# Fault Classification in 3D Objects with Missing Parts using Multi-view Images

*Benjamin Provan-Bessell*

Master of Science

Data Science

School of Informatics

University of Edinburgh

2023

# Abstract

Anomaly detection and fault classification is the task of detecting if instances do not fit the norm behaviour or are faulty or 'incorrect'. Fault classification in manufactured objects within industry is an important task to ensure final products are sound and function correctly. This project devises and implements a proof of concept system to automatically detect faults in an object when compared to an image representation of a 3D CAD reference model. This project creates an original dataset of images of 3D models with semantic missing parts, for which the task of missing part detection and classification can be performed. This dataset contains multi-view images of the 3D models such that 3D Computer Vision techniques can be applied to take advantage of the multiple views of each object. Missing part detection can be expressed as a fault classification task. This report introduces a novel technique, a Multi-view Comparison Network (MVCN) that can represent a 3D model as a set of multi-view images, compare it to a query image of an object, and detect if that object is missing a part or not. The dataset and task have particular challenges such as diverse, small parts and similarity learning, and difficult sub-tasks such as 3D learning and inherent pose detection. This project shows empirically that multi-view representations of 3D objects are useful for the fault detection task on query images. The Multi-view Comparison Network outperforms the Single-view Comparison network by 14.7% accuracy and 17.3% precision, and outperforms the single-view CNN baseline classifier by 3.3% classification accuracy and 0.4% precision in the standard fault classification scenario over a range of categories. Further enhancements to the MVCN via aggregated training brought a final accuracy of 81.2% accuracy and 82.5% precision, with a final improvement of 14.7% accuracy and 15.2% precision over the baseline. The MVCN was tested over 3 further test setups: unseen object, unseen anomaly, and difference learning. The successful application of the Multi-view Comparison Network to unseen anomaly detection highlights the model's fault detection ability, and the ability to differentiate if the object in the query image is missing the same part as the reference model in a difference learning test setup shows the model's 3D learning capability.

## Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

## Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Benjamin Provan-Bessell*)

# Acknowledgements

Many thanks to Hakan Bilen for his supervision and guidance throughout this dissertation. I am also grateful for discussions with Ruihui Yan and Wei-Ling Liu, my peers working on similar projects. Finally, I have a great appreciation for the support my family and friends provided throughout this period.

# Table of Contents

# Chapter 1

# Introduction

Anomaly detection and fault classification is the problem in which faults, anomalies, or outliers should be detected as instances which deviate from correct instances or normal behaviour. Instances that are, ‘faulty’ or ‘out of the ordinary’ should be indicated as such. Anomaly detection and fault classification is an important task, especially in industrial settings where the testing of manufactured objects is a crucial aspect of the industrial process to ensure that products are sound and will not fail. There are many different approaches to performing anomaly detection, with many techniques requiring engineers to do it manually. Automation of anomaly detection can have distinct advantages such as freeing the time of engineers and being more accurate than human predictions. More standard techniques for performing automatic anomaly detection involve statistical or other mathematical analysis of sensor readings of the manufactured object [1]. Data-driven Machine learning (ML) approaches have also been used to perform anomaly detection, with Deep Learning providing multiple techniques [2].

In an industry setting where parts or objects are manufactured, inspections are usually implemented during or after manufacture and are generally done manually from photos or video. An engineer must check to see that the object is not damaged or contains defects. This process is time-consuming and prone to human error. An automatic process could improve speed and accuracy, overall increasing the quality and efficiency of the manufacturing process. For these reasons, it is desirable to create an automated system that can decide if a manufactured object is faulty or not, given a query image of the completed object. Commonly these objects have virtual 3-D object representations such as a Computer-Aided Design (CAD) plan of the object to be produced. This detection process can be supported by images of the correct manufactured object, rendered from a 3D model. The query image can then be compared to the correct virtual

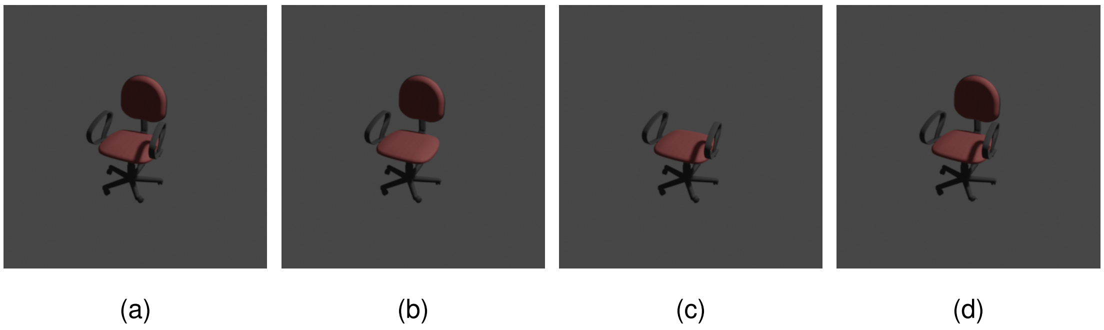


Figure 1.1: Correct and Faulty examples. Faults are induced by removing parts from models. Example (a) has no missing parts, while (b), (c) and (d) are missing a chair arm, chair back and a wheel respectively. Some missing parts are easy to detect (the chair back) compared to other parts that are barely noticeably missing (the wheel).

object (3-D model).

Computer vision (CV) has emerged as a powerful tool for detecting faults and anomalies in various industrial and manufacturing processes. CV leverages advanced image processing and machine learning techniques to analyze visual data from cameras and sensors, enabling the automated detection of defects and anomalies. By capturing and processing images or video streams, computer vision algorithms can identify patterns, textures, and structures associated with normal and faulty conditions. These algorithms can be trained to recognize specific defects or learn normal behaviour, making them adaptable to various industries and applications. Convolutional Neural Networks (CNN) [3, 4, 5] are used as feature extractors for the images, after which various Deep Learning architectures are used to combine, compare and contrast the features to perform fault classification.

The first task of this project was to create a dataset, as no multi-view missing part dataset of 3D models existed. Multi-view images mean that multiple images of the same object from different locations around the object are captured. This was performed using *Blender* [6], software for editing and rendering 3D models. This involved the creation of scripts to automatically load virtual 3D models, parse the constituent objects that made up each individual part, remove each part such that the object would be missing a part generate multiple camera views, and render the model from each view. Missing part occlusion (when the part is not visible from the viewpoint) was taken into consideration, and a script to detect whether the part was visible was created. Figure 1.1 shows examples of an object with and without missing parts, and Figure 1.2 shows a multi-view representation of an object. The PartNet Mobility dataset [7, 8, 9],

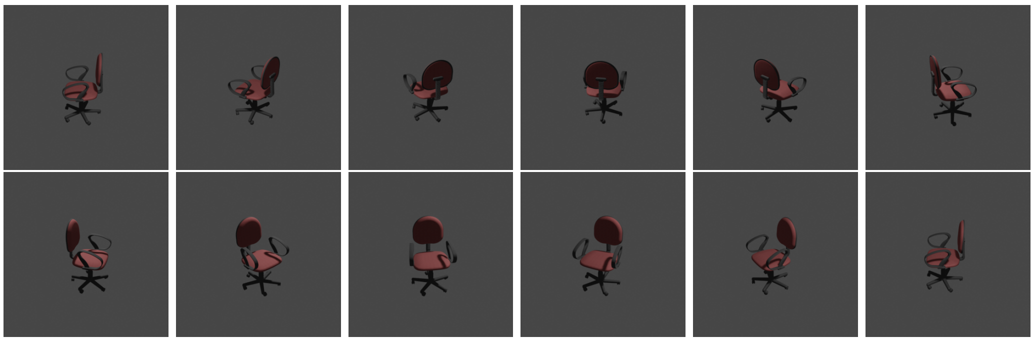


Figure 1.2: Multi-view Example of a chair

a repository for 3D models with parts, and a labelled part hierarchy was used for the source of 3D models. This repository contains over 2000 models from 46 categories. The final image dataset consisted of 152,000 images over 29 object categories.

With the multi-view image dataset, a CV system was created to classify the missing parts. For a baseline algorithm, a single-view CNN binary classifier was implemented. This took as input a single query image and would output a label indicating if the object in the image was faulty or not (missing a part or missing no parts). A Single-view Comparison Network (SVCN) was implemented to compare a query image to a single-view reference. This method was taught whether the images of the two objects were the same or not, i.e. they were classified as similar if both images contained no faults, and dissimilar if one of the images was faulty (missing a part). This was extended to the implementation of a Multi-view Comparison Network (MVCN), where 12 multi-view images of a reference object were combined and then compared to a query image. The similarity score output of this model was analogous to an anomaly score, as the reference object was a correct object missing no parts (as would be the case in the industrial scenario). A high similarity score indicated that the object in the query image was in the same state (e.g. not missing a part), while if the query image was deemed dissimilar, then it meant that the query image was missing a part (faulty). See Figure 1.3 for an example multi-view query image pair.

This task of fault classification and 3D learning has specific challenges such as anomaly detection with small faults, contrastive metric learning, and dealing with synthetic images. Moreover, difficult sub-tasks such as 3D learning and inherent pose detection must also be solved. When all these separate factors are compounded in the singular anomaly detection task, the fault classification task can be quite challenging, especially for certain categories of objects. Furthermore, because there is no standard solution that can be directly applied to this task, new techniques had to be developed.

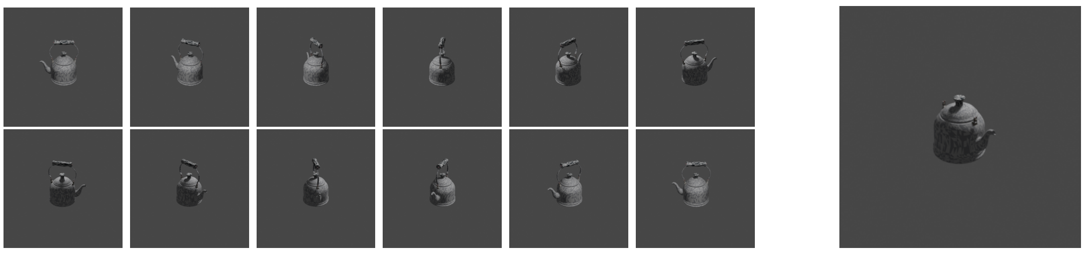


Figure 1.3: Multi-view negative pair. The reference model (a) contains 12 views of the correct instance, and the query image (b) contains a faulty view where the handle is missing.

This project implements a system that can automatically detect faults in objects and can serve as proof of concept for applications in industry. The proposed system takes as input a 3D object instance and a query image of the object taken from an unknown view. The system outputs whether or not the object in the query image contains a missing part, that is if the object is faulty. This project focuses on fault detection from images of 3D models (see Figure 1.1). In addition, the focus will be on fault classification rather than anomaly localisation. There are several ways to represent the 3D reference model such as multi-view images, Point Cloud or Voxel representation. This project focuses on the use of multi-view images only.

Experiments empirically demonstrate the value of comparing multi-view representations of 3D objects to a query image for fault classification. The proposed MVCN surpasses the SVCN and baseline in terms of performance in the standard fault classification scenario across multiple object categories. The MVCN exhibits significant enhancements over the SVCN, showing a 14.7% accuracy improvement and a 17.3% precision increase. Moreover, then MVCN achieves a 3.3% higher classification accuracy and a 0.4% higher precision compared to the baseline. The MVCN's performance is further boosted by fine-tuning and training on all categories together, resulting in a final accuracy of 81.2% and precision of 82.5%, demonstrating a 14.7% accuracy and 15.2% precision increase over the baseline model. The MVCN's efficacy extends to additional test setups. Notably, the model can detect unseen anomalies and showcases its capacity for fault detection by identifying missing parts in query images. Furthermore, the model can be adapted to learn the difference between a more general query image relative to a reference model, underscoring its adeptness for 3D learning.

