i), \mathcal{C}(o_t))}{\sum_{j=1}^{N} \mathcal{T}(\mathcal{C}(S_j), \mathcal{C}(o_t))} \tag{2.10}$$

where $\mathcal{C}$ stands for the CEDD descriptor generating function and $\mathcal{T}$stands for the Tanimoto coefficient. Since each state of the object models can cover a large directional range, we will use the average CEDD vector, of available model samples within, to represent the whole state with a single descriptor. The sequence of retrieval lists, generated by independent queries, is evaluated by the Viterbi algorithm to combine the values of Eq. 2.5, Eq. 2.9, and Eq. 2.10 to get the most probable state sequences. To achieve object retrieval, we have to find the most probable state sequence $\hat{S}_k$ with the above steps for all possible candidate objects. To select the winner object $\hat{k}$, we have to compare the observations with the most probable state sequence:

$$\hat{k} = arg \max_{\forall k \in M} (\frac{\sum_{i=1}^{T} \mathcal{T}(\mathcal{C}(o_i), \mathcal{C}(\hat{S}_{k,i})}{T}) \tag{2.11}$$

where k denotes object k in M.

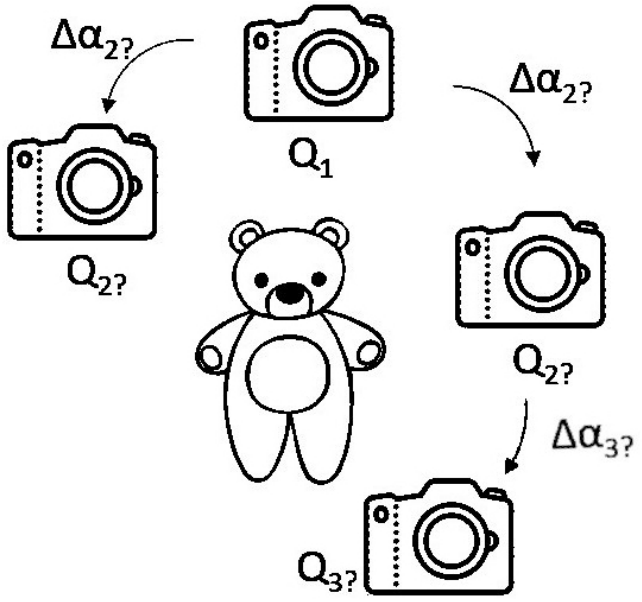

Figure 2.14: Object is recognized continuously in a sequence of queries. New queries (Q_2 and Q_3) are planned by the analysis of previous shot(s) (Q_1).

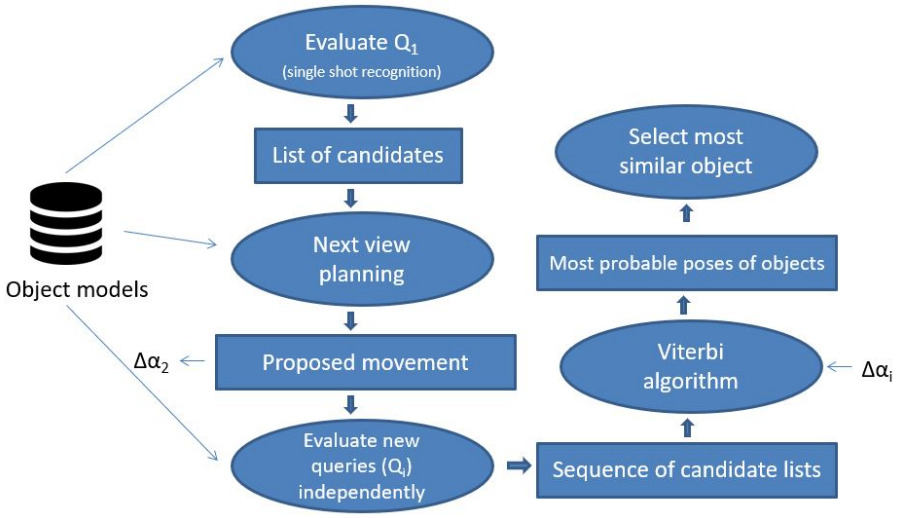

Figure 2.15: Overview of the proposed multiview method.

2.7.2 Active Recognition with HMM

The overview of our algorithm is shown in Figures 2.14 and 2.15. An object is being captured by several shots from different directions. These different views can be also considered as different relative poses. As the camera moves, we get the change of relative pose ($\Delta\alpha_i$) by the IMUs sensors. Each captured image is independently evaluated, and the probability of all possible objects is estimated.

Active recognition is a relatively old idea in pattern recognition, and it is typical to extend non-active methods. Without discussing such techniques, we refer the reader to the survey in [53]. Active vision systems can be classified, according to their next view planning strategy, into two groups:

1. systems that take the next view to minimize an ambiguity function;

2. systems incorporating explicit path planning algorithms.

We have chosen the first strategy and here we discuss a method that is very close to human's behavior to move around an object to become acquainted with its appear-

ance from different directions. Based on a rapid evaluation of the first observations, we hypothesize which objects have high probability and we plan the following movements to find those views that can reduce ambiguity. Now, based on the preliminary models, each object k will be represented with N_k descriptors computed as the average of descriptors within a given viewing range:

$$\tilde{c}_{k,i} = 1/N_k^i \sum_{l=1}^{N_k^i} c_{k,l} \tag{2.12}$$

where $c_{k,l}$ stands for the descriptors of object k within the interval i. The similarity between these average views can be computed with the Tanimoto coefficient and can be stored in a matrix S of size $NN_k \times NN_k$. After making the very first observations, we are to evaluate the retrieval list(s) $\mathcal{L}$, and as $\alpha(\tilde{c}_{k,i})$ provides the estimate of orientation for the most probable object k in state i, we can also compute the similarity of object views to the left (and to the right accordingly):

$$S_{left} = \sum_{\tilde{c}_j, \tilde{c}_l \in \mathcal{L}, j \neq l} \mathcal{T}(\tilde{c}_{j,left}, \tilde{c}_{l,left}) \tag{2.13}$$

where $\tilde{c}_{j,left}$ and $\tilde{c}_{l,left}$ are the next views left to the most similar views to the query being in the already existing retrieval list $\mathcal{L}$.

Finally, we should move the camera either to the left or to the right depending on the similarity of views of the possible candidates:

$$Decision = \begin{cases} \text{Move to left} & \text{if } S_{left} \leq S_{right} \\ \text{Move to right} & \text{if } S_{left} > S_{right} \end{cases} \tag{2.14}$$

resulting in the more discriminating direction. The performance of this *active* approach will be compared to the *non-active* recognition in Section 2.7.3.

2.7.3 Experiments and Evaluations

Variations in the datasets (see Figures 2.2 and 2.4) were used to show how our temporal methods can improve the performance of the weak classifiers under different circumstances. Since CEDD mainly relies on edge-like features, strong additive noise or (motion) blur can influence result.Charts are generated by taking the average of 10 experiments with randomly generated queries with all 100 and 1000 objects of COIL and ALOI datasets, (That is the total number of queries was the multiple of 11000).

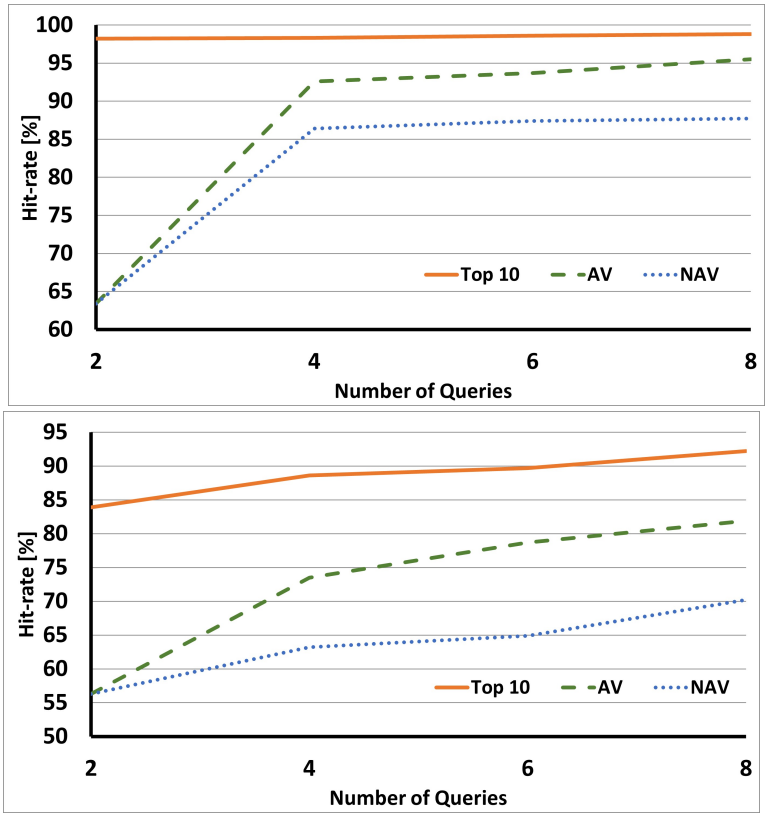

Figure 2.16: Comparison of non-active and active recognition when all queries are occluded. Top graph: COIL-100, bottom graph: ALOI-1000 datasets. Continuous curves show the GT being in the top 10 items of the retrieval list.

In all measurements, we see the advantage of using multiple queries: all curves monotonically increase as the number of queries increases. The first two charts

of Figure 2.16 help the understanding of our idea for active vision on some test data where all queries were occluded. The continuous curves show whether the ground truth (GT) objects are within the top 10 candidates of the retrieval lists ($\mathcal{L}$). Since our next view planning makes its decision based on the 10 most probable candidates of the first two retrieval lists, we expect to get results below this curve but above the non-active approach. We could measure performance gain over non-active recognition between 6.2% and 13.8% in these experiments.

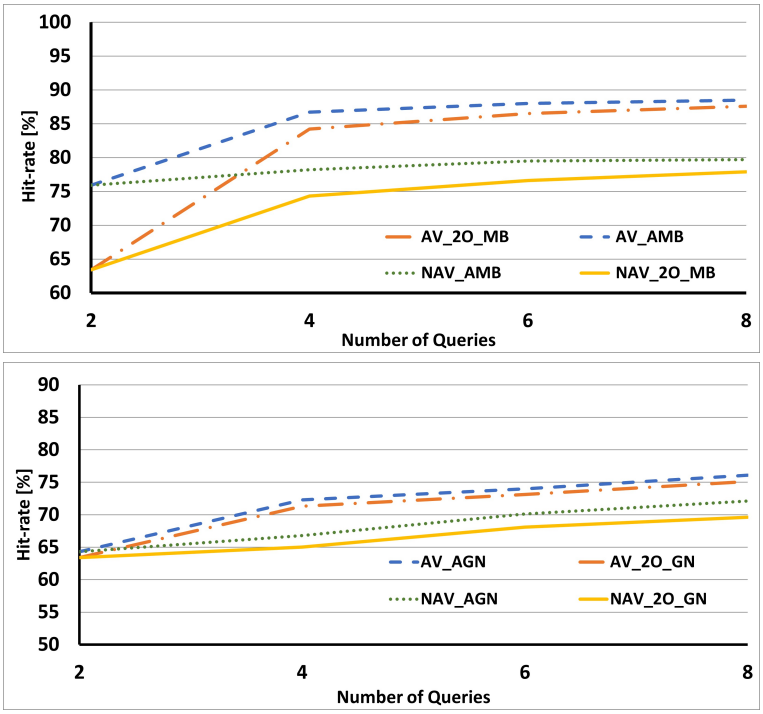

Figure 2.17: Comparison of active and non-active recognition on the distorted COIL dataset. First graph: motion blur, second graph: Gaussian noise.

Figure. 2.17 and Figure 2.18 show other experimental results regarding the COIL and ALOI datasets respectively. In these tests either all queries were loaded with Gaussian noise (GN) or motion blur (MB), or the first two queries were partially occluded while the remaining ones were loaded with MB and GN (these are denoted with 2O_MB and 2O_GN). In all cases the increase of the number of queries resulted

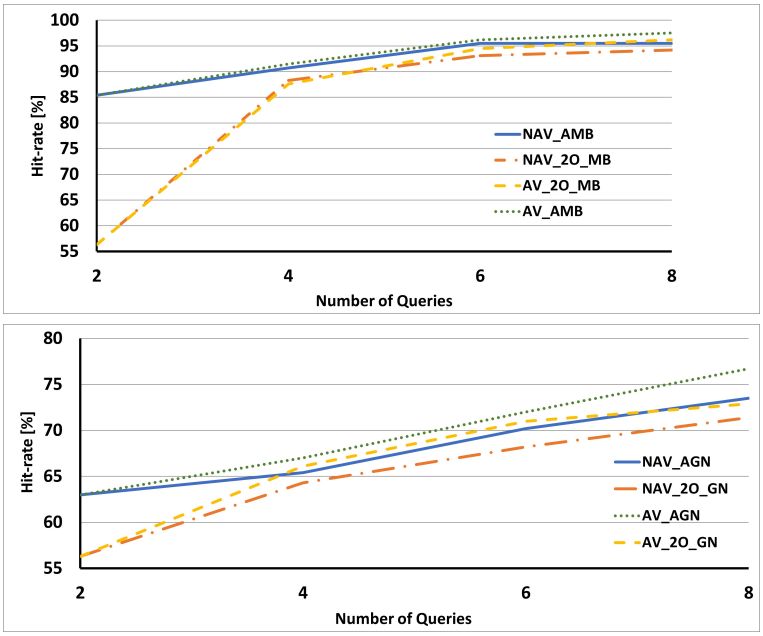

Figure 2.18: Comparison of active and non-active recognition on the distorted ALOI dataset. First graph: images with motion blur, second graph: images with Gaussian noise.

in higher Hit-Rate and active vision outperformed the non-active.

Table 2.1: Average Hit-Rates (%) in case of different query distortions applying 8 sequential queries.

	COIL-100		ALOI-1000	
	AV	NAV	AV	NAV
All queries occluded (AO)	95.5	87.7	81.9	70.2
2 qrs. occ. oth. GN (2O_GN)	75.1	69.6	72.9	71.4
All queries GN (AGN)	76.1	72.1	76.7	73.5
2 qrs. occ. oth. MB (2O_MB)	87.6	77.9	96.2	94.2
All queries MB (AMB)	88.5	79.7	97.5	95.5

For an alternative presentation of some parts of the above data we included a table (Table 2.1) of results for the 8 queries cases. It is clear to see that while in the case of the smaller dataset (COIL-100 with 100 object classes) there is a significant advantage of the active method, contrary, in case of a large number of object classes

in case of ALOI dataset, this decreased to around 1% in general. While this effect is natural, it is less significant in the case of good quality images as we can read from Figure 2.16 where only occlusion happened but no other type of noise.

2.7.4 About Space and Time Complexity

While any single shot feature extraction technique can be applied in the proposed framework, in this section, we used the very compact CEDD descriptor. It occupies 144 Bytes per image, while the orientation information requires not more than 4 Bytes. Running on plain CPUs (Intel Core i7 × 4MHZ), the memory and running time requirements are given in Table 2.2.

Table 2.2: Memory and running time requirements of the HMM and LSTM models.

Number of queries	2	4	6	8
Method		AV-HMM-CEDD		
Memory	30 KB	60 KB	89 KB	120 KB
Time (sec)	0.0127	0.0288	0.0385	0.0512
Method		ConvLSTM		
Memory	392 MB	787 MB	1.2 GB	1.6 GB
Time (sec)	0.0263	0.0422	0.0598	0.0751

2.7.5 An Alternative: ConvLSTM

Although traditional RNNs can easily learn short-term dependencies; they have difficulties to learn long-term dynamics as the gradients which are back-propagated can vanish or explode. Gradients are values used to update a neural network's weight. [54] LSTM is a type of RNN addressing these problems as the LSTM cells allow gradients to flow unchanged, to avoid the gradient vanishing problem during training, while learning both long-and short-term dependencies. LSTM solved this

problem by using three gates (Forget gate, Input gate and Output gate) to decide which data in a sequence is important to keep or throw away. LSTMs are efficient techniques for the sequential linkage of observation data. In computer vision they are mostly utilized for the processing of dynamically changing data such as motion behavior [55] and tracking of objects [56]. Not only temporal data can be processed by LSTMs: in [57], apple diseases and pests are detected. Here the purpose of LSTM was to combine the features of three deep models namely AlexNet, GoogleNet and DenseNet201. [58] applies a much more interesting approach to address action-driven weakly supervised object detection. The proposed temporal dynamic graph LSTM architecture recurrently propagates the temporal context on a constructed dynamic graph structure for each frame. That is, temporal action information patterns can help the recognition of visual objects. Similarly, approach [59] combines the output of independent detection but not with HMMs but with LSTMs called Association LSTM.

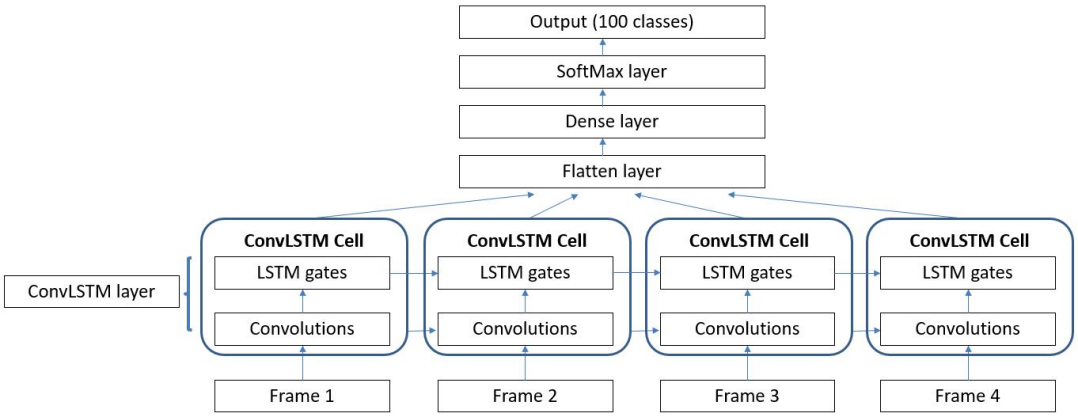

Figure 2.19: Overview of the tested ConvLSTM framework in case of four sequential queries.