# Chapter 2

# Related Work

Fault detection and missing part classification in 3D models and objects is an understudied topic. To the best of my knowledge, detecting missing parts of 3D objects using multi-view images has not been attempted before. This chapter will survey various areas related to classifying 3D faults from images and is divided into the following categories of Related Work: 2D object classification, 3D object detection and classification, Multimodal Deep Learning, fault and anomaly detection and classification techniques, and Contrastive Learning.

## 2.1 2D image object classification

Computer Vision (CV) object detection and classification are crucial to the development of a missing part classification system from images, as single-view and multi-view images must first be processed using 2D CV techniques. Object classification is the task of automatically identifying and categorizing objects within digital images or videos. The possible output classes are defined, and then for a given input image, the classification method should return the class of the most likely object within that image. This process can be generalised to multi-label classification where multiple objects within the image are classified. Object detection refers to the task of localizing objects and categorizing the localized objects within images. This Literature Review will focus on classification techniques as this project did not perform fault localization.

Improvements in Deep Learning have brought significant progress to CV. Convolutional Neural Networks (CNN) [3] are the standard deep learning method used for object classification and detection. However more recent advancements such as transformers [10] and larger pre-trained models applied to object classification and detection have contributed to improved state-of-the-art performance.

The ImageNet [11] and CIFAR [12] datasets are the main image classification benchmarks for recent computer vision techniques. ImageNet contains over 14 million hand-annotated images to include what is contained in the image, using over 20,000 different labels, while CIFAR consists of 60000 32x32 colour images in 10 classes.

CNN [3, 4] based architectures have proven to be very effective in image recognition and classification tasks, and fault detection tasks. Convolutional layers consist of multiple filters, which slide over the input image to extract relevant features. The convolutional layers are responsible for learning hierarchical representations of the input, where lower layers capture simple features like edges and corners, while deeper layers capture complex and abstract features like object parts and textures. Pooling layers are used to reduce the spatial dimensions of the feature maps while retaining the most relevant information. At the end of the CNN architecture, one or more fully connected 'dense' layers may be used to make the final predictions or classifications. CNNs can automatically learn hierarchical features from raw input data, making them effective in capturing spatial information and achieving high accuracy.

Multiple stages of development have brought deeper, larger, and more powerful CNN architectures capable of better performance. This is enabled by innovations in Deep Learning architecture and techniques, and hardware improvements. *AlexNet* [13] has an architecture of 5 convolutional layers followed by 3 fully connected layers. *VGGNet* [14] increased layer width, and had a depth increase of up to 19 layers. The *Inception* network created a new architecture that managed to increase the size of the models while leaving the computations unchanged [15]. The inception layers are repeated several times to create the 22-layer deep model GoogLeNet. Further improvements to *Inception* used connections inspired by *ResNet* [16], stacked $3x3$ filters and Batch Normalisation [17] to increase performance and accuracy [18]. *ResNet* [16] utilised skip connections to reduce the vanishing gradient problem, and multiple parameter layers to learn the representation of residuals between input and output. This architecture allowed the use of much deeper networks (up to 152 layers), although shallower versions of *ResNet* (with 18 or 50 layers) also perform well. *EfficientNet* [19] introduced an architecture that balances network depth, width, and resolution, to achieve high classification accuracies on the ImageNet dataset while using fewer parameters than previous models. See Chai et al. [5] for a more in-depth survey of modern CV techniques.

State-of-the-art object classification performance is now achieved using Transformer based architectures. Transformers are a type of deep learning architecture that use the concept of attention [10], and were originally used for natural language processing. The Vision Transformer *ViT* [20] is an adaptation of the transformer architecture to images. The Vision Transformer divides an input image into smaller fixed-size non-overlapping patches, which are then treated like words in natural language processing. Positional embeddings are introduced to encode the relative position of each image patch in the sequence, which helps the model understand the spatial layout of the patches. CoATNet [21] combines the strengths of convolution and attention, by merging CNN and Transformer architectures. More recently, *BASIC* [22] and *ViT-G* [23] have created larger and more accurate transformer-based models. However, these models are enormous, containing over 2 and 3 billion parameters for *BASIC* and *ViT-G* respectively, and they have been trained on vast amounts of data, with *BASIC* being trained on 6.7 billion image-text pairs. The *Efficient ViT* architecture [24] was used for the deepfake facial detection challenge which tried to classify images of faces as real or artificially generated. This architecture combined the use of *EfficientNet* CNN architecture and Vision Transformers.

While Transformers achieve state-of-the-art accuracy results for object classification techniques, CNN-based models achieve high accuracy and are more efficient learners with respect to the amount of data and time needed to train an accurate model.

## 2.2 3D object classification, detection, and recognition

3D object classification is a challenging task in CV that involves recognizing and categorizing three-dimensional objects. Unlike traditional 2D object classification which deals with a single image, 3D object classification operates on 3D representations of a given object. CV generally refers to vision within a 2D representation. For example, you are given a single image of a scene as input, and you want to classify the object in the image. However, this 2D image is of a real-world 3D object, which exists in a 3D space. The image of that object is just one singular view of that object. There are (potentially) an infinite number of views that could be taken of that object. You could walk around an object in a circle, and take multiple (infinite) pictures of it, the object is now represented as a multi-view image. Virtual 3D objects can have other representations such as a voxel grid, point cloud, or volumetric data. This data captures the entire 3D model in a different format than images. Voxel grid, point cloud and volumetric data can be used for 3D object detection and classification, but techniques using multi-view images are focused on.

Compared to 2D object classification, 3D object classification poses unique challenges. 3D data can be large and computationally expensive to process, requiring specialized architectures and efficient algorithms. Additionally, 3D objects can have varying shapes, orientations, and scales, making it essential for the classifier to be robust to these geometric transformations. This section provides a brief review of techniques that focus on using multiple images as the 3D object representation.

*MVCNN* [25] provides a method that combines 12 images of the 3D object, taken at a 30-degree angle, and then at 30-degree increments in a horizontal circle. A 2-stage CNN processes the images. The first CNN extracts features from the 12 input images. Then the second CNN combines all of the features, before object classification is performed using a softmax layer. *RotationNet* [26] provides 3 cases of different numbers and orientations of views. It combines views together and learns the view location independently, as well as the pose alignment of the object, alongside the class of the object. HMVCM [27] is a more complicated method that learns view-level context, by using bi-directional LSTMs in combination with CNNs for feature extraction. Qi et al. [28] provide a more detailed review of 3D object classification. These techniques have a common high-level architecture. Each uses a type of CNN object classification network *backbone*. Common options include *AlexNet* [13], *VGGNet* [14], or *ResNet* [16]. Features of the input images are extracted using the *backbone* network and are then combined in various ways depending on the technique (such as linear layers, maximum aggregation, convolutions), before being passed on to a head prediction network, which performs the classification decision based on the aggregated features.

There are other CV tasks that involve 3D objects and image representations of objects such as: 3D object detection, 3D object retrieval, and 3D object and scene reconstruction. *DETR-3D* [29] is a 3D object detection transformer based model that extends *DETR* [30], a 2D object detection transformer based model. *DETR-3D* is able to localise and identify 3D objects. 3D object retrieval is another related task. Two methods (Dual level embedding alignment *DLEA* [31] and Holistic Generative Adversarial Networks *HGAN* [32]) represent the 3D model as a set of multi-view images. These methods use 2D CNNs to extract features from multi-view images before using feature fusion modules to combine the multi-view images, before using these representations to help with 3D object retrieval. *DLEA* and *HGAN* use attention-based techniques to combine multi-view images together before they are used to aid in 3D object retrieval. *DLEA* optimizes the space between features by using cross-domain feature extraction.

*HGAN* learns the most representative single view from the multi-view, before using a GAN to generate an image which is more representative of the image of the object to retrieve. *ROCA* (Robust CAD Model Retrieval) [33] is a model that takes as input an image of a scene, detects the 3D objects from the image, retrieves and aligns 3D CAD models from a shape database, and predicts an alignment of the objects. This is a type of 3D scene construction from a 2D image. This model uses a CNN object classification network *ResNet* [16], and a CNN object detection network *Mask R-CNN* [34] to extract image features. Depth features are also calculated using a Multi-Scale Feature Fusion module. These sets of features are combined with custom modules to predict the class of each object and to estimate the pose, and overall position of each object within the scene. Finally, the scene is reconstructed with models available being placed in the estimated locations. Soltani et al. [35] introduced a method to synthesize 3D objects. They created a deep generative model based on a Variational AutoEncoder, that took as input Multi-view Depth Maps and synthesized detailed 3D point clouds as output.

## 2.3 Multimodal Machine Learning

Modality refers the the 'mode' in which something exists, or is experienced or expressed. In machine learning the modality of training refers to data format or mode, e.g. RGB images, or audio sampled at 44.1 kHz. Multimodal Machine Learning is the application of multimodal data (i.e. multiple data sources from different modes) to perform analysis on all of the data sources together. Using multiple sources and modes of data can be beneficial as you can provide more information, and information from different perspectives or orientations. This more closely resembles how humans and animals process information, we typically use information from multiple senses before making a decision or coming to a conclusion. For instance, when recognising if an animal is a dog or cat, we would look at the animal, and will also listen to see if it barks or miaows. Adapting this same example to a Computer Vision classifier, it could be of great benefit to include a sound of the object in the image (such as a bark of a dog, or a miaow of a cat), if the image is unclear or blurry the audio input could help improve the accuracy of the decision, thereby improving model performance.

In the area of Computer Vision, Multimodal learning combining images and text has been used to great effect for image generation, GANs [36] such as AttnGAN [37], the transformer-based model DALL-E [38]. For image classification, BASIC [22] has been trained on image-text pairs. Face detection has been enhanced with the use of features from both the face and iris in [39]. FusionNet [40] introduces a deep learning architecture that uses attention mechanisms to fuse information from different modalities effectively demonstrating its application to machine comprehension tasks. Unsupervised Cross-Modal Alignment of Speech and Text Embeddings [41] addresses the problem of aligning speech and text embeddings without using paired data. Bayoudh et al. [42] provide a more in-depth review.

The use of Multiple images or views, compared to a query image can be viewed as a type of multimodal learning, as the images from multiple different views can be interpreted as data of the same subject, but from different modes, i.e. different perspectives or orientations. A review of techniques focused on this 3D type of multimodal CV can be found in Section 2.2.

## 2.4 Anomaly Detection Techniques

Anomaly detection can be described as discriminating events that are 'out-of-the-ordinary' or 'anomalous'. Automatic anomaly detection is performed when a process automatically analyses events or objects and filters or picks out instances that are atypical [43]. Many technologies have been deployed to improve the accuracy and sensitivity of fault detection and diagnosis techniques. The techniques range from simple ML models relying on custom implemented features, to deep learning techniques used directly on data from machine sensors' [44, 45]. Recently, with the rise of data availability, compute power and model architectures, Deep Learning based anomaly detection has emerged as an area of research and industrial application. Deep Learning has shown impressive capability in processing high dimensional and temporal data, overtaking simpler methods. [2].

There are two distinct types of Deep Learning based anomaly detection. Unsupervised learning attempts to learn the distribution of 'normal' behaviour, such that it can identify when behaviour is 'abnormal'. Supervised anomaly detection can be used to create highly accurate models, but it requires labelled datasets. Labelled data within the area of anomaly detection for industrial processes is often scarce, therefore employing unsupervised methods in these instances is desired. Autoencoders, Variational Autoencoders, and Generative Adversarial Networks are popular techniques used to perform unsupervised anomaly detection [46, 47, 48, 36, 49].

Supervised learning uses labelled data to train a model to identify specific types of faults or anomalies. Using these methods, anomaly detection can be formulated as a binary classification or detection problem. Or in the case that there is more than 1 fault and the type of fault should be specified, fault detection can be formulated as a multi-class classification or detection problem.

## 2.5 Contrastive Learning

Contrastive Learning generally refers to Machine Learning techniques that compare and contrast two or more inputs. Loss functions such as Binary Cross-Entropy or Mean-Squared Error generally try to directly predict a label or output values. Metric losses (also called Ranking Loss) try to predict distances between examples by learning an embedding space. Instead of assigning a label to a single input, multiple inputs are given to which the model learns a similarity between the inputs. For example, in facial recognition, it is necessary to know if two pictures of a face are of the same person or not. Using Ranking loss, one can train a model that will produce a small (or zero) distance between the pictures of faces of the same person, and to produce a large distance when the faces are not of the same person. This type of learning can be applied to 3D fault detection. The query image can be compared to the representation of the 3D model, where if the image contains a non-faulty object the distance should be small, but if the query image contains a faulty object, the distance should be large.

Many types of metric losses have had applications in siamese neural networks with distance loss in Weinberger et al. [50], or Contrastive loss applied to facial recognition in FaceNet [51]. Instead of comparing pairs of inputs, triplets of inputs are compared. This type of loss function called Triplet Loss [52] commonly uses an ‘anchor’ input, and then a ‘positive’ and ‘negative’ example. The ‘positive’ example is from the same class, and the ‘negative’ example is from a different class. In the embedding space, the loss function pushes away the ‘negative’ example from the anchor and pulls the ’positive’ example towards the anchor. More recently, this type of triplet loss has been generalised to use *n* ‘positive’ and *m* ‘negative’ samples per anchor in Supervised Contrastive Learning [53]. This loss function was shown to be particularly effective at creating distinct embeddings for image classification, however, it requires a very large batch size for effective training. This type of contrastive loss has also been applied in a self-supervised manner in Chen et. al’s work [54], where ‘positive’ examples are cropped augmentations of the anchor image, and ‘negative’ examples are examples of a different class.

# Chapter 3

# Dataset Creation

Most image fault detection datasets are only single view (such as MVTec [55]), and also do not have missing parts but contain more general faults such as deformations and dents etc. There is a 3D version of this dataset [56], which contains depth scans of objects, (not multi-view images), and faults are deformations and dents etc. as in the 2D version of the dataset. Furthermore, this dataset is for unsupervised techniques only. Modelnet10 and Modelnet 40 [57] are datasets that contain 3D shapes in 10 and 40 object categories commonly used for 3D object classification, however, they have no faults or different parts.

There is no available dataset which contains objects with missing parts in a 3D space. Therefore, a dataset had to be created to train and test an automatic method to perform the missing part classification. The dataset should contain images of objects with missing parts, so tasks such as detecting or classifying the missing parts, as well as detecting if a 3D model instance is faulty or not (is missing a part or not) can be performed. The dataset should capture the objects when they have all parts (positive image) or when they have missing parts (faulty, or negative image). Multiple images of the same object from different viewpoints should be captured.

One of the major initial tasks of this project was to create this dataset[1]. Object instances were loaded and rendered automatically, and a method to automatically remove parts was also created, enabling the generation of an image dataset. This dataset is technically novel in the methods for removing parts to create faults, and methods to remove images with occluded parts. This chapter will describe and discuss: assessing and selecting the 3D models, importing and removing semantic parts of each object,

[1]The dataset used can be re-generated by following instructions and code at https://github.com/bprovanbessell/masters

rendering and representing the objects as a set of multi-view images, dealing with occluded parts, creating a train and test split, and the specific challenges of this particular dataset.

This project was used in 2 other related projects: Missing Object Part Classification with Point Clouds and Images and Optimization of 3D objects missing part detection based on Faster RCNN. Students Ruihui Yan and Wei-Ling Liu completed these projects and contributed to the development of this dataset. Wei-Ling Liu proctored the lists of possible objects and parts with assistance from Ruihui Yan and Benjamin Provan-Bessell. Benjamin Provan-Bessell created the scripts to load the 3D models and render images of them. Ruihui Yan created the scripts to parse semantic parts, and hide them for each model with assistance from Wei-Ling Liu. Benjamin Provan-Bessell created the script to create the object occlusion mask, and filter the views accordingly such that missing parts are not occluded.

## 3.1 Virtual 3D model repository

The PartNet-Mobility dataset [7] is a collection of over 2000 articulated virtual 3D objects with hierarchical part annotations. This dataset contains 46 categories of objects of everyday objects, including but not limited to: ‘box’, ‘chair’, ‘eyeglasses’, ‘faucet’, and ‘laptop’. Each object category contains between 20 and 300 separate instances. This dataset is a continuation of the PartNet [8] and ShapeNet [9] datasets. PartNet contains over 25,000 models, covering 24 object categories, with each object containing hierarchical part information, and ShapeNet is a broader 3D model dataset containing over 50,000 distinct models with category and alignment annotation. The dataset constructed in this project used objects from the PartNet-Mobility dataset.

Within each object category of the PartNet-Mobility dataset, there are multiple object instances [2] (different models) for the same type of object, for example, multiple models of a laptop. Each of these models is made up of multiple objects `(.obj files)`, which have an associated material `(.mtl file)`. All object files together create the whole model. Each instance is generally made up of 5 to 100 object files, depending on the model.

Each object instance is made up of multiple parts. For example in a pair of eyeglasses, there are 3 parts: the main body of the glasses (main frame and lenses) and the

[2]The conventional naming for virtual 3D models can be confusing: each model is made up of multiple ‘objects’. ‘Instance’ or ‘Object instance’ is used to refer to each individual model.

two legs. Each part is made up of multiple objects. This information is notated in the `result.json` file related to each instance. Different models have different numbers of parts and different part makeups. Some instances are simple and contain a small number of parts (minimum of 2), while some other instances contain more than 30 parts, and are incredibly detailed. In most models, this lends a natural hierarchy of parts to the instance. For instance, the legs of eyeglasses must be connected to the main body of the glasses. This is notated as each part having 'children' in the `result.json` file. One of the limitations of this method is that parts of instances are not always annotated in the ways one would ideally like. For example, intuitively one may think that eyeglasses could often be missing lenses or just a part of a leg. However, this is limited due to the fact that many instance models do not have the annotations of these separate parts, or that these parts are not made of separate object files. As a result, missing parts are not always as fine-grained as one may expect.

## 3.2 Object Selection and Part Annotation

One of the first tasks was to choose object instance categories that would be suitable for this missing part detection. Parts had to be able to be removed, and this removal has to be visible, or the computer vision system will not be able to detect it. Using the PartNet-Mobility hierarchical part annotation `result.json`, the 'first-level' children were used as parts to remove. These were the parts that appeared on the first level of the parts hierarchy tree. In some categories, parts are only visible from certain viewpoints, this is called part occlusion, which had to be dealt with using other techniques (see Section 3.5). Some parts are not visible at all e.g. the part annotated 'body' of the stapler, which semantically is the part that holds the staples is not visible unless the 'lid' (outside cover of the stapler) is removed. For the latter case, the parts which were never visible of the part could not be used as faults.

Each object category was manually assessed for suitability based on whether the parts were visible. In some cases, entire object categories were deemed not suitable. Similarly, each missing part for each object category was assessed, and in cases these parts were not visible, the individual part was not used.

## 3.3 Instance Importing and Rendering

Images of the object instances had to be taken to form an image dataset to train and test the Computer Vision system. *Blender* [6] is free-to-use software that can create, edit and render virtual 3-D models (among other things). This software was used to create the tools to visualise the various 3-D instances and their faults. While having a rich GUI and set of tools for manipulating and viewing 3-D objects, *Blender* also has a Python API that can be used to perform functions possible with the GUI.

A Python script (see pseudocode in Algorithm 1 below) was created using the *Blender* Python API to automatically import instances and render them from various different viewpoints. Images of each model with and without missing parts were created. When instances were rendered, the view number and the part that was removed were stored as metadata so labels could be constructed for the images later.

**Algorithm 1** 3D model instance rendering pseudocode

```
for Instance in Category do
    Load 3D model to Blender by importing each .obj file for that instance.
    Render images of model according to selected viewpoints.
    for Part in Parts to remove do
        Hide all objects corresponding to that part for rendering
        Render images of model according to selected viewpoints.
        Unhide all objects corresponding to that part for rendering
    end for
    Remove model
    Clear Scene
end for
```

## 3.4 Image representations of 3D models

This project chose to focus on image representations of 3D objects only. There are many options to consider when rendering an image of a 3D object: image resolution, object views, scene lighting, rendering engine, and object materials. This section motivates and discusses the image resolution and object views used.

### 3.4.1 Image resolution

The image resolution refers to how many pixels (width and height) the image is. This dictates the size of the image. Higher-resolution images contain more information, but are larger and take up more storage, and can be harder to process. Smaller images contain less information but can be easier to process. Images were generated at a size of $360 \times 360$ pixels. Images could be downscaled if necessary. The main image input size used for CNN feature extraction architectures was $224 \times 224$, as this was based on ImageNet classification architectures.

### 3.4.2 Object views

A 3D object lies in 3D space. There are an infinite number of 2D views of that object. For example a chair in an empty room. It is possible to walk around the chair in a circle, at a fixed distance (radius) away from the chair. If you keep the chair in the centre of your viewpoint, you also have a view angle of the chair. This simple fixed rotational viewpoint generation method was used as default. The default view angle, number of photos per rotation and radius were 30 degrees, 12 and 5 respectively (see Figure 3.1 case (i)). This setup was inspired by Multi-view convolutional neural networks (MVCNN) for 3d shape recognition [25]. Many other representations of the 3D object are possible. In RotationNet [26], they present 3 viewpoint cases where more pictures are taken from different camera points (see Figure 3.1). As you increase the number of viewpoints, you increase the information about the object available, however, this comes at the cost of having to process the extra information.