To compare our active vision HMM model with one of the best known DNNs for time series, we implemented a so-called ConvLSTM network accepting several query

frames based on the technique given in paper [60]. The overview of the framework, after our modification, is illustrated in Figure 2.19. It can process query frames in a directional sequential order (either left or right), the 10 convolution kernels have size 3 by 3. It is known that DNNs are sensible for the training: high numbers of sample images are required under similar viewing conditions to those at inference. To accomplish this, either sophisticated augmentation techniques are required or large synthetic datasets are used relying on the CAD model of the objects.

For each view in the sequence of query, we created a ConvLSTM cell. ConvLSTM layers will do a similar task to LSTM but instead of matrix multiplications, it does convolution operations and retains the input dimensions. The extracted features from each cell were integrated in a flatten layer followed by a fully connected layer. Finally, a Softmax layer was applied to perform classification.

2.7.6 Comparison to LSTM

In our experiments, we trained the ConvLSTM on the whole COIL-100 dataset (7200 images divided into number of sequences of length N, $N = 2, 4, 6, 8$) and tested its recognition performance on the partially occluded version. We already showed the Hit-Rates of our proposed framework in Figure 2.16. For comparisons with LSTM Figure 2.20 gives the mAP (mean average precision) values. It can be interpreted that the HMM can handle the untrained occluded queries much better. The running time and memory usage of the LSTM model is given in Table 2.2. Please, also consider that the training took about half an hour for 100 epochs with an NVIDIA Quadro P6000 GPU with 24 GB RAM.

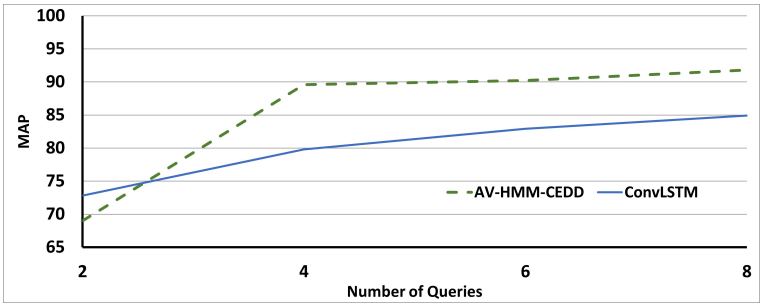

Figure 2.20: Comparison of active HMM and ConvLSTM on occluded queries.

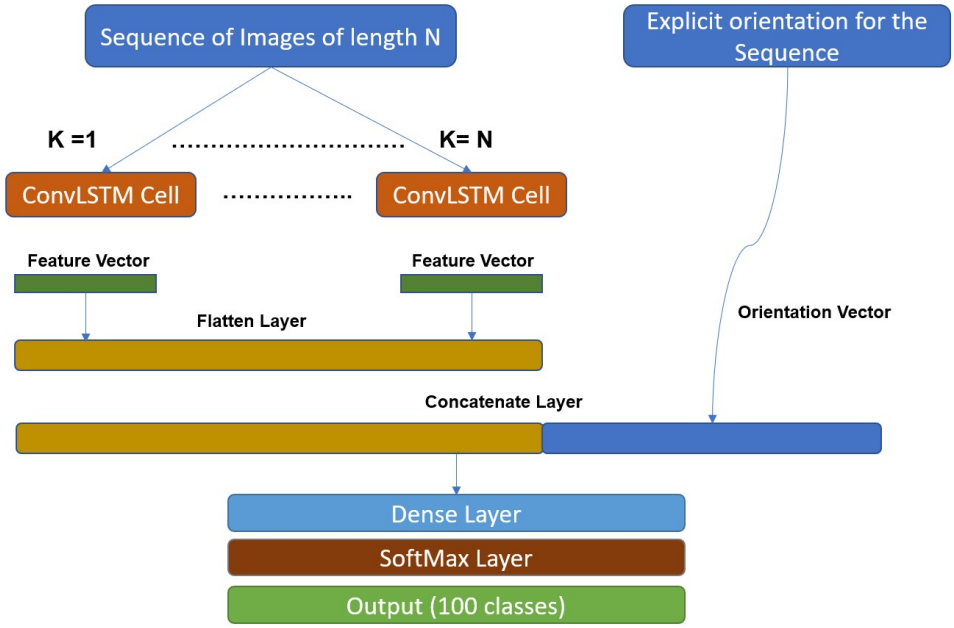

Figure 2.21: LSTM architecture with explicit pose.

2.7.7 Comparison to LSTM with Explicit Orientation

HMM with active vision utilized the pose explicitly to have the ability of choosing the most discriminant view for recognition (left or right). To compare our proposed method with LSTM with explicit pose, we created a new architecture for LSTM to accept two inputs (one for sequence of images and the other for explicit orientation pose). Figure. 2.21 illustrates this architecture. We used the same settings as in Section 2.7.6. Figure. 2.22 shows that mAP values at different numbers of queries (2,

4, 6, 8). It is obvious that LSTM with explicit orientation pose gives better results than LSTM with implicit orientation pose, but AV-HMM-CEED still overcomes both of them.

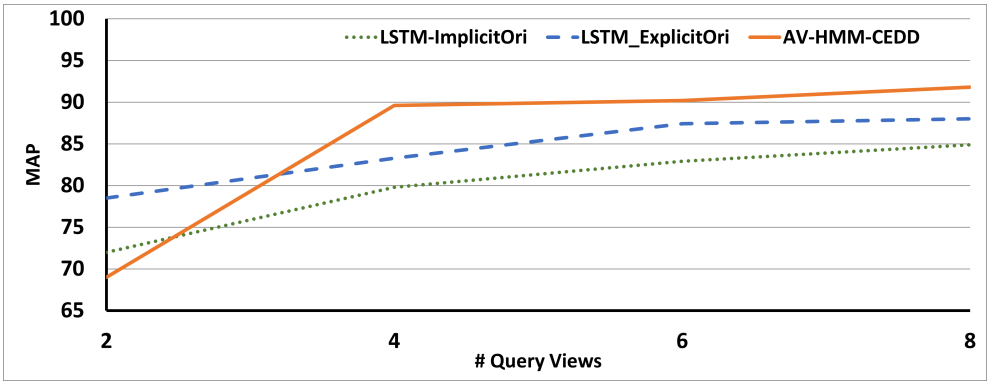

Figure 2.22: LSTM with explicit orientation results.

2.8 Summary

First, we combined neural networks with hidden Markov models for Multi-view object recognition. Our main contribution is to show that even if the neural networks could not be optimally trained and region-based detection is not possible, they can still be used for multiple-view object recognition with the help of information fusion. We show that relative pose changes can give enough information for the HMMs to estimate the most probable state sequences and thus finding the most probable object visible on the images. Consequently, we could recognize the object and roughly estimate its orientation. The performance improvement via information fusion is significant as shown in our experiments. The advantage of the method is that it can be used with any single shot recognition technique (not only the tested VGG16 network) and the usage of the IMUs information can easily support the HMMs which thus do not require any training. Second, we showed how active vision and information fusion can help the recognition of 3D objects in a HMM framework

if only weak classifiers are applied. The possible application of such approaches can be important in embedded systems or if sensors with limited resources are to be used for example in future's autonomous, wearables or IoT devices. The proposed AV-HMM-CEDD technique is computationally lightweight, requires small memory and can incorporate other classifiers, not only the presented CEDD. The effectiveness of the method was tested with a large number of experiments in various conditions and with comparisons with LSTM implementations.

This chapter was summarized in Thesis I, please see Chapter 5, Section 5.1.

Chapter 3

Detecting Traffic Sign Defects

The automotive industry is undergoing a paradigm change from human-driven vehicles to self-driving vehicles powered by artificial intelligence. A self-driving car (sometimes called an autonomous car or driver-less car) is a vehicle that uses a combination of sensors, cameras, radar, and artificial intelligence (AI) to travel between destinations without a human operator. To be considered as fully autonomous, a vehicle must be able to go to a predefined location without the assistance of humans on roads that have not been adapted for its use.

AI technologies power self-driving car systems. Developers of self-driving cars use vast amounts of data from image recognition systems, along with machine learning and neural networks, to build systems that can drive autonomously. The neural networks identify patterns in the data, which are fed to the machine learning algorithms. These data include images from cameras on self-driving cars from which the neural network learns to identify traffic lights, trees, curbs, pedestrians, street signs, and other parts of any given driving environment. We are a long way away from having a completely self-driving car. [61] discussed different types of phantom attacks that causes the advanced driving assistance systems (ADASs) and autopilots of semi/fully autonomous vehicles to consider depthless objects (phantoms) as real.

In another example, a Tesla vehicle has been tricked into spontaneously accelerating over the speed limit with just sticking two inches of tape on a speed limit sign [62]. As we can see in Figure 3.1, McAfee security researchers placed a stripe of black electrical tape over part of a 35mph speed limit sign to slightly extend the middle of the "3". This tiny alteration made the car's camera misread the sign as 85 mph. The cruise control system then immediately accelerated towards this speed until the driver hit the brakes.

Figure 3.1: Researchers at McAfee placed a two-inch long piece of electrical tape horizontally across the middle of the '3' on a 35 mph (left) speed limit sign, causing the car's camera system to misread it as 85 mph (right) [62].

There is no doubt that traffic signs are very important parts of the road infrastructure considering either human drivers, driver assistance systems, or autonomous vehicles. Thus it is natural that traffic sign detection and classification [63, 64] have a huge attention from researchers. Contrary to the public perception, the detection and recognition of traffic signs with high accuracy is still an unsolved problem, especially in real-life conditions [63]. The challenges and difficulties of traffic sign/light detection can be summarized as follows: Illumination changes: The detection will be particularly difficult when driving into the sun glare or at night. Motion blur: The image captured by an on-board camera will become blurred due to the motion of the car. Bad weather: In bad weather, e.g., rainy, and snowy days, the image quality will be affected. Real-time detection: This is particularly important for autonomous

driving.

Figure 3.2: Traffic sings, as examples for the 7 distortion classes, under investigation: faded, covered, scribbled, correct, covered and faded, covered and scribbled, faded and scribbled.

Besides unfavourable weather, lighting, and imaging conditions the unwanted defects of traffic signs may heavily affect the accuracy of such systems, (see Figure 3.2). It is inevitable to develop systems to monitor the state of traffic signs, by detecting the different errors and supporting their maintenance. For this purpose, we proposed and compared different deep learning strategies for traffic sign defects in this chapter.

3.1 Datasets

3.1.1 Traffic Signs Distortion Dataset (TSD Version I)

There are several traffic sign datasets available but those contain no information about distortions. Our training traffic signs dataset includes 21 different types of traffic signs, see Figure 3.3 for sample images. The dataset, named Traffic Signs with Defects (TSD version I), contains 6016 images captured by dashboard cameras. We classified the dataset into 8 defect classes (Faded, Covered, Scribbled, No_error, Covered & Faded, Covered & Scribbled, Faded & Scribbled, Covered & Faded & Scribbled). Each defect class contains a different number of traffic signs see, Figures 3.4 and 3.5 for the exact number in each defect class. This dataset was used for training. For testing we used a separate dataset introduced in Section 3.1.1.1.

Figure 3.3: Traffic signs in the training dataset with their class codes.

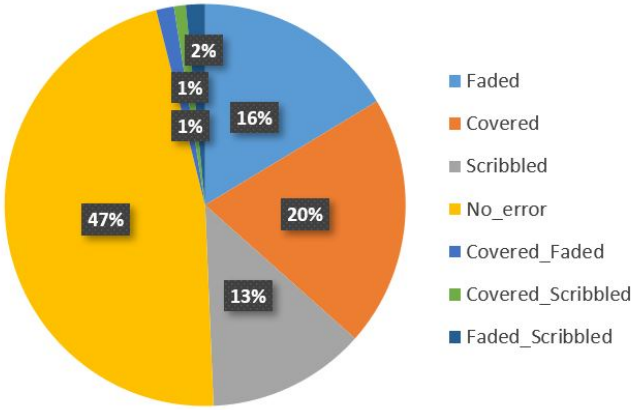

Figure 3.4: Percentage of images for each defect class in the training dataset.

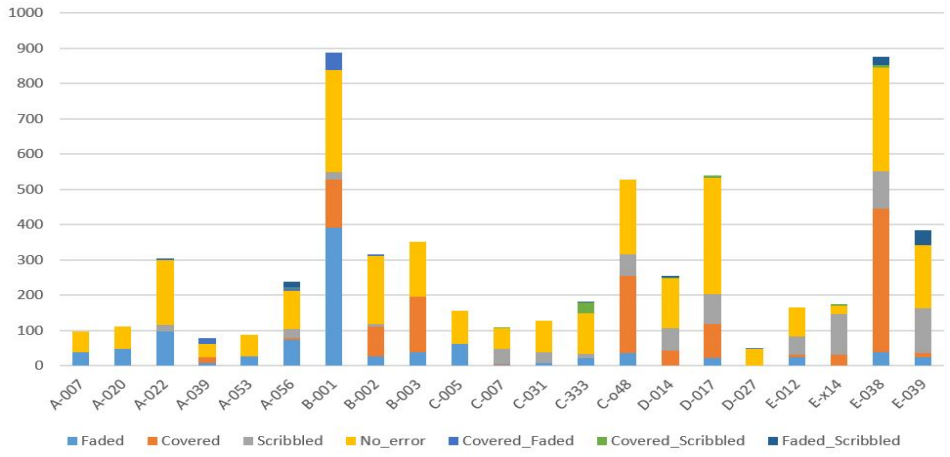

Figure 3.5: Distribution of traffic signs of the training data with each error class.

3.1.1.1 Untrained Classes of Traffic Signs for Testing (TSD Version I)

Figure 3.6: Examples images for the untrained traffic sign types.

Traffic signs could be considered as good test objects since we have lots of types of them. Unfortunately, it is not easy to collect a large number of real traffic sign images with different defects. We created a testing dataset with 25 traffic sign classes that were not used in the training. This dataset contains 490 images loaded with one of the following 4 defects: faded, covered, scribbled and errorless. See Figure 3.6 for examples of the new object classes and Figure 3.7 for the number of images in each defect class.

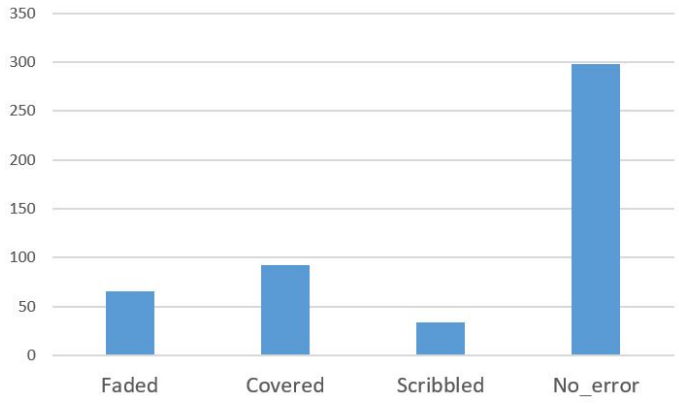

Figure 3.7: Distribution of the untrained traffic signs with each error class.

3.1.2 Traffic Signs Distortion Dataset (TSD Version II)

Figure 3.8: Examples for the 4 defect classes under investigation: (**a**) faded, (**b**) covered, (**c**) correct,(**d**) scribbled.

Because our previous dataset was heavily unbalanced and contains only 24 different traffic signs classes for training, we created a new dataset, Figure 3.8 shows some sample images of our collection, captured by dashboard cameras. In this dataset, we collected all traffic signs in our previous datasets for training and testing and added more traffic sign images. Additionally, we removed the mixture of classes to make the new dataset more balanced. The dataset, named Traffic Signs with Defects (TSD version II), contains 3792 images for training and 669 images for testing in 66 sign classes and 4 defect classes (faded: 1723, covered: 418, scribbled: 421, correct: 1899).

3.2 The Proposed Fusioning Convolutional Siamese Neural Network Architecture (FC-SNN)

Recently, the combination of deep learning algorithms with visual inspection technology allows the differentiating anomalies in objects mimicking human visual inspection. In this chapter, a new convolutional siamese neural model is presented to recognize different types of defects. We propose to train special SNNs to predict if the pairs of the images belong to the same defect class or not. We assume that in case of satisfactory training data, our network can generalize the visual appearance of visual defects, thus we can apply the same network for new object classes without retraining. The main contribution of this chapter is a siamese type network which, beside computing the difference of the features, contains the concatenation and further processing of these features. In the section, we refer to it as fusioning convolutional siamese neural network (FCSNN).

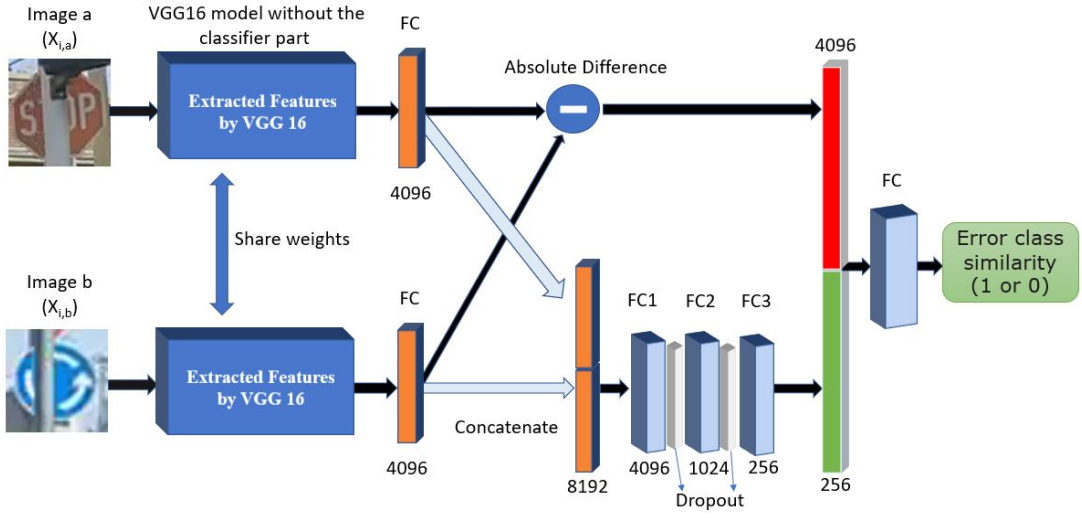

Figure 3.9: The proposed fusioning convolutional siamese neural network architecture.

For feature extraction, we used the ImageNet pretrained VGG16 model [65] as the parallel sub-networks as shown in Figure 3.9. The pair of input images ($X_{i,a}$ and $X_{i,b}$) are passed through the VGG16 networks to generate the fixed length feature vectors, then we added a fully connected classifier type layer to each branch to learn how to interpret the extracted features on our dataset. Thus we have two vectors of length 4096. We utilized these vectors in two different ways. First, we computed the absolute difference between the two feature vectors by L_1 distance. In the second branch, we concatenated the two feature vectors into one vector and fed it to three fully connected layers and two dropout layers with a dropout ratio of 0.2. At the end, we concatenate the two branches into one vector and feed it to a fully connected sigmoid layer to generate the similarity score output. The model was compiled using the Adam optimizer and the binary cross entropy loss function. The learning rate was set to 0.0004.

Figure 3.10: Examples of training pairs with their labels.

3.2.1 Training

To train a siamese network, we must create pairs of images: there can be pairs where both images are from the same error class and others where the two images are from different error classes. Figure 3.10 shows a few examples of how these pairs can look, good pairs (pairs with identical defect attributes) will be given label 1 and distinct pairs are labelled 0. We will generate these pairs randomly from all the defect classes in the training data, thus the dataset contains pairs of (X_i, Y_i) where Y_i is the required output (1 or 0). Recall that the input to our system will be a pair of images and the output will be a similarity score between 0 and 1.

3.2.2 N-way One-Shot Classification

We will perform N-way one-shot classification [66] as a strategy in order to evaluate the models. In one-shot classification, the query image in the test set X_{q_j}, $1 \leq j < \infty$ (the queries) is compared with each elements of the support set X_{s_i}, $i = 1, ...N$. We tested the model with $N = 20$ (20-way one-shot classification).
If we consider the set pairs of images is $P = \{(X_{s_i}, X_{q_j}), i = 1, ..., N\}$ and by ordering it in decreasing order, then the class of the first element of P with high confidence, $C_{X s_i}$, is considered as the decision of the network:

$$C_{X s_i} = C(\underset{C_{P_i}}{\operatorname{argmax}}\, Confidence(P_i)) \tag{3.1}$$

In case of correct prediction, the elements of the pair with the maximum confidence value, will be from the same error class. During testing, the generation of the support set, and the evaluation of the prediction is repeated k times:

$$CorrectPercentage = 100 \times n_{correct}/k \tag{3.2}$$

where k is total number of trials and $n_{correct}$ equals the number of correct predictions.

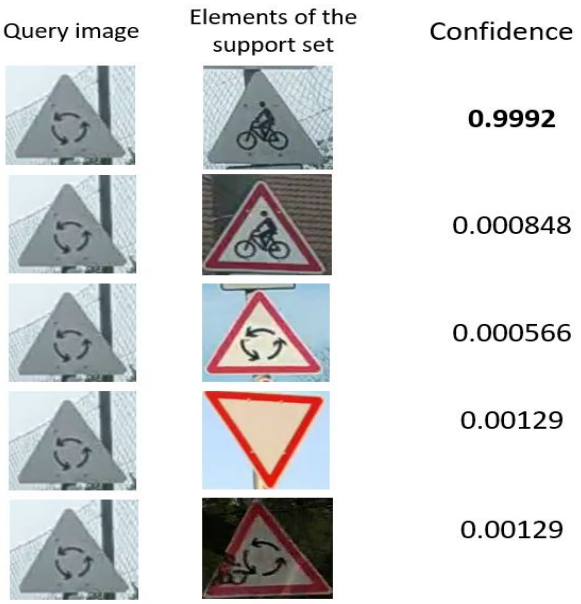

Figure 3.11: An example for 5-way one-shot testing. Since the pair with the same defect has the highest confidence, the network made a correct classification.

In Figure 3.11, we show an example of a 5-way one-shot classification. It is expected that the pair of images in the first line will have the highest confidence since they have identical defects.

3.2.3 Experimental Results

We evaluated the proposed method on two different datasets: our own traffic signs dataset and a dataset with disk castings[1] from the Kaggle website[2].

[1]We have tested the proposed approach on different datasets. In case of a casting dataset, we could achieve 99.60% accuracy. For more details, please check our paper [67].

[2]https://www.kaggle.com/ravirajsinh45/real-life-industrial-dataset-of-casting-product

The proposed FCSNN architecture was used to estimate whether two images are from the same error class or not. The testing method itself is quite strict since in the case of 20-way one-shot classification the recognition is evaluated as correct if the right pair has the highest confidence from the 20 comparisons.

Table 3.1: Accuracy of three siamese networks on untrained traffic signs (TSD version I) in three independent tests.

Test cases (distribution of images)	SNN [68]	SNN [68] with VGG16 Features	FCSNN
12 traffic sign classes, 66 faded and 162 errorless	43.3%	80.6%	92%
21 traffic sign classes, 92 covered and 195 errorless	7.5%	27.9%	28.9%
6 traffic sign classes, 34 scribbled and 54 errorless	9.4%	11.5%	25.8%
Weighted average accuracy	22.32%	43.12%	52.81%

The testing dataset introduced in Section 3.1.1.1 was used for testing. It contains 25 traffic sign classes that were not used in the training of the FCSNN. We evaluated the defect classification accuracy of SNN [68], SNN with VGG16, and FCSNN with the 20-way one-shot classification technique. The average results of 10 experiments are given in Table 3.1. Three test cases, for the error classes faded, covered, and scribbled, were evaluated independently and their weighted average is also given. The performance of SNN is the lowest. To be able to measure the contribution of our proposal we replaced the feature extraction part of [68] with VGG16. This modified network gave significantly better results. Our proposed FCSNN outperformed this variation with circa 9.5% in average. The results of the proposed method are good only for the faded class, but for the rest of the defect classes, the performance needs more improvement. In the following section, we introduced a new method to handle this problem.

3.3 Recognition of Traffic Signs Defects with SVM based on CNN Confidence Values

In the previous section, the idea was not only to differentiate but also to fuse information from the twin branches. For faded signs, measured on untrained traffic sign classes, 92% accuracy was reached by the method denoted as FCSNN, but other types of defects were much less successfully recognized. To enhance the performance of DNNs, it is a common way to use an ensemble of classifiers. The drawback of such solutions is the increase of computational resources (time and memory) during training and also in testing the queries with a set of networks. Contrary, we propose to use only one FCSNN but we compare the query image to many support images (in our test 5, from each defect class, chosen randomly). In this concept the support images are considered as reference images and the task of the SVM is to learn how the confidence of the decisions of the FCSNN can help in the recognition of defects.

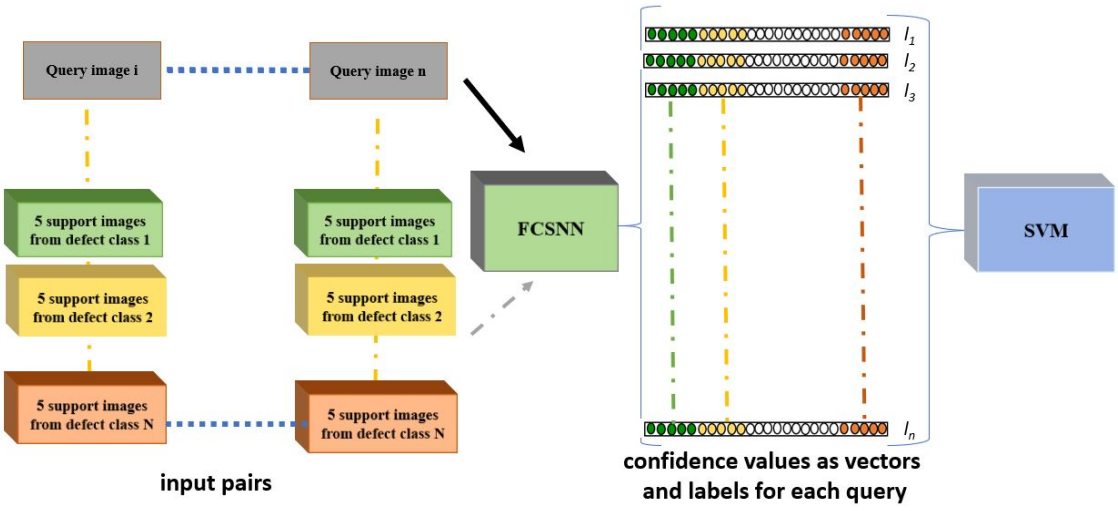

Figure 3.12: Combination of FCSNN with SVM in the training process. SVM is trained on the confidence values and true labels of several comparisons of training images and the elements of the support set.

In previous section, to estimate the defect type, a random pair of images (the query image and one image from the support set) were evaluated by the FCSNN to judge the defect similarity. Now, instead of one image from each defect type, we use multiple images from each defect classes as support and train a SVM on the obtained confidence values of decisions of several comparisons of the same query with different support images. The new idea of our approach is to learn from the 'imperfections' of inferences of the DNNs: the SVM has the ability to make better decisions by observing how the trained DNN behaves when it compares different queries to different support images. This information is represented in the vectors of confidence values. See Figure 3.12 for the illustration of the proposed combination of the FCSNN and SVM.

First, FCSNN was trained with 3792 images from TSD version II dataset. The pretrained FCSNN is used to compute the confidence values of the query image paired with each image in the support set. Second, to train the SVM, we used the pre-trained FCSNN model to create matrix of confidence values of size $M \times K$, where M equals the number of images in the training set (in our case 3792)and K equals the total number of images in all support set for all classes. For testing, the pre-trained SVM will recognize the defect class.

3.3.1 Experiments and Discussion

To evaluate the new concept we made experiments with the TSD datasets; training and testing subsets were introduced in Section 3.1.2. Additionally, we introduced our new dataset (TSD dataset[3]) of over 4000 traffic signs in 66 classes and 3 types of defects (covered, faded, and scribbled) on 2562 images (plus 1899 error-free images).

[3]https://keplab.mik.uni-pannon.hu/images/tsd/

The structure of the FCSNN was identical to [67], training used the Adam optimizer, batch size 32, 150 epochs, with a learning rate 0.0001. For base models we have chosen VGG16 and ResNet50.

The SVM and the FCSNN were trained independently. The SVM was using the radial basis function as the kernel and the classes were balanced during its training.

3.3.1.1 Experimental Results on TSD Version I

Table 3.2: FCSNN-ResNet50-SVM evaluation on TSD version I. The accuracy on the 490 test images of untrained classes of traffic signs was 77%.

Defect class	**Metrics**			
	Precision	**Recall**	F_1 **score**	**# of queries**
Covered	0.64	0.59	0.61	92
Faded	0.82	0.91	0.86	66
Correct	0.89	0.82	0.85	298
Scribbled	0.35	0.53	0.42	42
Weighted avg.	0.80	0.77	0.78	490

We tested the performance of the proposed method on the dataset introduced in Section 3.1.1. Here the SVM is trained with confidence vectors of length 7×5 (as we have 7 defect classes and 5 support images from each class) and the defect labels (from l_1 to l_n). Table 3.2 shows the results precision 0.80, recall 0.77 and F1-Score 0.78. The weighted accuracy is 77%.

3.3.1.2 Experimental Results on TSD version II

We tested the performance of the proposed method on the dataset introduced in Section 3.1.2. Here the SVM is trained with confidence vectors of length 4×5 (as we have 4 defect classes and 5 support images from each class) and the defect labels (from l_1 to l_n).

Table 3.3: FCSNN-ResNet50-SVM evaluation on TSD version II. The accuracy on the 669 test images of TSD was 88%.

Defect class	Metrics			
	Precision	**Recall**	**F_1 score**	**# of queries**
Covered	0.84	0.86	0.85	258
Faded	0.91	0.90	0.90	63
Correct	0.86	0.92	0.89	285
Scribbled	0.87	0.62	0.72	63
Weighted avg.	0.88	0.88	0.88	669

For performance evaluation precision, recall, and F_1 score were computed for each defect class and the average accuracy for the whole test set.

Table 3.3 shows that there is a large difference between the different error classes. For fading we could achieve 0.90 F_1 score, while for scribbled traffic signs we could reach 0.72. The average F_1 score is 0.88 and the average accuracy is also 88%.

3.3.2 Comparing FCSNN-SVM with Different DNNs Models

In this section, we discuss different neural network approaches to find various errors on already detected traffic signs. Two major approaches are investigated: convolutional neural networks to learn the features of defects, and siamese convolutional neural networks to compare traffic signs with others with known distortions. While the former models are known for their good performance in object recognition in general, the later networks are often used for the detection of defects of objects. Neither approach requires information about the type of the traffic sign itself.

To get a more comprehensive overview of the ability of DNNs for our task, and to compare siamese and feature-based classification DNN approaches we tested 9 models. Six of them contain siamese structure [69]:

SNN of [68], SNN with VGG16 backbone, FCSNN with VGG16 backbone [67], FCSNN with ResNet50 backbone, FCSNN with VGG16 backbone plus SVM classification, FCSNN with ResNet50 backbone plus SVM classification and VGG16, ResNet50 [70], EfficientNet-B6 [71].

Table 3.4: Evaluation of SNNs and feature based classification approaches. In case of four methods (first four lines) we could increase performance by choosing the defect class with the highest average confidence on the support support set.

	F_1 **score**	
Method	**Top-1**	**Class average**
SNN [68]	0.833	0.844
SNN-VGG16	0.861	0.863
FCSNN-VGG16 [67]	0.868	0.870
FCSNN-ResNet50	0.869	0.870
FCSNN-VGG16-SVM	0.870	
FCSNN-ResNet50-SVM	0.880	
VGG16	0.860	
ResNet50	0.860	
EfficientNet-B6	0.890	

In Table 3.4 we list the test results: column 'Top-1' gives F_1 score values based on the decisions of the SVMs (fifth and sixth lines), on the highest confidence values of the SNNs (first four lines), or of the classification networks (last three lines); column 'Class average' is based on the average of confidence values of each defect class of the support images.

The difference among them, in average F1-score, is between 0 and 0.057. The difference between the best and 2nd is 0.01. The advantage of the proposed siamese network (FCSNN) is that starting from the base models (VGG16 and ResNet50) and applying FCSNN+SVM resulted in 0.01 and 0.02 F_1 score improvements, respectively. Another contribution is that SVMs can further improve the performance: FCSNN-VGG16 $\rightarrow$ FCSNN-VGG16-SVM: 0.002 (improvement), FCSNN-Resnet50

→ FCSNN-Resnet50-SVM: 0.011 (improvement). EfficientNet [71] is a promising family of models (from B0 to B7) which can balance between accuracy and the necessary resources (memory and computation time). The largest variant (B7) achieved state-of-the-art 84.4% top-1 accuracy on ImageNet in 2019, and it could reach the same performance as the previous state-of-the-art model but being 8.4x smaller and 6.1x faster on inference [71]. We used the smaller B6 member of the family, reaching the best top-1 accuracy. Our technique can use any backbones, but we didn't test EfficientNet as backbone as it is the task of the future.

3.4 Summary

We proposed a new siamese neural network architecture (FCSNN) to recognize the defects of different objects. The previously proposed networks were extended by several layers and the original features, besides computing the difference, were retained for fully connected layers. Two datasets were used for evaluation with 20-way one-shot testing. These tests show that the proposed architecture performs significantly better than previous solutions for such cases. Moreover, we test the generalization properties of our network to learn the latent defect-specific features by predicting the errors of new untrained object classes with a different appearance.

We introduced two datasets (TSD versions I and II) of defected traffic signs that can be very useful for evaluating anomaly detection (e.g. for maintenance purposes) and for the research community to test the error-prone abilities of traffic sign recognition methods. We proposed a new mechanism to combine the output confidence values of siamese networks with SVM with the help of support set images. Compared to the VGG16 and ResNet50 base models the achievement of the proposed FCSNN-VGG16-SVM and FCSNN-ResNet50-SVM models is 0.01 and 0.02 in F_1

score on TSD version II, respectively. Overall, we compared 9 different DNN approaches on the introduced TSD version II dataset.

The advantage of this approach, compared to the concept of an ensemble of networks, is that only one network is to be trained and maintained (one model can handle many traffic sign and error classes).

This chapter was summarized in Thesis II, please see Chapter 5, Section 5.1.

Chapter 4

Classification, Zero and Fast Few-Shot Learning of Steel Surface Defects

4.1 Introduction

There is a variety in the appearance of surface defects in industry, for example, hot-rolled steels, solar panels, electronic commutators, steel rails, fabrics, printed circuit boards, magnetic tiles, and more. Steel is the most important metal in terms of quantity and variety of applications in the modern world. Surface defects in hot-rolled steel can be associated with the steel production process, its casting, deformation conditions, crystallization of the ingot, etc. They have a considerable impact on the metal's technological qualities during future processing as well as on its operational features. Due to the manufacturing process and environmental conditions, steel surfaces can have a variety of defects. The non-uniform surface brightness and the variety of shapes of defects make their detection challenging. Additionally, new defects, which have never been seen before, may arise throughout

the manufacturing process. In the beginning, these new defects may have few-shots, so we have to go through fast incremental learning to incorporate them into the classification model as soon as possible.

Steel surface defects show various random patterns, which are good targets for the testing and comparison of concurrent classification methods. Figure 4.1 and Figure 4.2 illustrate samples of the two popular benchmark datasets: the Xsteel surface defect dataset (X-SSD) [72] and the Northeastern University surface defect database (NEU) [73]. There are six categories in NEU and seven in X-SSD, with two types of defects present in both. Since the two common artifacts (inclusion and scratches) look somewhat different, we handle them independently.

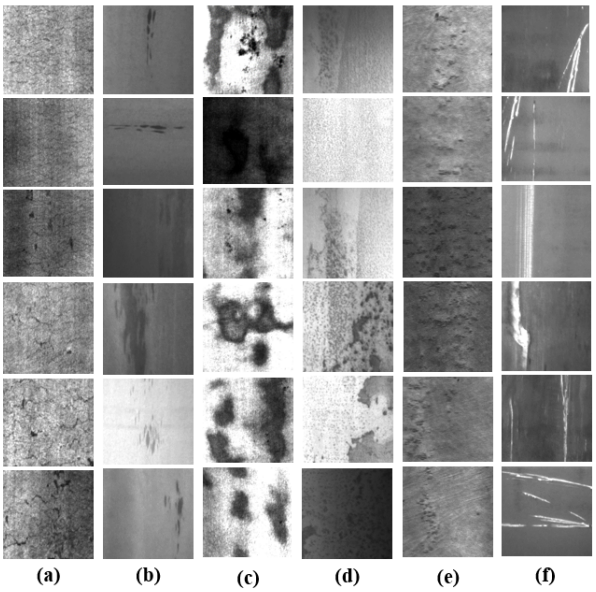

Figure 4.1: Examples for the six kinds of defect classes of the Northeastern University surface defect database (NEU) [73]: (**a**) crazing (Cr), (**b**) inclusion (In), (**c**) patches (Pa), (**d**) pitted surface (Ps), (**e**) rolled-in scale (Rs), (**f**) scratches (Sc).