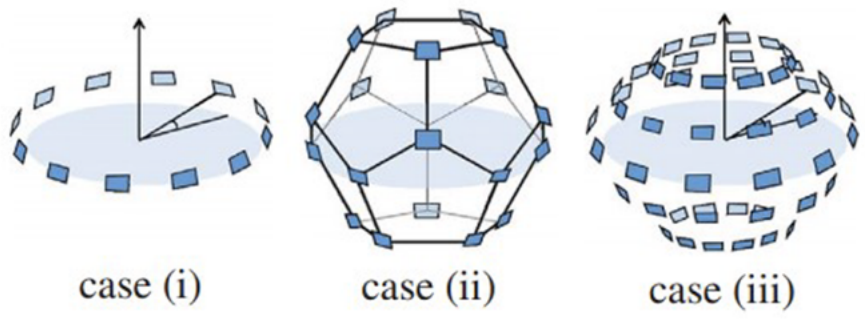


Figure 3.1: RotationNet [26] cases for viewpoints

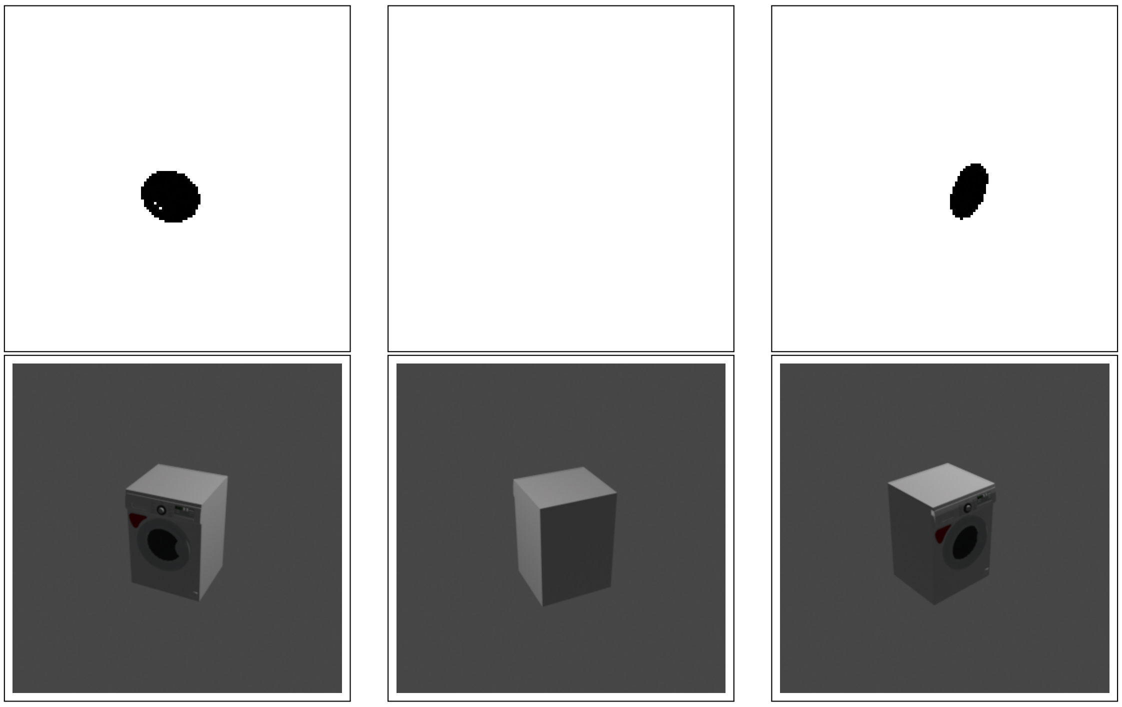


Figure 3.2: 3 views of a Washing Machine instance, the row of images represents the anomalous object mask, and the bottom row is the rendered view. As can be seen in (b) Occluded view, depending on the view, the missing part will not be visible.

## 3.5 Part Occlusion

One of the problems with multi-view images is that in some views the entire object is not visible. For certain objects, this means that parts are not always visible, this can be referred to as part occlusion (see Figure 3.2).

For instance in the case of the washing machine (see Figure 3.2), the fault is that the door is missing. However, the door is not visible in many of the view angles, as the main body covers or ‘occludes’ the door. This makes it difficult to learn, as a ‘faulty’ labelled view image will be identical to a ‘non-faulty’ labelled view image. This will make training much harder, as the model is told that the same image is both ‘faulty’ and ‘non-faulty’.

This problem does not occur in all the categories. This is a challenging issue to address as it can also happen in the real-life scenario. Given a query image from an unseen view, you cannot be sure whether there is a fault or not that is occluded. Further work investigating the effect of occluded parts is needed.

This issue of part occlusion was dealt with by removing the images that had a missing part that was occluded. Each view had to be checked if the part that should be missing was visible from that view. To perform this automatically, for each view a part mask was rendered. This part mask was created using combinators and rendering passes using *Blender*. One can set object index values to objects (remembering that each 3D model instance is made of multiple objects). This is done by setting the `pass_index` of each object that constitutes the missing part to 1, while the `pass_index` of the objects that make up the rest of the model is set to 0. When setting differing object indexes in *Blender*, you separate the rendering of each model into different layers. In this instance, you render the anomalous part in the first layer. After this anomalous part has been rendered, the rest of the model is rendered. By creating a combinator in *Blender*, you can render an alpha channel. By creating a custom combinator, you can pipe the result of the object index pass rendering to the alpha channel or a render, (N.b. you must use the '*cycles*' render engine for this processing to have any effect). When combined with the above settings of object indexes, this rendering results in an image that is white, except for the anomalous part, which shows up as black if it is visible, or the entire render will be white if it is not visible (occluded by other parts) see Figure 3.2.

## 3.6 Training Data Generation

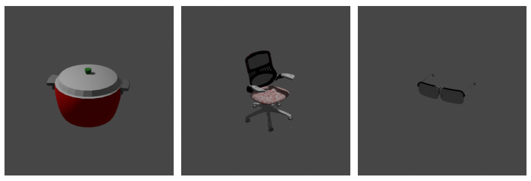
The task of fault classification can be expressed as a binary classification problem. To train a model to perform this task, the dataset should be configured to support the training and testing of this task.

The dataset is in the following format after all models with and without missing parts are rendered. For each object instance, there exists a set of multi-view images for: the object in its complete state (where it is missing no parts), and for the object with each possible missing part. Depending on the instance and missing parts available, the object can be missing 1 or more parts. Please note that faults and missing parts are analogous in this scenario. Furthermore, in the views in which the missing part would be occluded, this view is removed from the dataset. This can result in several cases:

1. There is a balance between the number of faulty images and non-faulty images.

2. There are more faulty images than non-faulty images.

3. There are more non-faulty images than faulty images.


(a) Non-Faulty images

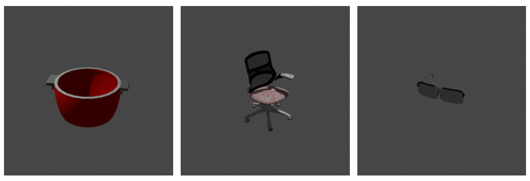

(b) Faulty images

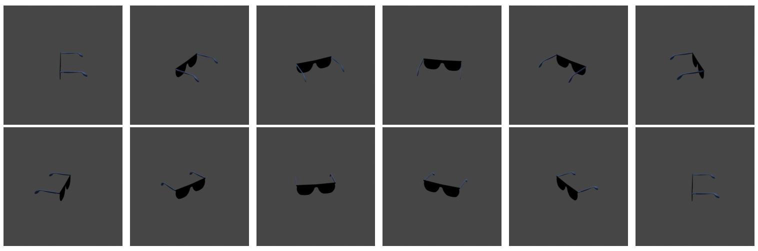
Figure 3.3: Non-Faulty and Faulty examples of the Kitchen Pot, Chair and Eyeglasses categories. For faulty examples, the Pot is missing the lid, the chair is missing the left arm, and the pair of glasses is missing the right leg.

To deal with unbalanced data in cases 2 and 3, images were sampled to ensure that there is a balanced number of faulty and non-faulty examples in the dataset. All sampling is performed with seeded random number generators to ensure consistent, reproducible samples. Other solutions such as Focal Loss [58] can be used, but this still leaves the problem of the accuracy metric being unrepresentative of true performance. For example, in a binary classification task with 80% positive examples, and 20% negative examples, the model could learn to naively classify all examples as positive, and still achieve a high accuracy of 80%, however, this is not a good indicator that the model is performing well. Another solution that could be adopted to solve this issue in the case that there is more than 1 fault is the formulation of the task as multi-class fault classification, where the type of the fault is also classified. This method however has its own limitations such as classes not existing for an unseen fault. This method was not investigated in depth in this project, and we leave it to future work.

Training datasets were set up for 3 specific models: Single-view Binary classification, Single-view similarity, and Multi-view similarity. Please see Section 4 for a detailed explanation of methods.

### 3.6.1 Single-view Binary classification dataset

For Single-view binary classification, each image was assigned a label of 0 if it was non-faulty, and a label of 1 if it was faulty. To ensure there is a balanced dataset sampling is performed: in case 2 the faulty images are randomly sampled without replacement such that the faulty samples number the same as the non-faulty samples, in case 3 the same is performed and vice versa for non-faulty and faulty images. See Figure 3.3 for non-faulty and faulty example images.


(a) Reference model views

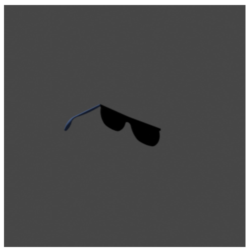

(b) Query image

Figure 3.4: Multi-view negative pair. The reference (a) contains 12 views of the correct instance, and the query image (b) contains a faulty view where a leg is missing.

### 3.6.2 Single-view and Multi-view similarity dataset

For Siamese networks, two samples are compared, and then their similarity is output. So for training and testing, models need pairs as input along with a similarity label. In the case of binary classification 'positive' and 'negative' pairs are created. A positive pair contains a non-faulty reference object representation and non-faulty query image and has a label of 0. A negative pair contains a non-faulty reference object representation and a faulty query image and has a label of 1. The similarity score output of this setup is analogous to an anomaly score, as the reference object representation is a representation of a correct object missing no parts (as would be the case in the industrial scenario). A high similarity score indicated that the object in the query image was in the same state (e.g. not missing a part), while if the query image was deemed dissimilar, then it meant that the query image was missing a part (faulty).

$$nsamples = min(\text{no. faulty samples}, \text{no. non-faulty samples}) \tag{3.1}$$

Single-view pairs are constructed as follows: For each instance, positive and negative samples are gathered. For positive pairs, *nsamples* (see equation 3.1) non-faulty images are sampled without replacement as the reference object, and each reference image is paired with a distinct non-faulty view of the same instance. For negative pairs, each reference image is paired with a faulty view of the same instance.

Multi-view pairs (Figure 3.4) are constructed as follows: A multi-view reference is created by using 12 non-faulty views of the instance, each 30 degrees apart in rotation, as described in Section 3.4.2. Positive pairs are created by pairing the reference model with *nsamples* non-faulty view with a 15-degree offset from one of the reference views. This ensures testing and training examples are distinct. For negative pairs, the reference is paired with *nsamples* faulty views of the same instance, also with a 15-degree offset from the reference views. Note that the 15-degree offset from the reference view is important, as it ensures that training and testing images remain distinct.

## 3.7 Training and Testing Data Split

A training, testing and validation split between the pairs was established using Subset Samplers and held out validation and test sets. This ensures that training, testing, and validation query images are distinct. 70% of the pairs were used for training, and 30% were held out for testing and validation, these samples were chosen randomly. To ensure a balanced number of faulty samples and non-faulty samples, negative images are sampled as described in Section 3.6.

Other testing scenarios such as unseen object, unseen anomaly, and difference learning were also experimented with. These scenarios can provide different difficulties, and some may mirror the real-life situation.

### 3.7.1 Unseen Anomaly

In this layout, anomaly types (i.e. missing parts) that were not present in the training set will be present in the test set. This will better evaluate the model's generalisation capability. This is also a more realistic test environment, as often not all anomalies will be recorded, or a part could break in a new, different way from the anomalies that are recorded. This type of training and testing is only possible for categories with more than 1 type of anomaly. For instance, Eyeglasses can be used, as they contain 2 missing legs. For this case, the training set contains only views where the left leg of an instance is missing, while the test and validation sets contain views where the right leg is missing.

### 3.7.2 Unseen Object

Training and testing object instances will be distinct. While this does not mirror the real-life test case, it further tests the generalisability of the model. This is a more challenging task as object instances can be diverse and differ considerably.