The image processing methods dealing with steel surface defects can be classified into two main categories:

- Detection: The task is to decide whether the image patch has any errors,

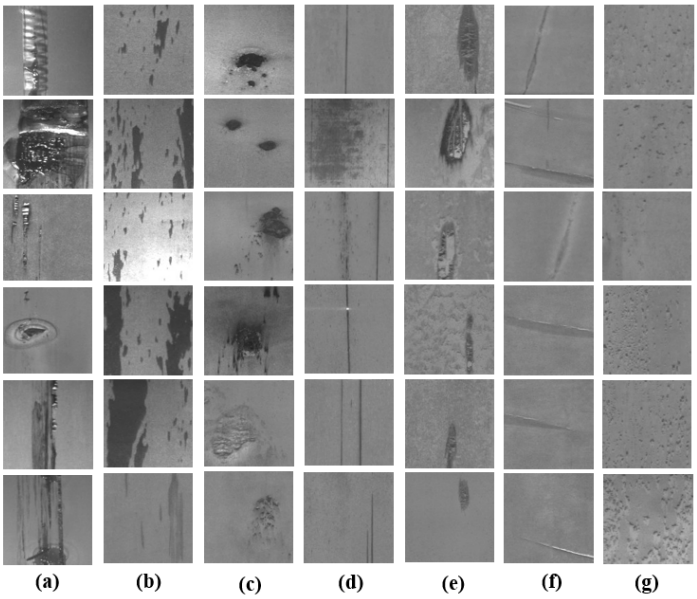

Figure 4.2: Examples for the seven kinds of defect classes of the Xsteel surface defect dataset (X-SSD) [72]: (**a**) inclusion (Si), (**b**) red iron sheet (Ri) , (**c**) iron sheet ash (Is), (**d**) scratches (Ss), (**e**) oxide scale of plate system (Op), (**f**) finishing roll printing (Fr), (**g**) oxide scale of temperature system (Ot).

sometimes the error should be localized (segmented) and also classified [21, 74, 75, 76, 77].

- Classification: In case of classification, the image patches should be labelled according to the visible defect type. Training examples are available to learn the visual features of known classes [78, 79, 80].

Surface defect detection can be considered as a typical problem in visual quality inspection thus it is related to many areas of pattern recognition, machine learning, image processing, and computer vision. In most applications, beside the questions of detection and classification, the following key problems are to be answered [81]:

- The real-time problem: The detection time should be fast enough to support the undergoing (production) process without significant loss in accuracy;

- Small target problem: Often the absolute or the relative (to the image) size of the defect is small. In this chapter, we do not have to face this problem directly since we use benchmark datasets, however, in a real-life application, the size can have an indirect effect on the real-time problem;

- The small sample problem: The number of defect images, used either in manual parameter tuning or in automatic learning mechanisms, is often very limited;

- Unbalanced sample problem: This problem mainly exists in the task of supervised learning and relates to the previous one. Often the number of normal samples forms the majority, while the amount of defected samples only accounts for a small part. The few-shot learning problem means that we have a very unbalanced set;

- Domain shift [77]: When the dataset used in training and the dataset in practice are captured under different conditions, it can result in poor detection performance.

In incremental learning, new data (i.e. new shots of previously seen or unseen classes) arrive in phases over time and we have to extend our classification model to include these new classes or new samples of classes. In general, in the literature different synonyms are used for this problem, such as continual learning [82, 83], lifelong learning [82, 84], or incremental learning [85, 86, 87, 88, 89, 90]. The three major challenges in these scenarios are:

- Catastrophic forgetting: It was already investigated in the early 1990s [91] that a neural network is going to have a lower performance of previously trained classes when re-trained or tuned with new ones in focus. The problem is most serious if we have no training data for the already trained classes, or

the dataset for re-training is unbalanced due to the missing samples from old classes. There are several approaches to avoid this case by methods such as scaling the weights of trained classifiers [92], using dual memories for storing old images and their statistics [93], or progressive incremental learning [94] where sub-networks are added incrementally to previous ones as the task is growing with new classes;

- Low number of representatives of new classes: In practical applications there is a data collection period when lots of samples are collected about the possible artifacts and used for model building. In contrast, when the system is in use for long periods, the undergoing background manufacturing processes (changes in the environment, or other influences) can result in new defect classes, which might appear in very few samples initially. In general, solutions are categorized into three main sets: using prior knowledge to augment the supervised experience (in our case data augmentation of the available few shots); modifying the model, which uses prior knowledge of known classes; and algorithmic solutions to alter the search for the best hypothesis in the given hypothesis space [95]. In extreme cases, we talk about zero-shot learning: when a new type of defect appears, it is a question whether we are able to classify it as a new class or if it will fall into an existing category. We would like to avoid this later case and thus we have to create a new category and be able to tune the existing model to consider it [78, 96, 97].

- Complexity of updating the classifiers: It is a practical problem during the application of deep learning models that the re-training or fine-tuning of new or updated models can be time consuming or computationally very complex. A fast adaptation to the extended set of classes and easy training procedures

are needed for real-life applications.

Typically, the above three problems appear simultaneously. The advantage of our proposed solution is that it addresses all of them with one architecture, instead of attacking the problems independently with different methods. The main contributions are the following:

- Instead of re-tailoring old models or defining new deep neural architectures, we show that a well-proven and efficient deep neural network (EfficientNet-B7 [71]) can give the best known accuracy for the classification of steel surface defects on two major benchmark datasets.

- We propose a novel architecture designed for incremental learning with the following features: it avoids the catastrophic forgetting of old classes, it has good performance in case of very few shots, and it can be trained very fast for new classes. To achieve this, we apply randomized networks concatenated to the feature extraction of a pre-trained DNN. The computation of its weights can be done very efficiently by the Moore–Penrose generalized inverse.

4.2 Related Works

Surface defect detection methods can be classified into traditional feature-based machine vision algorithms and deep learning algorithms [81]. Since all of the latest papers belong to the later set, our review contains such approaches. We review related papers in three groups: steel surface classification methods, zero-shot learning approaches, and few-shot learning approaches (with some particularly tested on steel surface defects).

4.2.1 Detection and Classification of Steel Surface Defects

In [76], authors introduced an improvement of the YOLOv4 architecture by adding a feature pyramid network module after sampling, on the so-called neck part of the network. Enhancing the feature information this way, the experimental results showed a better average detection accuracy (92.5%) than the original network. Tests were carried out only on three defect types of the NEU dataset (crazing, patches, scratches) and more improvements were expected by the authors if larger training sets could be involved. In [75], an improved faster R-CNN model was proposed by using deformable convolutions instead of conventional convolutions to get complex features for the detection of the various artifacts. Moreover, the RestNet backbone-based network was enhanced by multi-level feature fusion. By these techniques, the detection rate could be enhanced by approximately 0.13 mAP. In [21], a framework called CPN (classification priority network) was described for the detection and classification of defects. In CPN, the image is first classified by a multi-group CNN, training different groups of convolution kernels separately to extract the feature map groups of different types of defects. Then, according to the classification result, the feature map groups (named multiple group CNN, MG-CNN) that may contain defects are separately input into another neural network, to regress the bounding boxes of the corresponding defects. The detection rate reached 96% and the recognition rate 98.3% on their own dataset. These seem to be good results; however, the zero-shot possibilities were not mentioned. In paper [74] proposed an end-to-end defect detection network (DDN). First, ResNet is used to generate feature maps of the input images. Then, these feature maps are fed into the proposed multi-level feature fusion network (MFN). MFN generates one feature vector from the lower-level and higher-level features. A region-proposal network is used to generate regions of interest (ROIs) based on these hierarchical features. Finally, a detector, which

consists of a classifier and a bounding box regressor, is used to produce the final detection results for each ROI. DDN results, when using ResNet50 as backbone, show that it could achieve 99.67% accuracy for defect classification and 82.3 mAP for defect detection. We note that the former value is equivalent to the plain ResNet50 according to paper [98].

[77] attacks the classification problem when there are changes in the features of the patterns, for example, the appearance of steel surface defect changes during long production intervals. Their framework is called DA-ACNN, since it combines domain adaptation (DA) and the adaptive learning rate of the convolutional neural networks (ACNN). To enable cross-domain and cross-task recognition, they included an additional domain classifier and a constraint on label-probability distribution to account for the lack of labels in a new domain. Additionally, to increase network performance, the normal distribution and a quadratic function are utilized to optimize the loss.

In [79], authors proposed a steel surface defection classification technique fine-tuned with the help of a feature visualization network. The proposed model consists of two main components: the VGG19 network, trained, fine-tuned, and later used for the classification of surface defects, and a decoding part (DeVGG19). The feature maps of the decoding part are used to find the best settings of VGG19 using image and feature map comparisons with the structured similarity index measure (SSIM), and while it is an interesting approach, it is not obvious for the reader why the aspects of the decoded visual appearance could have a positive feedback on the parameters of the network used purely for classification. This interesting approach could reach 89.86% on the NEU dataset, which is below the state-of-the-art (see VSD in Section 4.6, Table 4.3).

In paper [80], different versions of ResNet were investigated to classify three kinds of defects on metal surfaces (scratches, scrapes, abrasions, and normal samples). The best results were achieved with ResNet152, with a classification accuracy of 97.1% on their own collection of images (including images of NEU). This paper is a good illustration that properly trained off-the-shelf networks can reach a good performance on steel surfaces. Another example is a modified AlexNet for feature extraction, where the classification was solved with a support vector machine [99]. Performance on all classes of NEU showed 99.70% accuracy. The highest known performance is published in paper [78] where VGG16 was extended with several layers for classification (ExtVGG16) and could reach 100% on the six classes of NEU, (see Figure 4.3).

4.2.2 Zero-Shot Learning

Zero-shot learning can be considered as a special case of few-shot learning, when there are no samples at all from some of the classes. In general, it models learned parameters for seen classes together with their class representations and rely on the representational similarity among class labels. Semantic attributes can represent the characteristics, as latent attributes, of samples. The success of zero-shot learning thus depends on the quality of semantic attributes: whether they have sufficient discrimination and expressiveness. However, semantic attribute annotations are very hard to get in industrial applications. One example of the usage of such an approach for surface defects is in paper [97] where the so-called GloVe model [100] was used to extract word vectors as the auxiliary knowledge source (since they used a different dataset, the results are not comparable to the others involved in our article). If no such kind of semantic attribute annotations are available, then still-clustering-type approaches can work. In [96], authors show a method in which zero-

shot learning may be used to detect steel surface defects with a siamese network. The network is to learn whether two input samples belong to the same class or not. In experiments, three defect classes were trained from the NEU dataset, and the siamese-based clustering was run on the other three classes, which were never shown to the network during training. The accuracy of this task, reaching 83.22%, could be surpassed in Section 4.4, where also a siamese network was used. This later approach uses a slightly more complex structure and there are also differences in dropouts and regularization, which may all play an important role to reach better accuracy (85.8%) at the same testing and training conditions.

4.2.3 Few-Shot Learning

A learning model fed with sufficient and high-quality data is more likely to yield more-or-less accurate results. However, sometimes it is difficult to collect enough defect samples to achieve a good training model based on traditional learning structures. This can be the case in the production of steel strips, where steel surface defects are scarce and hard to collect. To overcome this problem, special approaches appeared, which can efficiently learn from a very small number of training data.

[101] shows a simple approach for few-shot learning of steel surface defects. Three feature extraction networks (ResNet, DenseNet, and MobileNet) are compared and two kinds of feature transformations are tested (mean subtraction and L2 normalization). For one-shot learning, classification is accomplished by choosing the nearest neighbor, whereas for the multi-shot setting, the average value of the known sample feature vectors of each class is used as a class prototype to compute the nearest neighbor for the query. For the one-shot case, MobileNet reached the highest accuracy with 87.3%, while for five shots 92.33% was achieved by DenseNet121 on the NEU dataset.

In [102], a more sophisticated semi-supervised learning model is described, which learns from both labeled and unlabeled samples. They use graph convolutional networks (GCN) [103], naming their model multiple micrographs graph convolutional network (MMGCN). GCNs construct graphs where the images are represented by nodes and their relationships by edges. Feature information propagates between connected nodes, and the distance of connected nodes is closer than the distance of the disconnected ones. MMGCN performs graph convolution by constructing multiple micrographs instead of a large graph, and labels unlabeled samples by propagating label information from labeled samples to unlabeled samples in the micrographs to obtain multiple labels, while final labels are obtained by weighting the labels. The experimental results demonstrate that the proposed MMGCN can achieve a lower computation complexity and practicality, and a higher accuracy than GCN. In fully supervised mode, MMGCN could reach 99.72% accuracy, while when the ratio of known labels was only 25%, accuracy was 98.06%.

Another approach for label propagation is in paper [104], where defects on the surface of lithium batteries are to be found. Training images were used to fine-tune a pre-trained ResNet10 model, then a k-NN graph was built to evaluate the distance between samples, including both the labeled and the unlabeled ones. To refine the scoring of samples, the label propagation method of paper [105] was used. Unfortunately, while classification accuracy on their own dataset was good, the effect of label propagation was not analyzed in the article.

Few-shot learning can be interpreted as the problem of unbalanced training sets. Article [106] proposes a meta-training approach to handle the unbalanced data caused by the newly arriving error classes having only few shots. Each training epoch is composed of so-called episodes and sub-sets of training data are called support sets. At the end of the episodes, the classification performance of a query set, based

on knowledge gained from the respective support set, is evaluated to fine-tune the model parameters. The tuning of DNN parameters is solved by a prototypical approach. Each class is represented by a prototype feature vector and the weights of the feature extractor DNN will be tuned to separate the sample features from other prototypes the most. The superiority of this training approach is compared to other traditional classification models by evaluations on a textile dataset.

4.3 The Benchmark Datasets

There are several steel surface defect datasets available for the benchmarking of algorithms. First, we mention GC10-DET [107], which contains 3570 gray-scale photos of steel surfaces, including 10 kinds of defects. Another new dataset, published in paper [108], consists of 21853 RGB images showing areas with and without failures called pitting, while in paper [108] , only two classes are specified—this dataset shows the progression of failures at different time moments, which can be very useful for those who want to detect failures as soon as possible.

To test EfficientNet-B7, we used the two benchmark datasets already illustrated in Figures 4.1 and 4.2, and described in the following subsections. The reason for choosing them, beside the relatively large number of classes and large variance in appearance of these errors, is the wide availability of the concurrent methods used in the evaluations. Since we found X-SSD more challenging (see the experimental section below), we analyze our few-shot learning technique only on the X-SSD.

4.3.1 Northeastern University Surface Defect Database (NEU)

NEU [73] consists of six different kinds of surface defects. Images, with resolutions of 200 × 200 pixels and having only one channel, were collected from hot-rolled steel strips. It contains 1800 images: each defect class has 300 samples. The classes, illustrated by Figure 4.1, are: crazing, inclusion, patches, pitted surface, rolled-in scale, and scratches. Unfortunately, paper [73] does not contain information about how the defects are generated during the production of the hot-rolled steel strips. The dataset was divided randomly into 80% for training and 20% for testing. Beside NEU, there is a dataset called NEU-DET [74] with the annotated bounding boxes of defects. This is out of focus from our perspective now. In the tests of EfficientNet-B7, images were resized to 128 × 128 pixels (without a loss of accuracy, as will be shown later).

4.3.2 Xsteel Surface Defect Dataset (X-SSD)

The work [72] introduced a dataset for hot-rolled steel surface defects with 1360 images. Each image has a resolution of 128 × 128 pixels and has three color channels. This dataset, as illustrated in Figure 4.2, consists of seven different defect classes: 397 red iron sheet (Ri), 122 iron sheet ash (Is), 238 inclusions (Si), 134 surface scratches (Ss), 203 finishing-roll printing (Fr), 63 oxide scale-of-plate system (Op), and 203 oxide scale-of-temperature system (Ot). Paper [72] gives some information how the different defects are generated during the production of the hot-rolled steel strips:

- Red iron sheet: high silicon content in steel and high heating temperature of slab;

- Iron sheet ash: accumulated contamination (e.g., dust, oil) falls onto the surface;
- Inclusions: inclusion of slags in the steel;
- Surface scratches: hot-rolling area with projections—dead or passive rolls can cause friction on the surface;
- Finishing-roll printing: the slippage between the work roll and the support roll can result in dot and short-strip damages on the surface of the work roll;
- Oxide scale-of-plate system: if the roller table is damaged it can also damage the surface of the rolled piece where the iron oxide particles can accumulate and they can be rolled into the steel in the subsequent rolling process;
- Oxide scale-of-temperature system: it can be caused by many things, such as improper temperature settings, high carbon content, and unwanted intense oxidation.

4.4 Zero-Shot Learning and Classification with Siamese Neural Network

Developing machine learning models that can perform predictive functions on data it has never seen before has become an important research area called zero-shot learning. Human are pretty good at recognizing objects in the world we've never seen before, and zero-shot learning could replicate that amazing human talent. Zero-shot learning aims to predict the correct class without being exposed to any examples belonging to that class in the training dataset.

We propose neural networks for the recognition of new defect classes and also for the classification of known types. For the former a zero-shot approach, based on a siamese network, is used learning features to classify unseen classes without a single training example. Additionally, we can utilize one branch of the same network for the classifications of previously trained defects. For performance evaluations, experiments were carried out on two benchmark datasets: the Northeastern University [72] and the Xsteel surface defect datasets [73] (see Figures 4.1 and 4.2). Our method is compared to the paper [96], which demonstrates the efficiency of siamese technology in detecting various defects in steel surfaces while minimizing the amount of training data required. The method was trained on three defect classes from the NEU dataset and was tested on the other three classes which were never seen to the network and achieved accuracy of 83.22%.

4.4.1 The Proposed Methods

4.4.1.1 Zero-Shot Learning to Detect New Types of Anomalies

In Section 3.2 to identify the defects of different objects, we proposed a siamese neural network architecture where, besides computing the difference of the features of the two inputs, the features are also concatenated and further processed. This gives the ability to the network to analyze the differences of features in the light of other features of the images.

In comparison of our previous proposal in Section 3.2, we made the following modifications:

- fully connected layers are added at the back end to increase the classification abilities;
- to prevent overfitting of the network with higher complexity, we added more

dropout layers and used l2 regularization at each fully connected layer except the last one, (see Figure 4.3).