### 3.7.3 Difference Learning

As a default, the data is set up as it would be in the fault detection scenario, where the model essentially attempts to learn if the object in the query image is missing a part or complete. We could however try and teach the model only to tell if the object in the query image is in a distinct state from the 3D model representation, that is to say, learn the 'difference' between the query image and the reference. In this case, positive pairs could also be created by comparing a 3D object reference (12 views) that does have a missing part, with a query image of that model which is missing the same part. A negative pair could consist of a 3D model reference missing a part with a query image of that object in any other distinct configuration (either missing no parts or missing a different part). For instance, a positive pair (no difference) would be where the 3D reference object of a pair of glasses is missing a left leg, and the query image is the same pair of glasses also missing a left leg. A negative pair could be where the reference 3D object is a pair of glasses missing the right leg, and the query image is of the pair of glasses missing no leg. This is in contrast to the fault detection scenario, where reference 3D objects can only ever be non-faulty, i.e. include all parts.

## 3.8 Challenges

This dataset brings some specific challenges such as: anomaly detection and contrastive metric learning, with 3D learning and inherent pose detection being difficult sub-tasks. While these challenges are often dealt with independently in separate CV systems, they have not been in the same task or system in conjunction with traditional CV systems. When all these separate factors are compounded in the singular anomaly detection task, it can be quite challenging, especially for certain categories of objects. This task requires a novel methodology as there are no standard or pre-existing solutions to this problem. The system has to be able to learn how to represent a series of images (multi-views) in 2 dimensions and somehow learn appropriate features and translations that can represent this in a 3D shape. This is currently an ongoing area of research in CV.

While this type of fault detection does not explicitly deal with pose estimation of parts and objects, it still has to learn from an unknown view. The model must learn to be view-invariant, that is to say, it must learn to detect missing parts regardless of what view the object is in based on the query image. In the multi-view case, the model must

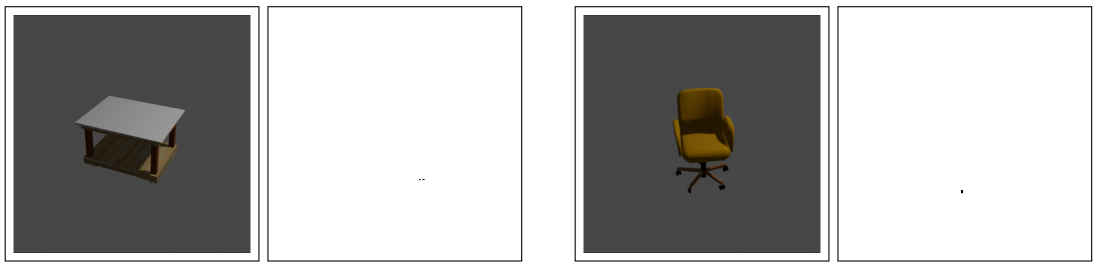

(a) Table with missing wheel (b) Chair with missing wheel

Figure 3.5: Examples of a table and chair with missing wheels. The missing wheel is barely detectable by the human eye, and when inspecting the occlusion masks (right side of each example, the small black dots represent where that wheel is), the wheel itself is very small and is difficult to detect.

be able to inherently learn some details about the position and importance of each view of the reference model, as it then has to compare these more important features to the query image.

Depending on the category, classifying and detecting faults can be extremely challenging. Often the parts that will be missing are quite small or are not very visible. Even though occluded parts have been removed from the dataset, in certain views the missing parts are quite difficult to detect (e.g. only a very small portion of the part is visible, see Figure 3.5).

Furthermore, missing parts can be very diverse. In the case of the table category, there are over 5 distinct types of missing parts, with large variability in terms of size, location, colour etc. Each of these parts does not necessarily occur in each instance, with some part types (such as 'foot' and 'runner') only appearing in 4% of instances. Such limited appearance of these parts makes it difficult to learn to distinguish and detect them.

To mitigate the effect of the lack of training data, transfer learning was used. Ideally, a larger dataset with more objects could provide a better training dataset. Models could be trained from scratch without using pre-trained ImageNet weights. This could help mitigate the domain gap of real-life images (used in training pre-trained models) and rendered images of synthetic models of this particular dataset.

This system also has to learn to compare a query image to a multi-view 3D model representation. This is different to the scenario in which standard contrastive learning is applied, where two or more samples following the same format are compared.

# Chapter 4

# Methodology

The proposed methods analyse input images (query images) which contain an object to determine if the object has a missing part or not (faulty or correct). This task can be formulated as a binary classification problem, where the probability of predicting the correct label (faulty/ non-faulty, or similar/dissimilar) is optimised. Due to the availability of the created labelled dataset, supervised machine learning techniques were utilised over unsupervised techniques. Three main methods are used: a baseline binary classification CNN model, a Single-View Comparison Network (SVCN), and a Multi-View Comparison Network (MVCN). The former takes as input a single query image, however, the latter two take as input a query image and an image reference representation of a 3D model. This process follows its expected use within industry:

For example in industry, a 3D CAD model of an object is available. An object of this type has been manufactured, and it should be tested for faults. Visual inspections of the manufactured object are conducted, and the object is compared to the 3D model. With this problem formulation, a single query (test) image should be tested, and the system should tell if the manufactured object in the image is faulty or not. In contrast to most fault classification problems, you have access to and can reference a ‘correct’ object, in the form of the 3D CAD model.

This formulation of the problem is related to similarity learning, where one learns if the query image is similar to the reference representation of the correct 3D model. In this scenario, the reference representation is a single image or a set of images of a 3D reference model. In the fault detection scenario, the reference representation is of the ‘correct’ 3D model only, emulating the real-life industrial application context.

Deep learning model architectures such as *BASIC* [22] or *ViT-G* [23] were not considered due to their scale, training times and amount of data required to train, as time and hardware constraints made the development of such models infeasible. Development of models focused on CNN-based architectures, as they are more efficient at learning with lower volumes of training data, and can take less time to train compared to transformer-based architectures.

The proposed method is similar to 3D object classification and retrieval techniques such as *MVCNN* [25], *RotationNet* [26], *DLEA* [31], and *HGAN* [32]. The proposed method, MVCN, introduces technical novelty in the application of multi-view images of 3D models to perform fault classification with contrastive learning. *MVCNN* and *RotationNet* combine multi-view images, however, they perform object classification only and have no similarity component. *DLEA*, and *HGAN* combine multi-view images which are then transformed into the query image domain. However, neither methods use similarity learning to compare the query image to the multi-view images but uses centroids of combined features, or GAN-generated features from multi-view images to perform the 3D object retrieval and classification. No methods use multi-view images for fault classification.

All methods attempt to minimise the Binary Cross-Entropy (BCE) Loss function (see equation 4.1). BCE loss quantifies the discrepancy between predicted probabilities and true labels for binary classification. It can be used to measure the similarity and dissimilarity between predictions $x$ (in the proposed methods if the query image is faulty, or if it is similar to the reference) and actual labels $y$ (the ground truth of if the query image is faulty, representing both similarity to the correct reference and the anomaly score).

$$\ell(x,y) = BCELoss = [y \cdot \log x + (1-y) \cdot \log(1-x)] \tag{4.1}$$

## 4.1 Image Feature extraction

Image feature extraction is a technique used by all proposed methods in this dissertation. This is a widely studied area of CV (see section 2.1). For image feature extraction modules, different variations of ResNet [16] were used. Versions of these networks with weights from ImageNet [11] provided good general image features. Due to the lack of high volumes of training data per object category transfer learning can be leveraged to enable effective training and fine-tuning on this smaller, more specific missing part dataset. Using pre-trained networks allowed the weights to be frozen for some experiments, vastly speeding up method training times which allow for a faster rate of development and testing of methods, while maintaining good accuracy. ResNet 50 and ResNet 18 were used due to their good performance in ImageNet classification, their availability, and ease of use for implementation. Additionally, ResNet is a popular choice for backbone feature extractor in 3D object classification [25, 26].

One drawback of this method is that these object classification models were designed for real-life images. There is a domain gap between real-life images and rendered synthetic images. Real-life images have more detailed backgrounds, often more than one object within the image, and different textures and colours compared to rendered images. Experiments where the final layers of the feature extraction models were fine-tuned (using pre-trained weights, but weights were not frozen) were conducted with the idea that this fine-tuning would mitigate the domain gap between real-life images and synthetic ones, while still making use of the advantages of transfer learning.

## 4.2 Baseline Binary CNN Classifier

The Baseline Algorithm for this task was a CNN binary classifier for a single image. This method represents a well-understood ‘standard’ approach against which other methods can be compared. If the methods using multi-view images can outperform this baseline, then the successful use of multi-view images has been shown.

ResNet50 [16] with ImageNet weights was used as a feature extraction network, followed by a classification head of a single linear layer, outputting a single neuron. Feature extractor network weights were frozen, so only the classifier head was trained. This model was trained with Binary Cross Entropy (Equation. 4.2) with Logits Loss, as such the final layer sigmoid activation could be omitted.

## 4.3 Comparison Networks

A Single-View Comparison Network and a Multi-view Comparison Network were implemented. Both methods use contrastive learning theory. In the scope of this project, it is desired to compare a real-life object to a 3D CAD model. In this scenario, we compare our test object (represented by an image i.e. the query image) to the reference (CAD) model (represented by one or more images). The Difference (or similarity) between the query image and the reference model is specified by a $[0, 1]$ metric (Equation. 4.2). 0 declares that the query image and reference have an identical missing part (or no missing part), and 1 declares that the query image and reference have nothing in common, i.e. have different missing parts. There is a distinct state for the object instance missing no parts and for each possible missing part. For instance, there is a distinct state for a chair object missing no parts, a distinct state for the chair missing each of: a left arm, a right arm, a backrest, etc. The Difference metric can be used as ground truth training labels with comparison networks. In this anomaly detection scenario, the process is driven by this defined difference metric between nominal and faulty objects, where the loss formulated using BCE (Equation. 4.1) is minimised. Note that the difference in view between the reference model and query image is not quantified. This is because methods should learn to be view-invariant as the query image will be from an unknown view in the industrial scenario. In comparison to standard similarity tasks where images must be classified as the same object as the anchor image or not, here the model must learn regardless of the view, if the query image is the same as the reference model or not. This can be particularly challenging, as views of objects can be diverse and differ substantially.

$$Difference = \begin{cases} 0, & \text{if query state} = \text{reference state} \\ 1, & \text{otherwise} \end{cases} \tag{4.2}$$

Two approaches using different representations of the reference object are introduced. The Single-view Comparison Network (SVCN) represents the CAD model with a single image of the correct object from a separate view to the query image. This architecture is extended by combining multiple views of the CAD model together. 12 images of the 3D (CAD) model from different views become the reference representation and are compared the single query image in the Multi-View Comparison Network (MVCN). Using these methods, it is possible to investigate whether or not the model can learn if a query image of an object has a fault by comparing it to a representation of the correct object, and if the use of multi-view images is beneficial to this task.

## 4.3.1 Single-view Comparison Network

A Siamese Neural Network (or Twin Network) is a network that has shared weights for two feature extraction branches. Two inputs are passed to the network, a feature vector is calculated for both of them and then the feature vectors are compared. This method is often used when performing similarity computations [59, 60]. Feature vectors are often compared by distance metrics, and the model is taught to differentiate classes by separating them in the feature embedding space, by creating a distinct space for each class. Clustering, K-Nearest Neighbours or Deep learning techniques can then be used to perform classification on features of the inputs. An alternative approach is to combine embedding extraction and classification approaches together. By passing both inputs through the same feature extraction network, the features can then be concatenated and passed through a Linear classification network that produces class outputs directly, as in FaceNet [51].

The SVCN method was created adapting by a Siamese Neural Network to the missing parts dataset. The approach of combining feature extraction was used for this method. A single correct reference image was compared to a query image. This pair was given a label of 0 if the query image was not faulty, and a label of 1 if the query image was faulty. A ResNet50 CNN was used to extract feature vectors of length 2048 from both images. These vectors were concatenated and then passed through a 2 layer Linear classification network with a single sigmoid-activated output, which outputs a probability of whether the two inputs were similar or not.