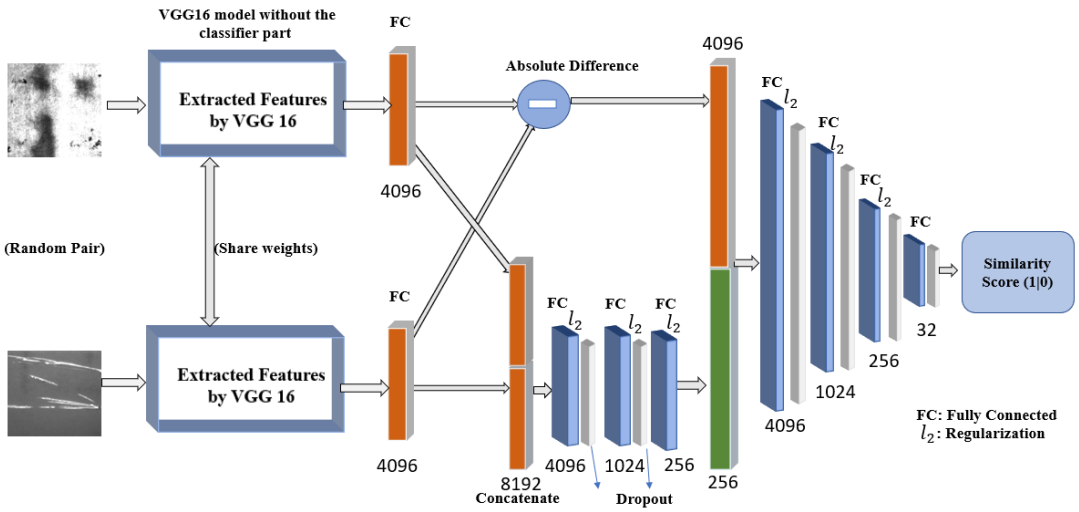

Figure 4.3: Proposed siamese architecture based on the fusing convolutional siamese neural network (FCSNN).

4.4.1.2 Classification of Known Classes

If new, unknown patterns are not expected then we have to solve the classification problem based on available training data. To solve this task we propose the truncated versions of our siamese network: only the upper part in Figure 4.3 is kept without computing the difference vector and removing the middle layer of 4096+256 length, we call it extended VGG16 (ExtVGG16). All training parameters are the same as for the siamese version. As we will detail in the experimental ExtVGG16 architecture is able to outperform all previous DNNs attempts.

The model was compiled using the Adam optimizer and the learning rate was set 0.0005. For training, a binary cross-entropy loss function was used.

4.4.2 Experiments and Discussion

We present experimental results on the two benchmark datasets, and we also give the results of concurrent recent methods where available.

4.4.2.1 Results on the NEU Dataset

To evaluate the performance of zero-shot learning, we followed the method of [96], thus we can give a direct comparison of results. The training dataset contains images of three defect classes: rolled in scale, inclusion, and patches. The testing dataset contains the other three remaining classes (scratches, crazing, and pitted-surface), which were never seen by the network. The training dataset was augmented by random rotations between 0° and 30°, vertical filliping, horizontal flipping, and zoom in the range equal to 0.2. To decide if the two images are from the same class, we threshold the confidence value for each pair at 0.5. Accuracy was measured on a large number of experiments. For all six possible pairings of the three untrained defect classes we formed 22,500 different pairs. Our siamese DNN model achieved accuracy equal to 85.80% (when VGG16 backbone was frozen during training) which is higher than the 83.22%, published in [96], as highest known value.

To evaluate the performance of classification, we compared the proposed method with the state-of-the-art algorithms; the best known accuracy on NEU was previously achieved by SBF-NET [98]. The dataset was divided randomly into 80% for training and 20% for testing. Images were resized to the size of 128x128, for augmentation we applied the same transformations as detailed before, and the training was running for 300 epochs. Optimizing all parameters of the network we obtained 100% accuracy, while if the layers of the VGG16 backbone were frozen accuracy reached 99%. Table 4.3 shows the comparison of all known alternatives (data of

other algorithms are taken from [98]).

4.4.2.2 Results on X-SDD

To test the ability to cluster unseen defects by our siamese approach the dataset was divided into training with four classes (finishing roll printing, oxide scale of temperature system, red iron sheet, and inclusion) and testing with the remaining three (iron sheet ash, scratches, oxide scale of plate system). Augmentation was identical to the above described, as well as other parameters of the training process. In these experiments, we achieved 61.55% accuracy. We found no reference data for X-SSD for such an experiment in other articles.

Contrary, many other methods are there to classify the defect classes of X-SSD. We found the best result in [72]. For training, we followed the same steps as described in [72], the dataset was divided randomly into 70% for training and 30% for testing, thus the training set contains 952 images, whereas the testing contains 408. All parameters of training are identical to our other procedures.
Beside accuracy Table 4.4 shows the computed macro-precision, macro-recall and Macro-F_1 score values for 13 different methods, including ours (source of data is [72]). Macro values refer to the averaging of class averages. Our network outperformed all previous approaches, only the technology introduced in the later section reached higher results.

4.4.3 Further Discussion

Our proposed DNN solution is the result of the evaluation of settings and parameters of a large number of experiments with both datasets. We achieved very good results in classification with the VGG16 backbone of siamese network, thus we found

Table 4.1: Comparison of accuracy values depending whether feature extraction was using ImageNet weights (Freeze) or were optimized (Unfreeze).

NEU Data-set				**X_SSD Data-set**			
Zero-shot		Classification		Zero-shot		Classification	
Freeze	Unfreeze	Freeze	**UnFreeze**	**Freeze**	Unfreeze	Freeze	**UnFreeze**
85.80%	83.42%	99%	**100%**	**61.55%**	60.59%	94%	**98%**

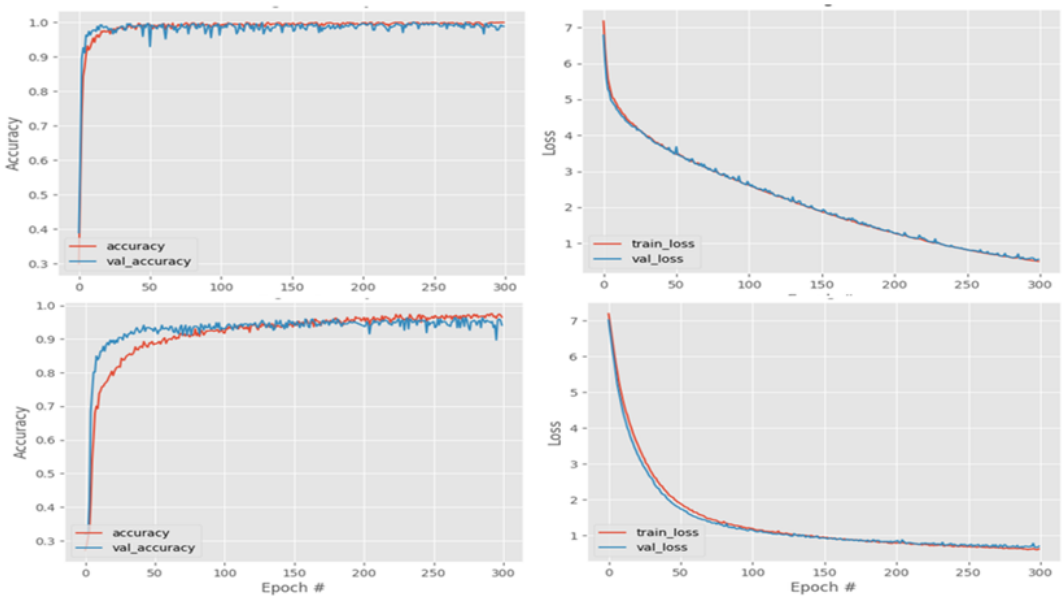

Figure 4.4: Accuracy and Loss value curves when training classification on X-SSD. First row: trainable layers of VGG16 backbone. Second row: freezed VGG16 backbone.

it interesting why VGG16 [51] network showed moderate performance in the classification of X-SSD samples, see Table 4.4. We could reproduce similar results if the weights of VGG16 were frozen (and using the ImageNet weights) but for classification we think this is not reasonable. In contrast, for the discovery of new defect classes the adaptation of weights to the few known classes can be disadvantageous. This hypothesis is supported by Table 4.1 and the curves of Figure 4.4.

4.5 Proposed Methods for Classification and Few-Shot Learning

What we have learnt in the previous section is that different ideas were implemented to improve the performance of DNNs for the detection of defects. Such were: feature pyramids [76], deformable convolutions [75], separated multiple classifiers for classes (MG-CNN) [21], SVMs for classification of DNN features [99]. Few-shot learning was attacked by label propagation [102, 104] and prototyping approaches [101, 106]. Now we show that state-of-the-art deep neural networks are capable of an almost-perfect classification of steel surface defects on the two benchmark datasets. Then, we discuss the problem of few-shot learning and introduce our proposed architecture.

4.5.1 Classification of Defects with EfficientNet-B7

There is a large number of DNNs for object recognition or classification and to compare their performance different benchmark datasets are created, such as ImageNet [65], MNIST [109], CIFAR-10, and CIFAR-100 [110]. If one network has high performance on some dataset it has high-probability for good performance on others, however, the different characteristics of data from different domains does not guarantee this. In 2019, Google Brain published open source EfficientNet [71], as a family of image classification models to achieve state-of-the-art accuracy, yet still being an order-of-magnitude smaller and faster than previous models. The members of the family are the differently scaled versions (from B0 to B7) of a base model. The largest variant (B7) achieved state-of-the-art top-one accuracy on ImageNet in 2019, and it could reach the same performance as the previous state-of-the-art model but being 8.4 times smaller and 6.1 times faster during inference. In [71], a compound scaling method was introduced to scale the depth (number of layers),

the width (number of kernels in a layer), and resolution (size of input image) of an existing model and a baseline network with fine-tuned layers in a balanced manner considering the computation limits. They followed the idea that a small model can be easily fine-tuned on a small problem, and that a systematic scaling, changing only the dimensions but not the main structure and operation of layers, should result in an efficient network. Compound scaling proposes the right ratio between dimensions, while the baseline network (EfficientNet-B0) was designed with the help of optimization [111] considering both accuracy and computational cost on ImageNet.

The building blocks of the base EfficientNet model, obtained through a neural architecture search, are the so-called mobile-inverted bottleneck [112] layers, originally developed to be used as mobile-size networks (networks able to operate in mobile and embedded devices). These blocks use well-proven existing techniques, such as residual connections and depth-wise separable convolutions, and on the other hand also apply the special-techniques-inverted residuals and a linear bottleneck. The interested readers can find details about these in paper [112].

Since its introduction, there is a very wide range of applications where EfficientNet or its variants showed very good accuracy, e.g., mushroom recognition [113], skin disorder recognition [114], iris recognition [115, 116], forest fire detection [117], steganalysis [118], or plant disease detection [119], just to mention a few recent articles. Our experiments, detailed in Section 4.6, show that EfficientNet-B7 outperforms all previously published architectures for both NEU and X-SSD datasets. Moreover, in the next subsection we use EfficientNet-B7 (pre-trained on ImageNet) as the backbone of our proposed architecture for few-shot learning.

4.5.2 Fast Few-Shot Learning of Steel Surface Defects with Randomized Network

As automation is spreading widely in manufacturing processes, the need for automatic anomaly detection is growing. However, if we trained our machine intelligence for a given task, there is always a non-zero probability that unseen events might happen that the system is not trained to handle yet. A part of this problem is few-shot learning, where new kinds of errors appear to be classified as soon as possible, typically with a very low number of training samples. We have to carry out incremental learning, since previously trained knowledge should not be forgotten. Moreover, the retraining of the whole architecture could be resource-demanding: the large amount of time, memory, and processing power is typically not available on site or in time. In a new approach, we answer these challenges by the combination of two very efficient neural network architectures. The solution, illustrated by Figure 4.5, is composed of two parts :

1. The backbone is a deep neural network with an output feature vector of relatively large dimension (1024). We have chosen EfficientNet-B7 for this task due to its design for optimal size, structure, and accuracy. The task of the backbone is to adapt from ImageNet weights to steel surface defects with high accuracy with conventional training;

2. The back-end structure, for the further processing of extracted features, is a two-layer neural network where the first layer contains random weights. Since the backbone is previously fine-tuned to a set of known classes of steel surface defects, the task of this back end is to quickly learn the new classes based on the features of the backbone.

The reasoning of using this architecture is as follows:

- EfficientNet-B7 is very efficient for the classification of steel surface defects (see Section 4.6). Near its output, it still has a lengthy (hopefully rich) feature vector used by our back end;
- The back-end has a random layer responsible for the generalization of fine-tuned feature values suitable for learning unseen classes;
- Since the back-end has only one layer to be trained, it can be explicitly computed with algebraic computations without a lengthy backpropagation method. These computations give optimal solutions in the least-squares sense.

4.5.2.1 Randomized Weights for Generalization and Fast Tuning

It is well known that randomizing weights in neural networks can result in improved accuracy. In paper [120], the input data of a single (hidden)-layer feed-forward neural network (SLFN) were weighted with random weights. Then, in the output layer, a fully connected network with bias was applied, where its weights could be calculated by solving a linear set of equations by standard numerical methods. In [121] the random vector functional links network (RVFL) was proposed, where beside the input patterns, their randomized version is also generated and fed to the output using the standard weighted connections. Extreme learning machines (ELM) [122] have subtle variations to these randomized networks (no direct connection between inputs and outputs like RVFL, other usage of bias than in [120]) but became more popular recently, in spite of reports that RVFL gives better performance than ELM in some cases [123]. For more information we propose to read [124], a review on neural networks with random weights.

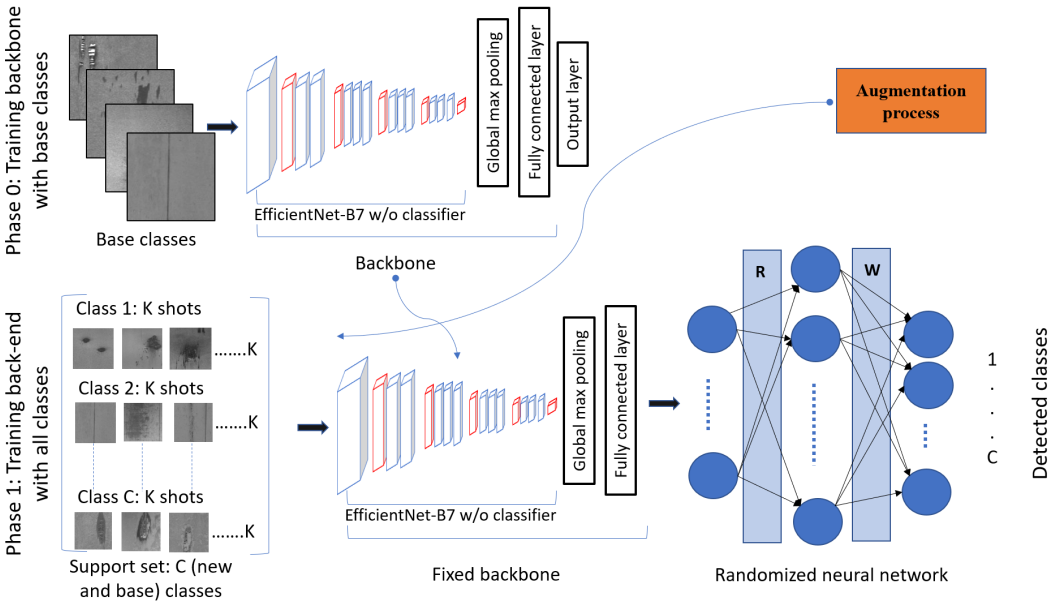

Figure 4.5: The proposed architecture (EffNet+RC) for few-shot learning. In phase 0, we train the backbone through a large number of samples of base classes. In phase 1 (and further phases), we use features extracted by the previously trained backbone. Here, only the weights W are computed with the help of a few samples, while weights R are random and fixed.

A formal description of the applied randomized back-end network follows. Consider a set of N distinct training samples (x_i, y_i), $i = 1, ..., N$. Then, a SLFN with L hidden neurons has the following output equation:

$$\mathbf{t}(x_i) = \sum_{j=1}^{L} \mathbf{w}_j \phi(\mathbf{r}_j \mathbf{x}_i + b_j), \tag{4.1}$$

where ϕ is a sigmoid function, $\mathbf{r}_j$ are the random and fixed-input weight vectors, b_j are the biases, $\mathbf{w}_j$ are the output weight vectors to be tuned, and $\mathbf{t}$ is the target vector (the outputs for the different classes). $\mathbf{R}$ and $\mathbf{W}$ in Figure 4.5 correspond to the weights here. Now, let us see how to compute $\mathbf{W}$ if $\mathbf{R}$ is set randomly. In practice, closed-form solutions are used to find $\mathbf{w}_j$ in a matrix form. Thus, we can shorten the equation:

$$\mathbf{T} = \mathbf{HW}, \tag{4.2}$$

as the outputs of all hidden neurons are gathered into the matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} \phi(\mathbf{r}_1\mathbf{x}_1 + b_1) & \cdots & \phi(\mathbf{r}_L\mathbf{x}_1 + b_L) \\ \vdots & \ddots & \vdots \\ \phi(\mathbf{r}_1\mathbf{x}_N + b_1) & \cdots & \phi(\mathbf{r}_L\mathbf{x}_N + b_L) \end{bmatrix}, \tag{4.3}$$

given $\mathbf{W} = \left(\mathbf{w}_1^T \cdots \mathbf{w}_L^T\right)^T$, and $\mathbf{T} = \left(\mathbf{t}_1^T \cdots \mathbf{t}_C^T\right)^T$.

A unique solution for this system can be given by using the Moore–Penrose generalized inverse (pseudoinverse) [125] of the matrix H, denoted as H†. From Equation 4.2:

$$\mathbf{W} = \mathbf{H}^{-1}\mathbf{T}. \tag{4.4}$$

To find the "best fit" (least squares) solution to the system of linear equations the pseudoinverse is computed [125]:

$$\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T. \tag{4.5}$$

Finally, we get the weights:

$$\mathbf{W} := \mathbf{H}^{\dagger}\mathbf{T}. \tag{4.6}$$

This explicit calculation of weights enables us to achieve an instantaneous fine-tuning of the architecture for new incoming classes in incremental learning.

4.5.2.2 Randomizing EfficientNet Features

Unfortunately, a neural network with only one hidden layer cannot cope with deep learning networks regarding accuracy. Besides, we would like to solve the problem of few-shot learning, where we have not enough information to train our network to the new classes. What we can do, is to transfer some information from the same domain, generalize it, and use this information for the classification of the new

classes. For this transfer, we propose to use the randomized back-end network to generalize previously learnt information (indirectly the feature extractors) stored in a deep backbone network.

Basically, this can be considered as a multi-phase incremental learning mechanism. In phase 0, we fine-tune a large network (based on EfficientNet-B7) with lots of samples of the base classes. This backbone network is constructed by leaving the classification block of EfficientNet-B7, adding a global max-pooling layer, and two fully connected layers. The first has 1024 neurons, the second is the fully connected output layer with the number of classes to be trained. After training it with the available samples, this network, without the output layer, is frozen and used in the next phase as the backbone. Phase 1 is an incremental step, where we fine-tune only the back-end randomized network given the 1024 feature vectors as input. Naturally, phase 1 can be repeated many times, as new shots (or classes) appear.

More formally, suppose we have the set of classes $C = \{C^B, C^1, ..., C^P\}$, where C^i denotes the new classes (or equivalently class labels) appearing at step i, P is the maximum number of steps in incremental learning, and B denotes the base classes with lots of samples available ($C^B = C^0$, and preferably $||C^B|| \gg ||C^i||$, for $i \neq B$). Correspondingly, $D^B, D^1, ..., D^P$ represent the training datasets, where $D^i = (x_s^i, y_s^i)$ $(s = 1, ..., N^i)$, and x_s^i is the s-th example in training phase i, and y_s^i is its corresponding class label.

Algorithm 2: Incremental training with few shots with randomized EfficientNet features.

Input: Training data in phases: $D^0, D^1, D^2, ..., D^P$; EfficientNet-B7 network; randomized SLFN classifier

Output: Trained backbone and back-end randomized classifier models

```
i = 0
Phase i: Train EfficientNet-B7 for base classes C^i by samples in D^i
Cut top layer from EfficientNet-B7 to create backbone
while i ≤ P do
    i = i + 1
    Randomly sample D^i and make the augmentation of samples
    Use the pre-trained backbone to extract latent features for x_s^i
    Train the randomized SLFN as given in Equation (4.6)
```

In phase 0, we train the backbone model for classes C^B. In later phases, we create a support set $D^i = (x_s^i, y_s^i)$, such that $y_s^i \in \bigcup\{C^l\}, l \leq i$.

While Algorithm 2 gives the pseudo-code of this process, parameters are given in Section 4.7. Please note that the incremental phase (or phases) can introduce new classes, not only shots.

The combination of EfficientNet-B7 backbone and the randomized classifier (RC) is named **EffNet+RC** in this chapter.

4.6 Experimental Results of Classification and Discussion

In case of both datasets we used the same hyper-parameters: The training was running for a maximum of 200 epochs, and we used early stopping to avoid over-fitting. For the loss function we have chosen categorical cross-entropy, the learning rate was set to 0.0001 in all cases.

To enrich the dataset we applied traditional augmentation by applying the following online image transformations:

- random rotations between 0° and 30°;
- vertical flipping;
- horizontal flipping;
- zooming randomly between 0 and 20% in size.

The description of our hardware and software configuration is given in Table 4.2.

Table 4.2: Running and testing environment.

OS	CPU	GPU	Keras	Python
Ubuntu 18.04	Intel(R) Xeon(R) Gold 5115 CPU @ 2.40 GHz	NVIDIA Quadro P6000 GPU with 24 GB RAM	2.3.0	3.6.9

4.6.1 Classification Results on the NEU Dataset

To evaluate the performance of classification, we compare EfficientNet-B7 with several state-of-the-art algorithms. Training and testing sets were defined as given in

Section 4.3. Previously, the best known accuracy on the NEU dataset was achieved by our variant of VGG16 [78], denoted as ExtVGG16, see Section 4.4. Table 4.3 shows the comparison of all known alternatives. All results are very close to perfection: EfficientNet-B7 resulted in 100% accuracy, as well as ExtVGG16. Although, the number of parameters of ExtVGG16 is 53 million compared to the 66 million of EfficientNet-B7.

Table 4.3: Comparison of the classification accuracy of different models on the NEU dataset. If not specified then information is based on [98]. Training/testing ratio is 80/20 in general. Best values are highlighted in bold.

Model	MMG -CN[1] [102]	SBF -Net	ResNet50 +MFN	Res-Net50	MVM -VGG	Res-Net34	Decaf	VSD[2] [79]	ResNet43 +MFN	AECLBP	OVER -FEAT	Classic ResNet50	BYEC	ExtVGG -16 [78]	Efficient -Net-B7 [71]
Accuracy	99.72%	99.72%	99.67%	99.67%	99.5%	99.33%	99.27%	89.17%	99.17%	98.93%	98.7%	98.67%	96.3%	**100%**	**100%**

[1] A quantity of 40% of images used for testing, 60% for training. [2] A total of 50 images per class used for testing, remaining images for training.

4.6.2 Classification Results on the X-SSD Dataset

In the case of X-SSD, we followed the same steps as described in papers, [72, 78], regarding the division for training and testing: 70% (952) images were used for training and the remaining 30% (408) for testing. The augmentation was identical to the above described, as well as other parameters of the training process.
The best previous results were also produced by ExtVGG16 [78], (a little below 100%), and could be now beaten slightly with EfficientNet-B7. Table 4.4 shows, beside accuracy, the computed macro-precision, macro-recall, and macro-F1 values for 14 different methods, including EfficientNet-B7. Macro-values refer to the averaging of class averages. We also included the confusion matrix and a misclassified example for EfficientNet-B7 in Figure 4.6.

We can conclude that the NEU dataset seems to be too easy since many methods reached over 99% accuracy and both ExtVGG16 and EfficientNet-B7 could result in perfect classification. There is a larger gap between the accuracy of the different

Table 4.4: Comparison of EfficientNet-B7 with other models on X-SSD (all data are based on paper [72] except for ExtVGG16 [78] and EfficientNet-B7). Best values are highlighted in bold.

Model	Esp -Net-v2	Ghost -Net	Shuffle -Net	Squeeze -Net	Xception	VGG16 [51]	ResN -et50	ResN -et101	ResNe -et152	RepVGG B1g2	RepVGG B3g4	RepVGG B3g4+SA	ExtV -GG16	Efficient -Net-B7
Accuracy	89.95%	88.72%	87.50%	91.42%	90.44%	92.65%	93.87%	87.01%	92.16%	88.97%	91.67%	95.10%	99.00%	**99.26%**
Macro-Recall	84.19%	87.87%	85.84%	83.21%	87.39%	90.46%	89.41%	88.30%	89.41%	82.04%	85.28%	93.92%	98.00%	**98.71%**
Macro-Precision	88.28%	86.93%	84.83%	90.36%	89.41%	91.70%	93.45%	88.18%	91.41%	90.79%	88.46%	95.16%	99.00%	**99.14%**
Macro-F_1 score	84.28%	87.07%	84.68%	84.15%	88.25%	90.92%	90.02%	87.05%	89.92%	81.58%	84.94%	93.25%	98.57%	**99.00%**

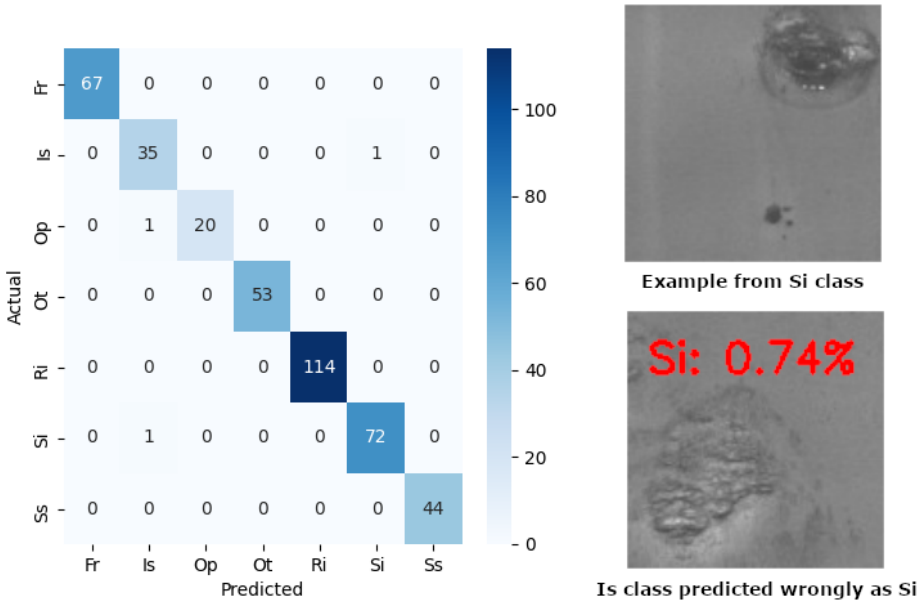

Figure 4.6: (**Left**) Confusion matrix of EfficientNet-B7 classification of the seven error types of the X-SSD dataset. (**Right**) An example image of the Si class and an Is defect wrongly classified as Si.

methods on X-SSD, where EfficientNet-7 showed the best accuracy.

4.7 Testing Few-Shot Learning with EffNet+RC

In these experiments we are implementing two phases of class increments. In phase zero, we train four classes of artifacts (finishing-roll printing, oxide scale-of-temperature system, red iron sheet, inclusion), then, in phase one, we are testing

both the four base and the three new classes (iron sheet ash, scratches, oxide scale-of-plate system). In further phases we do not increase the number of classes, only the number of shots (K). When configuring the datasets for training and testing the few-shot incremental models, we had to make a different setup than for the previous classifications. Since in one-shot learning there is no sample for the validation of the learning process, so we cannot apply an early stopping mechanism, we run each process for 100 epochs, and alternatively, for 200 epochs. Table 4.5 summarizes how X-SSD was cut into parts for training and testing purposes for the 4 base and the 3 incremental classes.

Table 4.5: The distribution of original X-SSD images in the training and testing datasets for few-shot learning. K is the number of shots in the experiments. The real number of images fed to the network during training is larger due to augmentation.

Model	**All Images of X-SSD: 1360**			
	Training: 1092		**Testing: 268**	
	Base Classes: 835	**New Classes: 257**	**Bases Classes: 206**	**New Classes: 62**
EffNet backbone	All	None	None	None
EffNet+RC	K shots	K shots	All	All
EffNet+FtC	K shots	K shots	All	All
Ft EffNet	K shots	K shots	All	All
Ft EffNet Unbal	All	K shots	All	All

When training any variant of EfficientNet we applied the same random augmentation of images in each epoch as it was described before. Since **EffNet+RC** does not have epochs we applied augmentation to have 700 extra training images to build up the matrix $\mathbf{H}$ in Eq. 4.3. Additionally, we tried different numbers for hidden neurons for randomizing the network, we found that the optimal hidden neurons was 20,000.

To evaluate the accuracy of the proposed architecture (**EffNet+RC**), we compared it with different alternatives. Since EfficientNet-B7 showed the highest accuracy for the classification of the seven classes, it was natural to use it as a reference. The

following variations are compared:

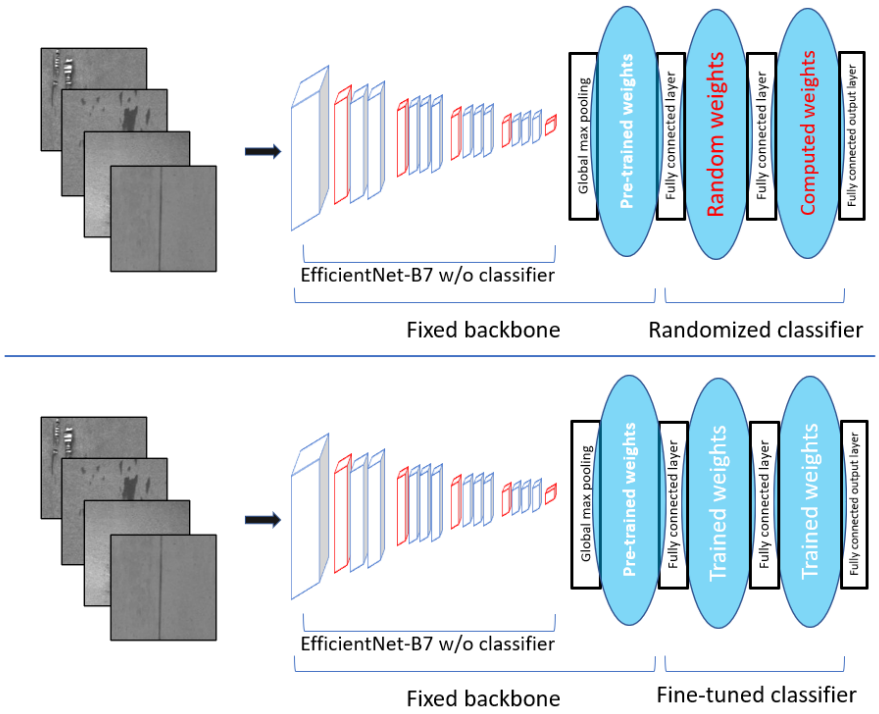

Figure 4.7: The illustration of the **Effnet+RC** and the **EffNet+FtC** networks. While the structures are similar, there is a big difference as the former should not be trained with backpropagation.

- **EffNet+RC**: Fixed EfficienNet-B7 backbone (in phase zero, see Figure 4.5) plus randomized classifier. To train this network we can use the explicit formula of Eq. 4.6;

- **EffNet+FtC**: This network only differs from **EffNet+RC** in that instead of random weights, backpropagation fine-tuned weights are used in the classifier, and the backbone is still frozen see Figure 4.7. The purpose of this network is to learn the effect of randomization (when compared to **EffNet+RC**). We ran the training for 100 and 200 epochs;

- **Ft EffNet**: Fine-tuned EfficienNet-B7 by a few shots (being augmented). To

keep the fine-tuning dataset balanced the ratio of base classes is the same as of the new ones;

- **Ft EffNet Unbal**: Fine-tuned EfficientNet-B7 with unbalanced data. The same as above, but possibly all samples from the base classes were used in fine-tuning. This means unbalanced training, since new classes were sampled only by a few shots.

The accuracy, measured on all (base and new), and also separately on base and new classes, are shown in Figures 4.8, 4.9, and 4.10, correspondingly. To investigate the effect of increasing the number of shots, K was set to 1, 3, 5, 10, 15, and 20. All experiments were repeated five times, randomly choosing the training shots, and averaging the results. All test images were used in the accuracy measurements (as indicated in Table 4.3).

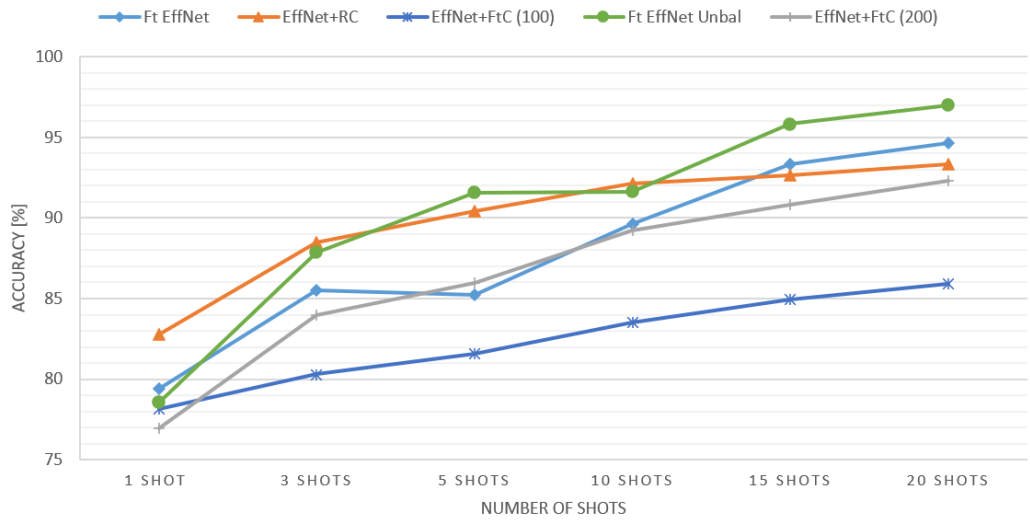

Figure 4.8: The accuracy of the different classification models for all classes (base and new). Ft EffNet: fine-tuned EfficienNet-B7; EffNet+RC: fixed EfficienNet-B7 backbone plus randomized classifier; EffNet+FtC: fixed EfficienNet-B7 backbone plus fine-tuned classifier; Ft EffNet Unbal: fine-tuned EfficienNet-B7 with unbalanced dataset.

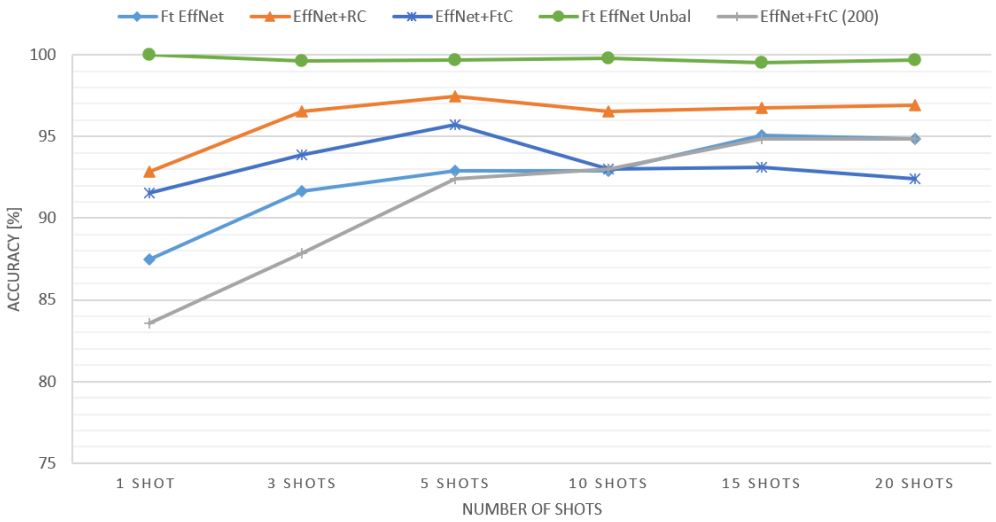

Figure 4.9: The accuracy of the different classification models for base classes. Ft EffNet: fine-tuned EfficienNet-B7; EffNet+RC: fixed EfficienNet-B7 backbone plus randomized classifier; EffNet+FtC: fixed EfficienNet-B7 backbone plus fine-tuned classifier; Ft EffNet Unbal: fine-tuned EfficienNet-B7 with unbalanced dataset.