### 4.3.2 Multi-view Comparison Network

A Multi-view Comparison network was implemented to take advantage of the 3D information available from the reference object, in the form of multiple views. This method can be seen as an extension of the Siamese Comparison Network. Instead of the reference object being represented by a single image, it is represented by 12 images, which are then compared to the query image.

Model architecture is as follows: Features from the 12 reference views and the query image are extracted using the same ResNet18 CNN. The 12 reference view features of size 512 are concatenated and combined in a View comparison Network. This is a 3-layer Linear Network with sizes of $12 \times 512$, 1024, and 512 respectively. Layers are interspersed with Rectified Linear Unit (Relu) activation [61] and Batch Normalisation [17] respectively. The output vector of this network (the combined view features) is concatenated with the ResNet18 extracted features of the query image. The View Comparison Network is similar to the head network used for the Siamese Comparison model, with a two-layer network activated by Relu. The combined view features and query image features are passed through this network to produce a single output, which is then sigmoid-activated. See Figure 4.1 for an architecture diagram. ResNet18 was used as this smaller model outputs smaller output features, which reduces the size and complexity of view combination and comparison networks, reducing training times

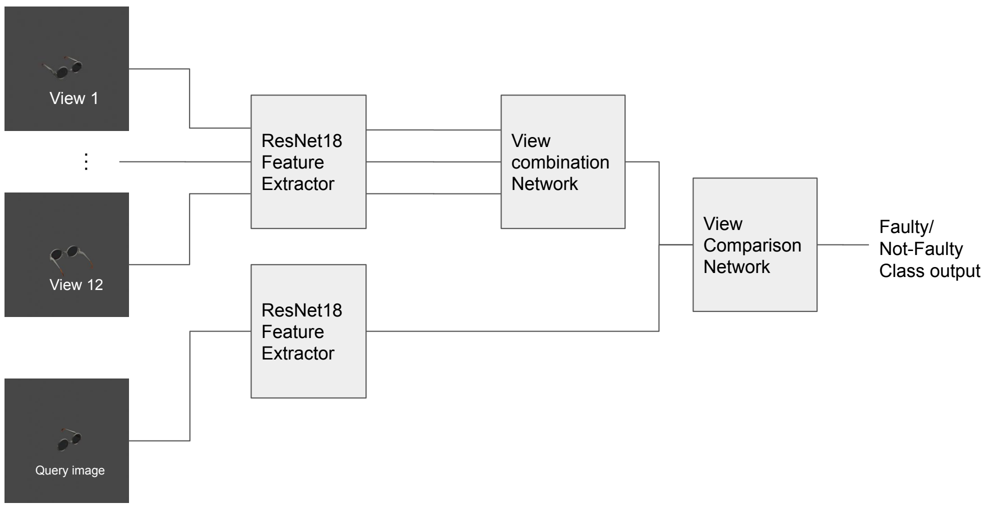

View 1
⋮
View 12
Query image
ResNet18 Feature Extractor
ResNet18 Feature Extractor
View combination Network
View Comparison Network
Faulty/ Not-Faulty Class output

Figure 4.1: Multi-View Comparison Network Architecture. 12 reference views, along with the query image share the same feature extractor. The features from the 12 reference views are combined in the View Combination Network before that combined view feature vector is compared to the extracted features of the query image in the View Comparison Network.

which allows experiments to be run on multiple categories.

There are many other architectures that could be used for performing the multi-view and query image feature combination and comparison, such as combining all ResNet18 into a comparison network and bypassing the view combination network module. Network depth and width could be experimented with as well. Attention and transformers could be used to experiment with view combination. Unfortunately due to time and computation restrictions of this project, these architectures were not possible to perform, and this investigation is left to future work.

## 4.4 Difference Learning Formulation

This project also investigates a more generalised type of 3D similarity learning, where a model is taught to recognise whether or not a 3D object is in a different state from the 3D reference object in the query image. The MVCN can be used for this task of Difference Learning without modification. By changing the layout of the data (see Section 3.7.3), the task is no longer expressed as fault detection, but as whether a query image of an object is in the same state as the reference (see Equation 4.2).

# Chapter 5

# Experiments

Methods were trained and tested over the missing parts dataset. This consisted of 29 different object categories and over 152,000 images. For brevity and ease of evaluation, 10 categories were chosen to highlight and contrast the method's best and worst performances. Tables of all results can be found in Appendix 6.2. Models were trained for each category separately by default.

Classification Accuracy and Average Precision [43] were used as metrics to evaluate model performance. To ensure classification accuracy is a meaningful metric datasets are balanced to include an even 50% split of anomalous examples and correct samples (see Section 3.6). All metrics shown are from the test set evaluation. Average Precision is used as it can be more sensitive and gives a better indication of a poorly performing model if the model is much better at classifying 1 class than the other.

Models were trained for up to 20 epochs, and the model with the best validation accuracy was saved. This model was used for testing. Large numbers of epochs were not needed for training due to the use of pre-trained image feature extractor networks, and due to the relatively small dataset size. Models were trained with the Adam [62] optimizer, and used the Binary Cross Entropy or Binary Cross Entropy with Logits Loss function. All models were implemented in PyTorch [1].

## 5.1 Binary Baseline

To ensure the soundness of the baseline, a toy dataset was used to verify correctness. This dataset consisted of (relatively) easy-to-classify images. If performance was poor, that could indicate that there was a problem with the baseline method. The baseline was

[1]Code for all experiments is available at https://github.com/bprovanbessell/masters

| Method | Baseline Metrics | | SVCN Metrics | |
|---|---|---|---|---|
| Category | Accuracy | Precision | Accuracy | Precision |
| Kitchen Pot | 95.2% | 96.8% | 70.3% | 60.7% |
| USB | 67.3% | 62.4% | 51.2% | 54.8% |
| Eyeglasses | 60.3% | 57.7% | 46.2% | 45.7% |
| Laptop | 78.8% | 77.4% | 71.8% | 69.7% |
| Globe | 97.2% | 95.0% | 90.1% | 83.0% |
| Chair | 52.7% | 54.8% | 41.5% | 42.4% |
| Table | 51.3% | 49.8% | 44.1% | 44.4% |
| Kettle | 58.5% | 63.6% | 54.0% | 49.2% |
| Storage Furniture | 64.0% | 60.8% | 55.9% | 52.5% |
| Washing Machine | 56.5% | 55.9% | 54.0% | 55.6% |
| Mean | 68.2% | 67.4% | 57.9% | 55.8% |
| Stdev | 15.9% | 15.8% | 14.3% | 11.9% |

Table 5.1: CNN Binary Classifier Baseline and Single-View Comparison Network Results per Object Category

set up to work accurately and correctly on the toy dataset achieving 95% classification accuracy on the toy dataset (*cats vs dogs, link*). The Binary Classifier CNN baseline was then trained and tested on all categories of the missing parts dataset. Classification Accuracy and Precision for each category are reported in Table 5.1.

## 5.2 Single-view Comparison Network

SVCN training was initially structured as a standard Siamese Network. This was tested with the Cats vs Dogs toy dataset, for which this model achieved 96% accuracy. In this experiment setup with the missing parts dataset, positive and negative pairs could occur between object instances. This model was unable to learn to differentiate faulty instances from non-faulty instances. The model could not do better than 50% accuracy, predicting one class regardless of the input. For this model to have any meaningful predictions, pairs had to be constructed per object instance (see Section 3.6).

The SVCN provides a useful baseline for the fault detection similarity task. While we can express this similarity task as the same fault detection task used for the binary classifier, the model is learning how to differentiate a query image from an object

| | Metrics | | Metrics (finetuned) | | Metrics (all) | |
|---|---|---|---|---|---|---|
| Category | Accuracy | Precision | Accuracy | Precision | Accuracy | Precision |
| Kitchen Pot | 89.6% | 81.7% | 97.6% | 95.1% | 93.3% | 87.5% |
| USB | 83.3% | 76.6% | 88.6% | 81.5% | 93.7% | 91.8% |
| Eyeglasses | 70.5% | 75.4% | 88.8% | 81.8% | 91.0% | 91.7% |
| Laptop | 82.0% | 73.1% | 80.8% | 71.5% | 95.0% | 94.2% |
| Globe | 95.5% | 92.8% | 98.3% | 96.6% | 96.3% | 93.6% |
| Chair | 48.8% | 46.8% | 65.0% | 71.4% | 62.8% | 71.0% |
| Table | 49.1% | 46.2% | 62.9% | 62.6% | 63.4% | 67.0% |
| Kettle | 63.8% | 59.0% | 71.5% | 64.3% | 79.5% | 78.9% |
| Storage Furniture | 67.9% | 68.5% | 85.4% | 90.3% | 81.4% | 87.2% |
| Washing Machine | 69.1% | 60.3% | 76.5% | 66.1% | 76.0% | 87.1% |
| Mean | 72.0% | 68.0% | 81.5% | 78.1% | 83.2% | 85.0% |
| Stdev | 15.0% | 14.2% | 11.8% | 12.1% | 12.1% | 9.1% |

Table 5.2: Multi-View Comparison Network Results per Object Category. Results for frozen and finetuned models, and the single model trained on all categories

model. We can compare the results of this method to the Multi-view case, and judge how providing extra information from multiple views can improve fault classification performance. See Table 5.1 for results.

## 5.3 Multi-view Comparison Network

The Multi-view Comparison Network (MVCN) was trained and tested on the missing parts dataset. Performance from both experiments can be found in Table 5.2. Experiments with fully frozen ResNet feature extractors and fine-tuning final layers were conducted, however, better results were found when fine-tuning. This method initially suffered from the exploding gradient problem, with however this was rectified by adding Batch Normalisation [17] layers in the view combination network.

Overfitting was an issue with this dataset, with training accuracy much higher than validation accuracy in most categories e.g. with the MVCN finetuned model, the Chair category has training accuracy of up to 73%, while validation accuracy has 58%, and the Kettle category has training accuracy of 90%, while validation accuracy is 70%. This indicates that the model learns characteristics of the training set, harming generalisability. Techniques such as Batch Normalisation and early stopping were used to

| | All Mean Metrics | | Picked Mean Metrics | |
|---|---|---|---|---|
| Method | Accuracy | Precision | Accuracy | Precision |
| Baseline | 66.5% | 67.2% | 68.2% | 67.4% |
| SVCN | 55.1% | 50.3% | 57.9% | 55.8% |
| MVCN | 69.8% | 67.6% | 72.0% | 68.0 |
| MVCN (finetuned) | **81.3%** | 75.6% | 81.5% | 78.1% |
| MVCN (finetuned, all) | 81.2% | **82.4%** | **83.2%** | **85.0%** |

Table 5.3: Overall average results. The best results appear in bold.

combat overfitting, however, this problem is also likely linked to the limited dataset size and diverse instances and faults. Further data augmentation techniques such as different lighting, different materials, and using slightly different view distances to the object should be investigated further. This method was also trained on all categories in a single training set, which resulted in a training set of 26000 pairs, with 3800 and 7500 pairs for validation and test set respectively.

## 5.4 Method Comparison

Table 5.3 displays the overall average metrics between all methods. Mean accuracy and precision are shown after averaging over all 29 categories, and the 10 handpicked categories (picked). Please note the baseline was not finetuned as it represents the simplest well-understood setup, and due to time limitations. What is more the non-finetuned MVCN outperformed the baseline, so experimentation was focused on this method. A single model trained on all categories as training data is marked by (all).