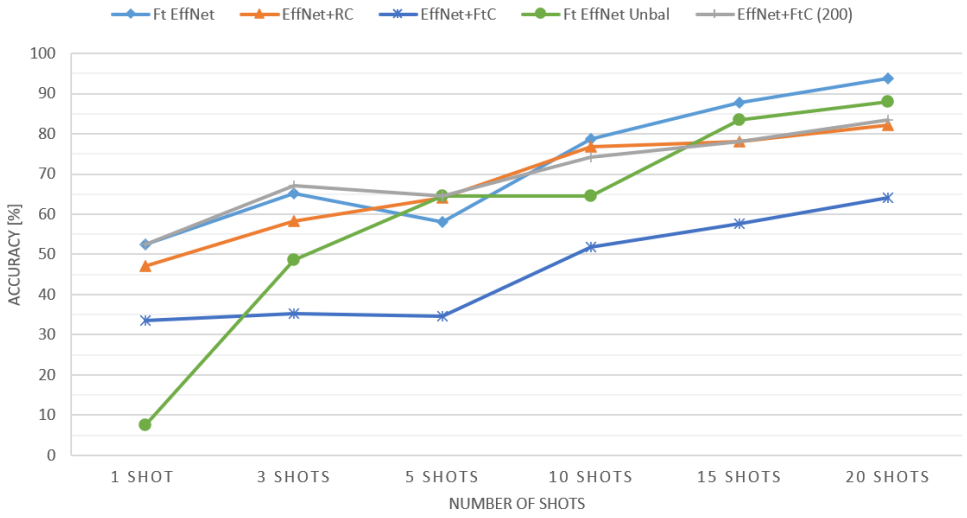

Figure 4.10: The accuracy of the different classification models for new classes. Ft EffNet: fine-tuned EfficienNet-B7; EffNet+RC: fixed EfficienNet-B7 backbone plus randomized classifier; EffNet+FtC: fixed EfficienNet-B7 backbone plus fine-tuned classifier; Ft EffNet Unbal: fine-tuned EfficienNet-B7 with unbalanced dataset.

When talking about few-shot incremental learning, in some scenarios, only few samples from the new and base classes are available for re-training. In our tests, except for **Ft EffNet Unbal**, the same number of shots were used from all classes for re-training. One would first expect that **Ft EffNet** could have the highest accuracy, since the whole network is re-trained, but this is not the case. While it can quickly adapt to the new classes, it gives the worst performance for the old ones (see corresponding curves in Figures 4.9 and 4.10). This is similar to catastrophic forgetting, and the drop from almost 100% to 87.5–93% is significant and the worst among all. Contrarily, if we involve more samples from the base classes to the re-training (and thus generate an unbalanced training dataset), the performance on old classes remains almost 100% but gives quite bad results for one to three shots of the new classes. It is illustrated by the green curve (**Ft EffNet Unbal**) in Figure 4.10. Now, compare the proposed **EffNet+RC** with **EffNet+FtC**. These two networks are similar, except for the fact that the later does not have a randomized fully connected layer, but all of its weights are trained by backpropagation (see Figure 4.7 for illustration). As can be read out from Table 4.6 and Figure 4.8, **EffNet+RC** outperformed not only **EffNet+FtC (100)** and **EffNet+FtC (200)** but all other models at $K = 1, 3, 10$. At a higher number of shots, it still remained in the mid-range. Our proposed randomized model behaved quite well for the old classes; meanwhile, it could nicely follow the best curve of the fine-tuned EfficientNet-B7 (**Ft EffNet**) for the new ones.

Table 4.6: The accuracy and rank of the different classification models evaluated on all classes (base and new). Best values are highlighted in bold

Method	1 Shot		3 Shots		5 Shots		10 Shots		15 Shots		20 Shots	
	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank	Accuracy	Rank
EffNet+RC	**82.75%**	1	**88.50%**	1	90.39%	2	**92.14%**	1	92.62%	3	93.31%	3
EffNet+FtC (100)	78.13%	4	80.29%	5	81.56%	5	83.50%	5	84.92%	5	85.89%	5
EffNet+FtC (200)	76.94%	5	83.95%	4	85.97%	4	89.25%	4	90.82%	4	92.31%	4
Ft EffNet	79.40%	2	85.51%	3	85.24%	3	89.62%	3	93.35%	2	94.62%	2
Ft EffNet Unbal	78.58%	3	87.83%	2	**91.56%**	1	91.63%	2	**95.81%**	1	**97.01%**	1

Beside the accuracy values in Table 4.6, we included the weighted F_1 score and rank of the different classification models evaluated on all classes (base and new) in Table 4.7. The position of the proposed approach remained the same as was in Table 4.6.

Table 4.7: The weighted F_1 score and rank of the different classification models evaluated on all classes (base and new). Best values are highlighted in bold

Method	1 Shot		3 Shots		5 Shots		10 Shots		15 Shots		20 Shots	
	F_1 Score	Rank	F_1 Score	Rank	F_1 Score	Rank	F_1 Score	Rank	F_1 Score	Rank	F_1 Score	Rank
EffNet+RC	**82.4%**	1	**87.00%**	1	89.20%	2	**92.20 %**	1	92.20%	3	92.6 %	3
EffNet+FtC (100)	76.8 %	4	77.4 %	5	78.6 %	5	82.4 %	5	84.6 %	5	85.6 %	5
EffNet+FtC (200)	78.75 %	2	86.00 %	3	85.5 %	3	90.5 %	3	91.5 %	4	92.5 %	4
Ft EffNet	78.20 %	3	85.8 %	4	83.4 %	4	89.8 %	4	92.6 %	2	94.80 %	2
Ft EffNet Unbal	70.8 %	5	86.2 %	2	**90.8 %**	1	90.8%	2	**95.8 %**	1	**96.8 %**	1

4.7.1 Time Complexity

While the average inference time for an image of size 128×128 is around 0.06 s, the time complexity of re-configuring, re-training, or fine-tuning the classification methods can still cause problems in industrial applications. Stopping the production while waiting for adaptation to avoid new defects, or continuing the production with an increased ratio of false products, can both result in high costs. Due to these reasons, the extremely fast re-tuning of our model has a significant advantage over others. Table 4.8 and Figure 4.11 both contain the training times of the different models. **EffNet+RC** has constantly very low time requirements, independently from the number of shots, at least one order faster than **EffNet+FtC (100)**, the next fastest. The reason why **Ft EffNet Unbal** seems to be so slow is due to the large number of training images in each epoch $(835 + k(shots) \times 3(new\ classes))$, while in case of **Ft EffNet**, the training set contained far less images (since it was balanced).

Table 4.8: Time spent for training at different number of k-shots.

Method	1 shot	3 shots	5 shots	10 shots	15 shots	20 shots
EffNet+RC	2.80s	3.01s	3.03s	3.04s	3.30s	3.50s
EffNet+FtC (100)	29.24s	29.49s	45.67s	53.9s	67.41s	78.09s
EffNet+FtC (200)	39.53s	39.83s	69.70s	88.29s	114.18s	136.35s
Ft EffNet	151.80s	159.80s	193.43s	243.85s	293.50	347.23s
Ft EffNet Unbal	1408.10s	1429.50s	1443.30s	1473.20s	1489.10s	1496.60s

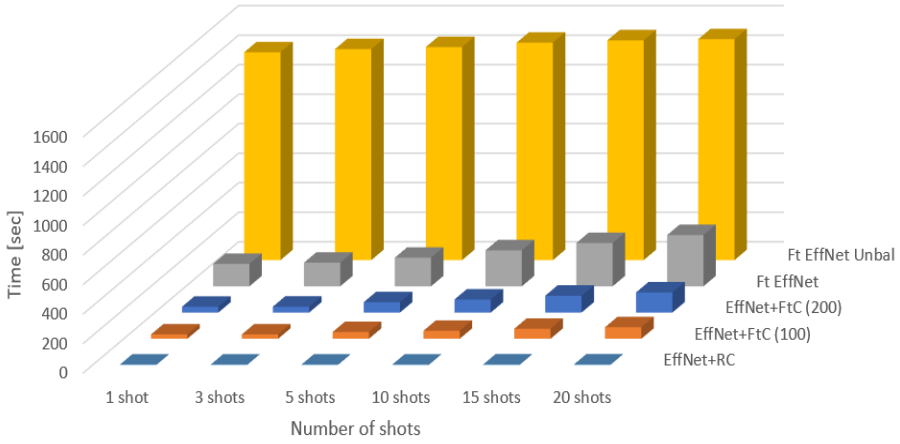

Figure 4.11: Elapsed training time of the different models under investigation.

4.8 Summary

In this chapter we showed that state-of-the-art DNNs (namely EfficientNet-B7) can solve the classification of steel surface defects, of the two often used datasets, almost perfectly and superior to other made-to-measure techniques. Besides, we have given a technique to handle common but inconvenient problems such as incremental and few-shot learning. Regarding very few-shots (1-3), our model outperformed other variants when classifying both old and newly appearing classes of steel surface defects, while it showed good performance for higher number of shots. Regarding the base classes, catastrophic forgetting could also be avoided. An important practical feature is that the fine-tuning for new classes or shots is significantly faster than the fine-tuning of any conventional DNNs. Our statements are validated with thousands

of experiments on steel surface defects. In the future, we plan to further improve the model by sub-network extensions, similarly to [94], and to investigate the model using other kinds of data.

We can summarize the main contributions of this chapter in the following points:

1. We showed that state-of-the-art DNNs (namely EfficientNet-B7) can solve the classification of steel surface defects of the two often-used datasets almost perfectly, and that they were superior to other made-to-measure techniques;

2. We applied randomized networks, concatenated to the feature extraction of a pre-trained DNN, to give a solution for the few-shot problem. Since the computation of its weights can be done very efficiently by the Moore–Penrose generalized inverse, the solution has the following advantages:

    - Regarding very few-shots (one to three), our model outperformed other variants when classifying both old and newly appearing classes of steel surface defects;
    - Regarding the base classes, catastrophic forgetting could be avoided;
    - The fine-tuning for new classes or shots is significantly faster than the fine-tuning of any conventional DNNs.

3. Additionally, we proposed a zero-shot approach based on the siamese network for the recognition of new defect classes and also for the classification of known types appearing on steel surfaces. We utilized one branch of the siamese network for the classifications of previously trained defects. For performance evaluations, experiments were carried out on two large benchmark datasets. Our architectures outperformed all previously tasked techniques and reached almost 100% accuracy on the NEU dataset and 98% accuracy on the X-SDD

dataset. For zero-shot learning the accuracy is 85.80% and 61.55% respectively on both datasets.

This chapter was summarized in Thesis III, please see, Chapter 5, Section 5.1.

Chapter 5

Conclusion

This chapter aims to discuss and draw conclusions from the results in order to answer the stated research questions.

In this thesis we dealt with the recognition of objects and detection of their defects. By using statistical models such as HMM and IMUs data, we could improve the convolutional neural networks performance for object recognition and pose estimation. We show that even if the neural networks could not be optimally trained and region-based detection is not possible, they can still be used for multiple-view object recognition with the help of information fusion. We show that relative pose changes can give enough information for the HMMs to estimate the most probable state sequences and thus finding the most probable object visible on the images. Additionally, we showed how active perception and information fusion can help the recognition of 3D objects and partially occluded objects in a HMM framework if only weak classifiers are applied. The possible application of such approaches can be important in embedded systems or if sensors with limited resources are to be used for example in future's autonomous, wearables or IoT devices. The proposed HMM technique is computationally lightweight, requires limited memory and can incorporate other classifiers, not only the presented CEDD.

Additionally, we dealt with two common domains in visual inspection to identify faults in objects (traffic signs and steel surface defects).

We investigated different methods to recognize defects of traffic signs. First, a new siamese neural network architecture (FCSNN) was proposed to recognize the defects of different objects. The previously proposed networks were extended by several layers and the original features, besides computing the difference, were retained for fully connected layers. Moreover, we test the generalization properties of our network to learn the latent defect specific features by predicting the errors of new untrained object classes with different appearance. Second, we proposed a new mechanism to combine the output confidence values of FCSNN siamese networks with SVM with the help of support set images to improve the recognition accuracy. The advantage of our approach, compared to the concept of ensemble of networks, is that only one network is to be trained and maintained (one model can handle many traffic sign and error classes.) Moreover, we introduced a new data-set (TSD) of defected traffic signs that can be very useful for evaluating anomaly detection (e.g. for maintenance purposes) and for the research community to test the error prone abilities of traffic sign recognition methods.

For steel surface defects, we showed that state-of-the-art DNNs (namely EfficientNet-B7) can solve the classification of steel surface defects of the two-benchmark used data-sets, almost perfectly and superior to other made-to-measure techniques. Besides, we have given a technique (based on EfficientNet with Randomized Classifier) to handle common but inconvenient problems such as incremental and few-shot learning. Regarding very few-shots (1-3), our model outperformed other variants when classifying both old and newly appearing classes of steel surface defects, while it showed good performance for higher number of shots. Regarding the base classes, catastrophic forgetting could also be avoided. An important characteristic from

practicality, that the fine-tuning for new classes or shots is significantly faster than the fine-tuning of any conventional DNNs. Finally, for a zero-shot learning, an approach based on a siamese network, was proposed.
Below, besides the scientific results, we also give the list of journal and conference publications related to this work.

5.1 New Scientific Results

In this thesis, we have studied different solutions to solve the problem of recognizing objects and their defects. The following theses summarize the new scientific results in three main points. We also give the corresponding chapters and publications.

Thesis I: Object Recognition Techniques using Deep Neural Networks and HMMs

Object recognition only in the last few years has made a significant improvement with the evolution of neural networks. While convolutional neural networks are very efficient in object recognition there is still need for improvements in many practical cases. For example if the performance from single images is not satisfactory, natural ambiguities (such as noise, occlusion and geometrical distortions) and the requirements of the computational and memory are high. Additionally, to interact with the objects of the environment, not only specific or generic object recognition is inevitable, but the determination of their pose is also essential.

1. **I proposed a framework to show that HMMs can be used to combine the data of CNNs and IMUs for object recognition and pose estimation**. I showed that relative pose changes can give enough information for the HMMs to estimate the most probable state

sequences and thus find the most probable object visible on the images and improve the performance. To show the efficiency of the proposed method, we compared the results with VGG16 network.

2. **HMM based knowledge can be improved by integrating it with an active vision approach, the proposed framework is called AV-HMM-CEDD**. I showed that how active perception and information fusion from sequential multiple shots can help the recognition of 3D objects even if the objects are occluded. Experimental results shows that our proposed method can handle much better the untrained occluded queries if we compare it with DNNs. Moreover, the proposed AV-HMM-CEDD framework is computationally lightweight, requires limited memory and can incorporate other classifiers, not only the presented CEDD.

This thesis is explained in detail in Chapter 2 and the related publications are **AM1**, **AM2**, **AM3**.

Thesis II: Detection of Traffic Sign Defects

There is no doubt that traffic signs are very important parts of the road infrastructure for vehicles. However, detection and recognition of traffic signs with high accuracy is still an unsolved problem, especially in real-life conditions. Beside unfavorable weather, lighting, and imaging conditions the unwanted defects of traffic signs may heavily affect the accuracy of such systems. It is inevitable to develop systems to monitor the state of traffic signs, by the detection of the different errors, supporting their maintenance.

1. **I proposed a new siamese neural network architecture to recognize the defects of a large number of classes of traffic signs**. The previously proposed networks were extended by several layers and

the original features, besides computing the difference, were retained for fully connected layers. Experimental results show that this approach can be applied to recognize anomalies in images with better performance for well-known feature learning approaches. Additionally, it is possible to use it to recognize defects in untrained types of objects.

2. **To improve the performance, I proposed a new mechanism to combine the confidence values of siamese networks with SVM with the help of support set images**. The advantage of our approach, compared to the concept of ensemble of networks, is that only one network is to be trained and maintained.

This thesis is explained in detail in Chapter 3 and the related publications are **AM4**, **AM5**.

Thesis III: Classification, Zero and Fast Few-Shot Learning of Steel Surface Defects

Steel is the most important metal in terms of quantity and variety of applications in the modern world. Due to the manufacturing process and environmental conditions, steel surfaces can have a variety of defects. In most applications, beside the questions of detection and classification, the following key problems are to be answered: real-time problem, small target problem, small sample problem, unbalanced sample problem. Moreover, in incremental learning, new data (i.e. new shots of previously seen or unseen classes) arrive in phases over time and we have to extend our classification model to include these new classes or new samples of classes.

1. **I proposed a new architecture to combine EfficientNet with randomized networks (EffNet+RC) for classification and few-shot**

learning, as well as for continuous learning of steel surface defects. This architecture achieved an effective training time at least an order of magnitude smaller than that of other classifiers. Moreover, it classifies defects with a good performance for a few shots.

2. **To deal with zero shot learning, I proposed a deep architecture of siamese network**. This network is useful for learning features to classify defects or to cluster unseen classes without a single training.

This thesis is explained in details in Chapter 4 and the related publications are **AM6**, **AM7**.

5.2 Publications

Publications of Amr Mohamed Nagy Abdo are listed below.

5.2.1 Publications Related to this Thesis

AM1. Czúni László and **Amr M. Nagy**: Improving object recognition of CNNs with multiple queries and HMMs. In Twelfth International Conference on Machine Vision (ICMV), pages 1143310, 2020.

AM2. Czúni László and **Amr M. Nagy**: Hidden Markov Models for pose estimation. In 15th International Conference on Computer Vision Theory and Applications (VISAPP), pages 1143310, 2020.

AM3. Amr M. Nagy, Metwally Rashad and Czúni László: Active multiview recognition with hidden Markov temporal support. In Signal, Image and Video Processing,pages 315–322, 2021 (**2020-IF: 2.157**).

AM4. Amr M. Nagy and Czúni László: Detecting object defects with fusioning convolutional siamese neural networks. In 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), pages 157–163, 2021.

AM5. Amr M. Nagy and Czúni László: Deep neural network models for the recognition of traffic signs defects. In 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), pages 725-729, 2021.

AM6. Amr M. Nagy and Czúni László: Classification and fast few-shot learning of steel surface defects with randomized network. Applied Sciences; 12(8):3967, 2022. (**2020-IF: 2.67**).

AM7. Amr M. Nagy and Czúni László: Zero-shot learning and classification of steel surface defects. In 14th International Conference on Machine Vision (ICMV), pages 386 - 394, 2021.

5.2.2 Publications not Related to this Thesis

A1. Alejandro R. Rodriguez, **Amr M. Nagy**, Zsolt Vörösházi, György Bereczky, László Czúni: Segmentation and Error Detection of PV Modules, In 27th International Conference on Emerging Technologies and Factory Automation, 2022.

A2. Amr M. Nagy, Ali Ahmed and Hala Helmy Zayed: Hybrid iterated Kalman particle filter for object tracking problems. In 8th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), pages 375–381, 2013.

A3. Amr M. Nagy, Ali Ahmed and Hala Helmy Zayed: Particle filter based on joint color texture histogram for object tracking. In International Image Processing, Applications and Systems Conference,pages 1–6, 2014.

5.2.3 Other Presentations

P1. Czúni László, **Amr M. Nagy**, M. Rashad: About the temporal support of active object recognition, Pannonian Conference on Advances in Information Technology (PCIT 2020), Veszprem, Hungary.

P2. Czúni László, **Amr M. Nagy**, M. Rashad: Low complexity 3D object recognition for mobile devices based on Markovian framework (HMM), in: 12th Conference of Hungarian Association for Image Processing and Pattern Recognition (KÉPAF-2019), Debrecen, Hungary.

P3. Czúni László, **Amr M. Nagy**, M. Rashad: Temporal models for 3D object recognition, in: World Congress on Artificial Intelligence and Machine Learning, (WCAIML-2019), Spain.

Bibliography

[1] Andrew DH Thomas, Michael G Rodd, John D Holt, and CJ Neill. Real-time industrial visual inspection: A review. *Real-Time Imaging*, 1(2):139–158, 1995.

[2] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *ArXiv Preprint ArXiv:1905.05055*, 2019.

[3] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[4] Qi Wu, Chunhua Shen, Peng Wang, Anthony Dick, and Anton Van Den Hengel. Image captioning and visual question answering based on attributes and external knowledge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1367–1381, 2017.

[5] Joseph W Foster, G Kemble Bennett, and Paul M Griffin. Automated visual inspection: Quality control techniques for the modern manufacturing environment. In *Proc. 1987 IIE Integrated Systems Conf*, pages 135–140, 1987.

[6] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[7] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey Vision Conference*, volume 15, pages 10–5244. Citeseer, 1988.

[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[9] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[10] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.

[11] Savvas A Chatzichristofis and Yiannis S Boutalis. CEDD: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval. In *International Conference on Computer Vision Systems*, pages 312–322. Springer, 2008.

[12] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 873–880, 2009.

[13] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *ArXiv Preprint ArXiv:1708.08296*, 2017.

[14] C-C Jay Kuo. Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, 41:406–413, 2016.

[15] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[16] Tamás Czimmermann, Gastone Ciuti, Mario Milazzo, Marcello Chiurazzi, Stefano Roccella, Calogero Maria Oddo, and Paolo Dario. Visual-based defect detection and classification approaches for industrial applications— A SURVEY. *Sensors*, 20(5):1459, 2020.

[17] Dandan Zhu, Ruru Pan, Weidong Gao, and Jie Zhang. Yarn-dyed fabric defect detection based on autocorrelation function and glcm. *Autex Res. J*, 15(3):226–232, 2015.

[18] Junjie Cao, Jie Zhang, Zhijie Wen, Nannan Wang, and Xiuping Liu. Fabric defect inspection using prior knowledge guided least squares regression. *Multimedia Tools and Applications*, 76(3):4141–4157, 2017.

[19] Shan Gai. New banknote defect detection algorithm using quaternion wavelet transform. *Neurocomputing*, 196:133–139, 2016.

[20] Jiaqi Xi, Lifeng Shentu, Jikang Hu, and Mian Li. Automated surface inspection for steel products using computer vision approach. *Applied Optics*, 56(2):184–192, 2017.

[21] Di He, Ke Xu, and Peng Zhou. Defect detection of hot rolled steels with a new object detection framework called classification priority network. *Computers & Industrial Engineering*, 128:290–297, 2019.

[22] Shuang Mei, Hua Yang, and Zhouping Yin. An unsupervised-learning-based approach for automated defect inspection on textured surfaces. *IEEE Transactions on Instrumentation and Measurement*, 67(6):1266–1277, 2018.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[24] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer, 2019.

[25] David Bojanić, Kristijan Bartol, Tomislav Pribanić, Tomislav Petković, Yago Diez Donoso, and Joaquim Salvi Mas. On the comparison of classic and deep keypoint detector and descriptor methods. In *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 64–69. IEEE, 2019.

[26] Nati Ofir and Jean-Christophe Nebel. Classic versus deep learning approaches to address computer vision challenges. *ArXiv Preprint ArXiv:2101.09744*, 2021.

[27] Jonas Gehring, Yajie Miao, Florian Metze, and Alex Waibel. Extracting deep bottleneck features using stacked auto-encoders. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3377–3381, 2013.

[28] Muhammad Attique Khan, Kashif Javed, Sajid Ali Khan, Tanzila Saba, Usman Habib, Junaid Ali Khan, and Aaqif Afzaal Abbasi. Human action recognition using fusion of multiview and deep features: an application to video surveillance. *Multimedia Tools and Applications*, pages 1–27, 2020.

[29] Muhammad Sharif, Muhammad Attique Khan, Farooq Zahid, Jamal Hussain Shah, and Tallha Akram. Human action recognition: a framework of statistical

weighted segmentation and rank correlation-based selection. *Pattern Analysis and Applications*, 23(1):281–294, 2020.

[30] Sumaira Manzoor, Sung-Hyeon Joo, and Tae-Yong Kuc. Comparison of object recognition approaches using traditional machine vision and modern deep learning techniques for mobile robot. In *19th International Conference on Control, Automation and Systems (ICCAS)*, pages 1316–1321, 2019.

[31] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[32] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[33] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[34] László Czúni and Metwally Rashad. The fusion of optical and orientation information in a markovian framework for 3D object retrieval. In *International Conference on Image Analysis and Processing*, pages 26–36. Springer, 2017.

[35] László Czúni and Metwally Rashad. Lightweight active object retrieval with weak classifiers. *Sensors*, 18(3), 2018.

[36] Savvas A Chatzichristofis, Konstantinos Zagoris, Yiannis S Boutalis, and Nikos Papamarkos. Accurate image retrieval based on compact composite descriptors and relevance feedback information. *International Journal of Pattern Recognition and Artificial Intelligence*, 24(02):207–244, 2010.

[37] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[39] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[40] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[41] Tobias Nöll, Alain Pagani, and Didier Stricker. Markerless camera pose estimation-an overview. In *Visualization of Large and Unstructured Data Sets-Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011.

[42] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.

[43] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *ArXiv Preprint ArXiv:1711.00199*, 2017.

[44] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1521–1529, 2017.

[45] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018.

[46] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3828–3836, 2017.

[47] Mengmeng Xu, Yancheng Bai, Bernard Ghanem, Boxiao Liu, Yan Gao, Nan Guo, Xiaochun Ye, Fang Wan, Haihang You, Dongrui Fan, et al. Missing labels in object detection. In *CVPR Workshops*, volume 3, 2019.

[48] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.

[49] László Czúni and Amr M Nagy. Improving object recognition of CNNs with multiple queries and HMMs. In *Twelfth International Conference on Machine Vision (ICMV 2019)*, page 94, 01 2020.

[50] László Czúni and Amr M Nagy. Hidden markov models for pose estimation. In *15th International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 598–603, 2020.

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

[52] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *ArXiv Preprint ArXiv:1804.02767*, 2018.

[53] Sumantra Dutta Roy, Santanu Chaudhury, and Subhashis Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, 2004.

[54] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[55] J. Tang, X. Shu, R. Yan, and L. Zhang. Coherence constrained graph LSTM for group activity recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligences*, 2019.

[56] P. Kulkarni, S. Mohan, S. Rogers, and H. Tabkhi. Key-track: A lightweight scalable LSTM-based pedestrian tracker for surveillance system. In *International Conference on Image Analysis and Recognition*, pages 208–219. Springer, Cham, 2019.

[57] M. Turkoglu, D. Hanbay, and A. Sengur. Multi-model LSTM-based convolutional neural networks for detection of apple diseases and pests. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–11, 2019.

[58] Y. Yuan, X. Liang, X. Wang, D. Y. Yeung, and A. Gupta. Temporal dynamic graph LSTM for action-driven video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1801–1810, 2017.

[59] Y. Lu, C. Lu, and C. K. Tang. Online video object detection using association LSTM. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2344–2352, 2017.

[60] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-Chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810, 2015.

[61] Ben Nassi, Dudi Nassi, Raz Ben-Netanel, Yisroel Mirsky, Oleg Drokin, and Yuval Elovici. Phantom of the adas: Phantom attacks on driver-assistance systems. *Cryptology ePrint Archive*, 2020.

[62] Trivedi S. Povolny, S. Model hacking ADAS to save safer roads for autonomous vehicles, February 2020. [Online; posted 4-May-2022].

[63] Dogancan Temel, Tariq Alshawi, Min-Hung Chen, and Ghassan AlRegib. Challenging environments for traffic sign detection: Reliability assessment under inclement conditions. *ArXiv Preprint ArXiv:1902.06857*, 2019.

[64] Zhongbing Qin and Wei Qi Yan. Traffic-sign recognition using deep learning. In *Geometry and Vision: First International Symposium, ISGV 2021, Auckland, New Zealand, January 28-29, 2021, Revised Selected Papers 1*, pages 13–25. Springer, 2021.

[65] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bern-

stein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[66] Brenden M Lake, Russ R Salakhutdinov, and Josh Tenenbaum. One-shot learning by inverting a compositional causal process. In *Advances in Neural Information Processing Systems*, pages 2526–2534, 2013.

[67] Amr M Nagy and László Czúni. Detecting object defects with fusioning convolutional siamese neural networks. In *16th International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 157–163, 2021.

[68] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2. Lille, 2015.

[69] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[71] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[72] Xinglong Feng, Xianwen Gao, and Ling Luo. X-SDD: A new benchmark for hot rolled steel strip surface defects detection. *Symmetry*, 13(4):706, 2021.

[73] Kechen Song and Yunhui Yan. A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Applied Surface Science*, 285:858–864, 2013.

[74] Yu He, Kechen Song, Qinggang Meng, and Yunhui Yan. An end-to-end steel surface defect detection approach via fusing multiple hierarchical features. *IEEE Transactions on Instrumentation and Measurement*, 69(4):1493–1504, 2019.

[75] Weidong Zhao, Feng Chen, Hancheng Huang, Dan Li, and Wei Cheng. A new steel defect detection algorithm based on deep learning. *Computational Intelligence and Neuroscience*, 2021, 2021.

[76] Haili Zhao, Zefeng Yang, and Jia Li. Detection of metal surface defects based on YOLOv4 algorithm. In *Journal of Physics: Conference Series*, volume 1907, page 012043. IOP Publishing, 2021.

[77] Siyu Zhang, Qiuju Zhang, Jiefei Gu, Lei Su, Ke Li, and Michael Pecht. Visual inspection of steel surface defects based on domain adaptation and adaptive convolutional neural network. *Mechanical Systems and Signal Processing*, 153:107541, 2021.

[78] Amr M. Nagy and László Czúni. Zero-shot learning and classification of steel surface defects. In *Fourteenth International Conference on Machine Vision (ICMV 2021)*, volume 12084, pages 386 – 394. International Society for Optics and Photonics, SPIE, 2022.

[79] Shengqi Guan, Ming Lei, and Hao Lu. A steel surface defect recognition algorithm based on improved deep learning network model using feature visualization and quality evaluation. *IEEE Access*, 8:49885–49895, 2020.

[80] Ihor Konovalenko, Pavlo Maruschak, Vitaly Brevus, and Olegas Prentkovskis. Recognition of scratches and abrasions on metal surfaces using a classifier based on a convolutional neural network. *Metals*, 11(4):549, 2021.

[81] Yajun Chen, Yuanyuan Ding, Fan Zhao, Erhu Zhang, Zhangnan Wu, and Linhao Shao. Surface defect detection methods for industrial products: A review. *Applied Sciences*, 11(16):7657, 2021.

[82] Ari Seff, Alex Beatson, Daniel Suo, and Han Liu. Continual learning in generative adversarial nets. *ArXiv Preprint ArXiv:1705.08395*, 2017.

[83] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *ArXiv Preprint ArXiv:1705.08690*, 2017.

[84] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[85] Eden Belouadah and Adrian Popescu. Deesil: Deep-shallow incremental learning. In *European Conference on Computer Vision*, pages 151–157. Springer, 2018.

[86] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *European Conference on Computer Vision (ECCV)*, pages 233–248, 2018.

[87] Chen He, Ruiping Wang, Shiguang Shan, and Xilin Chen. Exemplar-supported generative reproduction for class incremental learning. In *BMVC*, page 98, 2018.

[88] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCarL: Incremental classifier and representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[89] Yong Luo, Liancheng Yin, Wenchao Bai, and Keming Mao. An appraisal of incremental learning methods. *Entropy*, 22(11):1190, 2020.

[90] Ghouthi Boukli Hacene, Vincent Gripon, Nicolas Farrugia, Matthieu Arzel, and Michel Jezequel. Transfer incremental learning using data augmentation. *Applied Sciences*, 8(12):2512, 2018.

[91] Roger Ratcliff. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285, 1990.

[92] Eden Belouadah and Adrian Popescu. Scail: Classifier weights scaling for class incremental learning. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1266–1275, 2020.

[93] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *IEEE/CVF International Conference on Computer Vision*, pages 583–592, 2019.

[94] Zahid Ali Siddiqui and Unsang Park. Progressive convolutional neural network for incremental learning. *Electronics*, 10(16):1879, 2021.

[95] 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):1–34, 2020.

[96] Aditya M Deshpande, Ali A Minai, and Manish Kumar. One-Shot recognition of manufacturing defects in steel surfaces. *Procedia Manufacturing*, 48:1064–1071, 2020.

[97] Yibo Guo, Yiming Fan, Zhiyang Xiang, Haidi Wang, Wenhua Meng, and Mingliang Xu. Zero-sample surface defect detection and classification based on semantic feedback neural network. *ArXiv Preprint ArXiv:2106.07959*, 2021.

[98] Tobias Schlagenhauf, Faruk Yildirim, Benedikt Brückner, and Jürgen Fleischer. Siamese basis function networks for defect classification. *ArXiv Preprint ArXiv:2012.01338*, 2020.

[99] Adel Boudiaf, Said Benlahmidi, Khaled Harrar, and Rachid Zaghdoudi. Classification of surface defects on steel strip images using convolution neural network and support vector machine. *Journal of Failure Analysis and Prevention*, pages 1–11, 2022.

[100] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[101] Shiqing Wu, Shiyu Zhao, Qianqian Zhang, Long Chen, and Chenrui Wu. Steel surface defect classification based on small sample learning. *Applied Sciences*, 11(23):11459, 2021.

[102] Yucheng Wang, Liang Gao, Yiping Gao, and Xinyu Li. A new graph-based semi-supervised method for surface defect classification. *Robotics and Computer-Integrated Manufacturing*, 68:102083, 2021.

[103] Lei Yang, Xiaohang Zhan, Dapeng Chen, Junjie Yan, Chen Change Loy, and Dahua Lin. Learning to cluster faces on an affinity graph. In *IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 2298–2306, 2019.

[104] Ke Wu, Jie Tan, Jingwei Li, and Chengbao Liu. Few-shot learning approach for 3D defect detection in lithium battery. In *Journal of Physics: Conference Series*, volume 1884, page 012024. IOP Publishing, 2021.

[105] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16, 2003.

[106] Zhu Zhan, Jinfeng Zhou, and Bugao Xu. Fabric defect classification using prototypical network of few-shot learning algorithm. *Computers in Industry*, 138:103628, 2022.

[107] Xiaoming Lv, Fajie Duan, Jia-jia Jiang, Xiao Fu, and Lin Gan. Deep metallic surface defect detection: The new benchmark and detection network. *Sensors*, 20(6):1562, 2020.

[108] Tobias Schlagenhauf, Magnus Landwehr, and Jürgen Fleischer. Industrial machine tool element surface defect dataset. 2021.

[109] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[110] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Department of Computer Science, University of Toronto*, 2009.

[111] George Kyriakides and Konstantinos Margaritis. An introduction to neural architecture search for convolutional networks. *ArXiv Preprint ArXiv:2005.11074*, 2020.

[112] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[113] Norbert Kiss and László Czùni. Mushroom image classification with CNNs: A case-study of different learning strategies. In *12th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 165–170. IEEE, 2021.

[114] Rashidul Hasan Hridoy, Fatema Akter, and Aniruddha Rakshit. Computer vision based skin disorder recognition using EfficientNet: A transfer learning approach. In *2021 International Conference on Information Technology (ICIT)*, pages 482–487. IEEE, 2021.

[115] Mohd Nadhir Ab Wahab, Amril Nazir, Anthony Tan Zhen Ren, Mohd Halim Mohd Noor, Muhammad Firdaus Akbar, and Ahmad Sufril Azlan Mohamed. EfficientNet-lite and hybrid CNN-KNN implementation for facial expression recognition on raspberry pi. *IEEE Access*, 9:134065–134080, 2021.

[116] Hitendra Garg, Bhisham Sharma, Shashi Shekhar, and Rohit Agarwal. Spoofing detection system for e-health digital twin using EfficientNet convolution neural network. *Multimedia Tools and Applications*, pages 1–16, 2022.

[117] Renjie Xu, Haifeng Lin, Kangjie Lu, Lin Cao, and Yunfei Liu. A forest fire detection system based on ensemble learning. *Forests*, 12(2):217, 2021.

[118] Yassine Yousfi, Jan Butora, Jessica Fridrich, and Clément Fuji Tsang. Improving EfficientNet for JPEG steganalysis. In *ACM Workshop on Information Hiding and Multimedia Security*, pages 149–157, 2021.

[119] Fei Gao, Jiming Sa, Zhuoer Wang, and Zhongyu Zhao. Cassava disease detection method based on EfficientNet. In *7th International Conference on Systems and Informatics (ICSAI)*, pages 1–6. IEEE, 2021.

[120] Wouter F Schmidt, Martin A Kraaijveld, Robert PW Duin, et al. Feed forward neural networks with random weights. In *International Conference on Pattern Recognition*, pages 1–1. IEEE Computer Society Press, 1992.

[121] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

[122] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[123] Ponnuthurai N Suganthan and Rakesh Katuwal. On the origins of randomization-based feedforward neural networks. *Applied Soft Computing*, 105:107239, 2021.

[124] Weipeng Cao, Xizhao Wang, Zhong Ming, and Jinzhu Gao. A review on neural networks with random weights. *Neurocomputing*, 275:278–287, 2018.

[125] Eliakim H Moore. On the reciprocal of the general algebraic matrix. *Bull. Am. Math. Soc.*, 26:394–395, 1920.