Classification accuracy and precision can vary widely based on category. Similar trends can be seen with all models. Baseline and MVCN models are able to accurately detect faults for the Kitchen Pot category with accuracy above 90% and precision above 80%, and for the Globe category, with accuracies and precision above 95% and 92%. This follows logically on qualitative analysis of Kitchen Pot and Globe images, they have small differences in appearance between views as they are mostly circular objects, and the missing parts are large and easily visible by the human eye (see Figure 3.3). Other categories such as Eyeglasses and storage furniture have lower accuracy and precision. This is understandable, as these categories have more than 1 missing part. When looking at images, the arms are much thinner and are harder to detect by the human eye. In some views, the arms are almost not visible, and appearance varies between views (see Figure 3.4). For Chair and Table categories, performance is poor, or lower for all models. For the Binary Baseline model, metrics are just above 50% for these categories. As a reference, a model that constantly classifies all input as a single class will achieve an accuracy of 50%. MVCN finetuned models achieve only 65.0% and 62.9% accuracy and 71.4% and 62.6% precision for these categories respectively. On qualitative analysis, this makes logical sense. The small faults in the chair and table categories are difficult to detect by the human eye (see Figure 3.5) and confuse models during training, which is detrimental to predicting other more visible missing parts such as missing chair arms and chair backs (see Figure 1.1).

The baseline model outperforms the SVCN. Similarly to the baseline model, the SVCN is unable to improve performance over a naive model for Chair and Table categories. The MVCN outperforms the SVCN in all categories. This indicates that the additional information from multiple views is useful for the comparison/similarity task of detecting a fault in the query image.

The MVCN without fine-tuning outperforms the Baseline model on average by 3.3% accuracy, and 0.4% in terms of precision. The baseline outperforms the MVCN in the Kitchen Pot (95.2% accuracy and 96.8% precision vs 89.6% accuracy and 81.7% precision) and Globe categories (97.2% accuracy and 95.0% precision vs 95.5% accuracy and 92.8% precision). This is understandable, as this architecture is specialised and optimised for 2D object classification, and these object categories do not differ much per view. However, the MVCN is able to outperform the baseline in categories which have more types of missing parts, and in which images differ more per view. This is particularly evident in the Eyeglasses category with the MVCN bringing a 10.2% increase in accuracy (70.5% vs 60.3%) and a 17.7% increase in precision (75.4% compared to 57.7%) compared to the baseline. A similarly large (13% accuracy and 5% precision) improvement can be found in the Washing Machine category, with smaller improvements being shown in the Kettle and Storage Furniture categories. This indicates the importance of multi-view images for 3D learning and fault detection in diverse objects, and highlights that one of the strengths of the MVCN is in 3D contextual understanding and learning of view-invariance. Please refer to Tables 5.1 and 5.2 for full results.

The MVCN that is fine-tuned outperforms all other models, with an increase of 11.5% in accuracy and 8% in precision compared to the non-fine-tuned version of the MVCN, and 14.8% accuracy and 8.4% in precision compared to the baseline. The

MVCN trained with a single model on all categories' combined data showed almost equal or better average performance than the fine-tuned MVCN model trained for each category. All category training was beneficial for precision mainly, increasing it by 7% compared to the finetuned MVCN with a model for each category. This type of training also improved the overfitting issue, lowering the gap between validation and training accuracies compared to most categories. This is likely due to the larger and more diverse dataset.

## 5.5 Non-standard Test Setups

Separate Training and Testing setups as described in Chapter 3 were experimented on. These testing setups were more challenging, and display the model's strengths and weaknesses under different test conditions. Training a single model on all categories' combined data was not performed due to time limitations for these setups. This training could improve performance, and these experiments are left for future work.

### 5.5.1 Unseen Object

When trained and tested on the Unseen object, the MVCN was unable to learn, with accuracy and precision remaining stuck at 50%, indicating that the model was constantly predicting a single class only. The MVCN learnt the training set, with training accuracy and precision metrics being representative of training metrics during training in the standard fault detection scenario. This inability to generalise to unseen models displays a major overfitting problem, and limitation of the method. Future work should investigate the causes and solutions to this problem, although I expect that the model may overfit the training set by looking at specific pixels, as opposed to bigger, higher-level features. This problem may possibly be alleviated using jittering and random crop data augmentations. A further experiment where a single model is trained on all categories (as in the standard scenario) should be performed, as the larger dataset with more diverse data may reduce the overfitting problem.

### 5.5.2 Unseen anomaly

The MVCN sees a significant performance drop (mean of 57% accuracy and 58.7% precision compared to 77.8% and 76.5% precision from the standard scenario). Please see Table 5.4. However, in comparison to the Unseen Object results these results

| | Unseen Anomaly | | Standard | |
|---|---|---|---|---|
| Category | Accuracy | Precision | Accuracy | Precision |
| Cart | 54.1% | 54.3% | 70.9% | 73.8% |
| Eyeglasses | 79.2% | 79.7% | 88.8% | 81.8% |
| Faucet | 49.0% | 47.8% | 82.0% | 79.0% |
| Luggage | 50.0% | 50.0% | 72.7% | 75.8% |
| Oven | 53.3% | 60.0% | 74.4% | 75.0% |
| Scissors | 65.4% | 71.6% | 80.9% | 78.8% |
| Chair | 49.1% | 49.2% | 65.0% | 71.4% |
| Switch | 59.3% | 62.8% | 72.9% | 67.6% |
| Table | 46.9% | 45.5% | 62.9% | 62.6% |
| Toilet | 66.1% | 64.5% | 87.2% | 81.6% |
| Trashcan | 45.8% | 47.1% | 85.0% | 81.2% |
| Window | 65.3% | 72.1% | 90.6% | 89.6% |
| Mean | 57.0% | 58.7% | 77.8% | 76.5% |
| Stdev | 9.8% | 11.0% | 8.9% | 6.8% |

Table 5.4: Unseen anomaly results (left) compared to Standard Fault detection (right) using Multi-view Comparison Network.

are encouraging. This test setup continues the trend where categories with diverse and small missing parts are demanding to detect (Chair, Table). This test setup also showed the Faucet, Luggage and Trashcan categories' unseen anomalies are difficult to detect. When investigating these faults they are also small and problematic to detect by the human eye (wheel on the luggage, the opener tab (lever) on the Trashcan). This test setup shows promising performance for some unseen anomalies, such as the Eyeglasses (79.2% accuracy and 79.7% precision), Window (65.3% accuracy and 72.1% precision), and Scissors category (65.4% and 71.6% accuracy and precision). High accuracy and precision show that this model is capable of classifying unseen anomalies.

While this test setup displays some limitations with the method in difficult categories, it also shows one of its strengths in dealing with unseen anomalies. This is a promising result, as this test scenario could occur in real life. You will always test on a model that you have seen before, however, some faults may not yet be recorded. Therefore these results show the aptness of this method for fault detection in industry.

| | Metrics | |
|---|---|---|
| Category | Accuracy | Precision |
| Kitchen Pot | 100.0% | 100.0% |
| USB | 95.9% | 96.4% |
| Eyeglasses | 50.7% | 52.1% |
| Laptop | 99.7% | 99.7% |
| Globe | 100.0% | 100.0% |
| Sitting Furniture | 54.2% | 60.4% |
| Table | 49.6% | 48.0% |
| Kettle | 82.3% | 97.6% |
| Storage Furniture | 49.9% | 54.0% |
| WashingMachine | 45.7% | 45.9% |
| Mean | 72.8% | 72.6% |
| Stdev | 23.4% | 32.0% |

Table 5.5: Multi-view Comparison Network Results per Object Category with Difference Learning

### 5.5.3 Difference Learning

With this setup, the general trend seems to be that more challenging categories have lower performance, while easier categories have better performance compared to the standard fault detection scenario. This seems to be because the model is not able to learn the small and diverse differences between query images and references, while categories with simpler-to-detect faults benefit from this difference learning formulation. This is backed by a higher standard deviation from the mean. This follows logically, as there are far more pairs with the categories with more faults: Previously there were only $nsamples$ pairs per instance, now there are: no. parts $+ 1 \times nsamples$, where $nparts$ is the number of missing parts per instance. The diverse and hard-to-detect faults can confuse the MVCN and make it much more difficult to tell if the objects in the images are similar or not, such as missing wheels and legs in the Chair category. This is reflected in the poor performance. The simple missing parts such as the missing lid of the Kitchen Pot are easy to detect if they are present/ not present in the query image compared to the reference. This is reflected in the flawless performance of the model in this category. See Table 5.5 for metric results.

# Chapter 6

# Conclusions and Future Work

There are many other considerations for future work that should be considered for this project, as due to the limited time and scope of this project only certain approaches were investigated.

## 6.1 Conclusion

This project introduces a novel 3D fault classification dataset. Using 3D models and semantic part annotations, a programmatic script was created to render images of synthetic objects with various missing parts using *Blender*. This dataset covers multiple views of each object and also deals with missing part occlusion to ensure a clean training and testing dataset. Methods to perform fault classification on this novel dataset were introduced. Single-view CNN classifiers and Comparison networks provided suitable baseline methods. The Multi-view Comparison Network (MVCN), an approach to representing a 3D model as a set of multi-view images before comparing it to a query image was developed.

This project showed the successful implementation of the MVCN method. When comparing a representation of a correct 3D model to a query image, the model can accurately tell if the object in the query image is missing a part or not, for a range of different categories. Favourable results of the MVCN (81.3% accuracy, 75.6% precision on average) compared to the single-view CNN classifier baseline (66.5% accuracy and 67.2% precision on average) and Single-view Comparison Network (55.1% accuracy and 50.3% precision on average) show that additional information of a multi-view reference is beneficial to performing missing part detection. However, lower accuracy in certain categories with more difficult-to-detect parts indicates that further work and investigations into improving multi-view combination and view feature comparison should be conducted. Further experiments with training a single MVCN model on data from all categories show improvements to overfitting and precision (81.2% accuracy, 82.4% precision).

The Multi-view Comparison Network was also applied to 3 different test scenarios: unseen anomaly, unseen object, and difference learning. While these setups do not necessarily mirror the real-life fault detection scenario, they test different applications of the model and display some strengths and weaknesses. The model performs poorly in the unseen object scenario, where the inability to see the object in the training set promotes more overfitting, and shows a significant method limitation. The MVCN did have success in the unseen anomaly test scenario (represented by missing parts of instances which did not occur in the training dataset). It performed worse in unseen anomaly classification than the standard scenario, however, it was able to accurately detect unseen anomalies for some categories. This is an important result, as unseen anomalies could happen in the real-life industrial fault detection scenario, this shows the strength and capability of the proposed method for application to fault detection in industry. The successful application of the MVCN to difference learning shows the method's suitability for 3D learning. Poor performance in categories with difficult-to-detect parts also highlights the difficulties of combining 3D learning, anomaly detection, and similarity learning.

## 6.2 Future Work

This project dealt with only synthetic images. This is due to the lack of availability of an image dataset containing real images of objects with missing parts, which are linked to 3D models. However, certain steps could be taken to augment the dataset to make it larger and more realistic. For model reference images, more views could be used as well as augmentations to lighting. One other way is to use no material for the CAD model representation. For the query images, steps to make them more realistic should be taken such as: different lighting scenarios, different materials used for parts e.g. changing the materials of the models such that it mirrors real life, different rendering engines could be used such that the images look more realistic, for instance, such that they have shadows, the background for these query images should be changed from a blank grey background to something that is more realistic or detailed. This could also help address the domain gap between synthetic images and real-life images. This gap should be further investigated, as specialised techniques may be needed to address these differences. This investigation was not considered in the scope of this project.

Unsupervised learning methods, such as learning correct model views only and then trying to detect anomalies from that could be performed. This could be performed using Autoencoders or GAN models. This method may perform well, and also be more applicable to scenarios where there is a lack of supervised training labels.

With Partnet-mobility object models, there is annotated part articulation. For instance, the lid of a laptop has an axis of rotation and an angle of rotation. It would be interesting if one could create a system to detect if the articulated part is damaged e.g. rotates at a different angle or in a different axis.

More complex methods to compare and contrast view features could help reduce overfitting and improve generalisation and model predictions. One could use attention to perform the multi-view feature combination and view comparison. For instance, with multiple views, some views are more important than others, as they have a clearer perspective of a certain part, e.g. the view directly looking at the arm of a chair, compared to the view in which the arm is partly obscured by the back of the chair. Perhaps with attention, the model is able to learn that the first view is more important than the other one and give more importance to this view. Moreover, other contrastive learning techniques such as triplet loss or supervised contrastive loss could create a more distinct embedding space for each missing part, resulting in better fault detection performance. The current distance metric (see Equation 4.2) takes into account missing part differences between the reference model and the query image, but it does not take into account the difference in view between the reference model and the query image. This could be formulated as a sliding scale, where more different view angles result in a higher difference metric.

Due to the time limitations of the project, experiments run were limited. Far more experiments involving different combinations of categories (such as training on all, or training with multiple similar categories for a single model) should be investigated. As shown with the MVCN model trained on all categories, larger volumes of more diverse data are beneficial to fault detection performance. This type of unified training should be applied to unseen anomaly, unseen object and difference learning training and testing setups, and could help reduce overfitting and bring performance improvements.

# Bibliography

[1] V Chandola, A Banerjee, and V Kumar. Anomaly detection: A survey. *ACM Reference Format*, 41, 2009.

[2] Guansong Pang, Chunhua Shen, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. 2021.

[3] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[4] Yann LeCun, Lawrence D Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Urs A Muller, Eduard Sackinger, Patrice Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261(276):2, 1995.

[5] Junyi Chai, Hao Zeng, Anming Li, and Eric WT Ngai. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6:100134, 2021.

[6] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[7] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[8] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[9] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[12] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

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

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[16] 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.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[18] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[19] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[21] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34:3965–3977, 2021.

[22] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Kenji Kawaguchi, Hanxiao Liu, Adams Wei Yu, Jiahui Yu, Yi-Ting Chen, Minh-Thang Luong, Yonghui Wu, et al. Combined scaling for zero-shot transfer learning. *arXiv preprint arXiv:2111.10050*, 2021.

[23] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

[24] Davide Alessandro Coccomini, Nicola Messina, Claudio Gennaro, and Fabrizio Falchi. Combining efficientnet and vision transformers for video deepfake detection. In *International conference on image analysis and processing*, pages 219–229. Springer, 2022.

[25] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[26] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[27] An-An Liu, Heyu Zhou, Weizhi Nie, Zhenguang Liu, Wu Liu, Hongtao Xie, Zhendong Mao, Xuanya Li, and Dan Song. Hierarchical multi-view context modelling for 3d object classification and retrieval. *Information Sciences*, 547:984–995, 2021.

[28] Shaohua Qi, Xin Ning, Guowei Yang, Liping Zhang, Peng Long, Weiwei Cai, and Weijun Li. Review of multi-view 3d object recognition methods based on deep learning. *Displays*, 69:102053, 2021.

[29] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2906–2917, 2021.

[30] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.

[31] Heyu Zhou, An-An Liu, and Weizhi Nie. Dual-level embedding alignment network for 2d image-based 3d object retrieval. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1667–1675, 2019.

[32] Weizhi Nie, Weijie Wang, Anan Liu, Jie Nie, and Yuting Su. Hgan: Holistic generative adversarial networks for two-dimensional image-based three-dimensional object retrieval. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(4):1–24, 2019.

[33] Can Gümeli, Angela Dai, and Matthias Nießner. Roca: Robust cad model retrieval and alignment from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4022–4031, 2022.

[34] 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.

[35] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni, and Joshua B Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1511–1519, 2017.

[36] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

[37] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1316–1324, 2018.

[38] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[39] Basma Ammour, Larbi Boubchir, Toufik Bouden, and Messaoud Ramdani. Face–iris multimodal biometric identification system. *Electronics*, 9(1):85, 2020.

[40] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension, 2018.

[41] Yu-An Chung, Wei-Hung Weng, Schrasing Tong, and James Glass. Unsupervised cross-modal alignment of speech and text embedding spaces, 2018.

[42] Khaled Bayoudh, Raja Knani, Fayçal Hamdaoui, and Abdellatif Mtibaa. A survey on deep multimodal learning for computer vision: advances, trends, applications, and datasets. *The Visual Computer*, pages 1–32, 2021.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[44] Pooja Kamat and Rekha Sugandhi. Anomaly detection for predictive maintenance in industry 4.0-a survey. In *E3S Web of Conferences*, volume 170, page 02007. EDP Sciences, 2020.

[45] Ljiljana Stojanovic, Marko Dinic, Nenad Stojanovic, and Aleksandar Stojadinovic. Big-data-driven anomaly detection in industry (4.0): An approach and a case study. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 1647–1652, 2016.

[46] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233–243, 2 1991.

[47] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[48] Tingting Chen, Xueping Liu, Bizhong Xia, Wei Wang, and Yongzhi Lai. Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder. *IEEE Access*, 8, 2020.

[49] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10265 LNCS:146–147, 3 2017.

[50] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[51] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[52] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[53] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[54] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[55] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad–a comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9592–9600, 2019.

[56] Paul Bergmann, Xin Jin, David Sattlegger, and Carsten Steger. The mvtec 3d-ad dataset for unsupervised 3d anomaly detection and localization. *arXiv preprint arXiv:2112.09045*, 2021.

[57] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[58] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[59] Davide Chicco. *Siamese Neural Networks: An Overview*, pages 73–94. Springer US, New York, NY, 2021.

[60] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005.

[61] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[62] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

# Appendix A

# Full Experiment Results

| | Metrics | |
|---|---|---|
| Category | Accuracy | Precision |
| Kitchen Pot | 95.2% | 96.8% |
| USB | 67.3% | 62.4% |
| Cart | 50.9% | 47.9% |
| Box | 64.8% | 70.6% |
| Pliers | 86.2% | 85.0% |
| WashingMachine | 56.5% | 55.9% |
| Lighter | 63.1% | 62.9% |
| Switch | 59.8% | 56.8% |
| Laptop | 78.8% | 77.4% |
| Bucket | 63.0% | 73.7% |
| Globe | 97.2% | 95.0% |
| Trashcan | 73.1% | 73.2% |
| Luggage | 45.3% | 40.0% |
| Window | 67.5% | 64.1% |
| Faucet | 60.9% | 64.7% |
| Eyeglasses | 60.3% | 57.6% |
| Kettle | 58.5% | 63.6% |
| Toilet | 68.4% | 74.3% |
| Oven | 52.1% | 50.8% |
| Stapler | 72.1% | 70.0% |
| Phone | 64.1% | 81.2% |
| Trash Can | 84.7% | 82.9% |
| Scissors | 61.1% | 59.3% |
| Dish Washer | 59.1% | 60.5% |
| Lamp | 68.1% | 70.3% |
| Chair | 52.7% | 54.8% |
| Table | 51.3% | 49.8% |
| Storage Furniture | 64.0% | 60.8% |
| Pot | 82.8% | 86.5% |
| Mean | 66.5% | 67.2% |
| Stdev | 12.7% | 13.7% |

Table A.1: CNN Binary Classifier Baseline Results per Object Category

| | Metrics | |
|---|---|---|
| Category | Accuracy | Precision |
| Kitchen Pot | 70.2% | 60.7% |
| USB | 51.2% | 54.8% |
| Cart | 40.8% | 40.7% |
| Box | 53.6% | 49.2% |
| Pliers | 51.9% | 48.4% |
| WashingMachine | 54.0% | 55.6% |
| Lighter | 56.3% | 50.7% |
| Switch | 49.3% | 47.8% |
| Laptop | 71.8% | 69.7% |
| Bucket | 44.8% | 43.2% |
| Globe | 90.1% | 83.0% |
| Trashcan | 65.3% | 61.9% |
| Luggage | 69.0% | 0.0% |
| Window | 52.4% | 54.1% |
| Faucet | 59.8% | 57.9% |
| Eyeglasses | 46.2% | 45.8% |
| Kettle | 54.0% | 49.2% |
| Toilet | 58.8% | 57.3% |
| Oven | 36.4% | 39.1% |
| Stapler | 44.6% | 43.8% |
| Phone | 65.4% | 42.9% |
| Trash Can | 54.2% | 50.0% |
| Scissors | 52.3% | 45.7% |
| Dish Washer | 40.5% | 41.1% |
| Lamp | 54.6% | 53.3% |
| Chair | 41.5% | 42.3% |
| Table | 44.1% | 44.4% |
| Storage Furniture | 55.9% | 52.5% |
| Pot | 68.8% | 73.9% |
| Mean | 55.1% | 50.3% |
| Stdev | 11.5% | 13.8% |

Table A.2: Siamese Comparison Network Results per Object Category

| | Metrics | |
|---|---|---|
| Category | Accuracy | Precision |
| Kitchen Pot | 89.6% | 81.7% |
| USB | 83.3% | 76.6% |
| Cart | 49.5% | 48.1% |
| Box | 72.8% | 71.4% |
| Pliers | 83.6% | 79.3% |
| WashingMachine | 69.1% | 60.3% |
| Lighter | 64.6% | 62.7% |
| Switch | 56.8% | 55.8% |
| Laptop | 82.0% | 73.1% |
| Bucket | 71.7% | 64.8% |
| Globe | 95.5% | 92.8% |
| Trashcan | 60.8% | 56.9% |
| Luggage | 43.2% | 55.0% |
| Window | 67.1% | 60.9% |
| Faucet | 64.4% | 64.6% |
| Eyeglasses | 70.5% | 75.4% |
| Kettle | 63.8% | 59.0% |
| Toilet | 74.2% | 70.7% |
| Oven | 53.7% | 52.8% |
| Stapler | 72.1% | 62.2% |
| Phone | 76.9% | 72.7% |
| Trash Can | 97.2% | 94.9% |
| Scissors | 68.1% | 73.3% |
| Dish Washer | 67.8% | 82.1% |
| Lamp | 76.4% | 79.2% |
| Chair | 48.8% | 46.8% |
| Table | 49.1% | 46.2% |
| Storage Furniture | 67.9% | 68.5% |
| Pot | 83.6% | 73.6% |
| Mean | 69.8% | 67.6% |
| Stdev | 13.5% | 12.5% |

Table A.3: Multi-view Comparison Network Results per Object Category

| | Metrics | |
|---|---|---|
| Category | Accuracy | Precision |
| Kitchen Pot | 97.6% | 95.1% |
| USB | 88.6% | 81.5% |
| Cart | 71.0% | 73.8% |
| Box | 81.6% | 77.8% |
| Pliers | 89.7% | 81.5% |
| WashingMachine | 76.5% | 66.1% |
| Lighter | 79.2% | 69.3% |
| Switch | 72.9% | 67.6% |
| Laptop | 80.8% | 71.5% |
| Bucket | 86.1% | 77.5% |
| Globe | 98.3% | 96.6% |
| Trashcan | 85.0% | 81.2% |
| Luggage | 72.7% | 75.9% |
| Window | 90.6% | 89.6% |
| Faucet | 82.0% | 79.0% |
| Eyeglasses | 88.8% | 81.8% |
| Kettle | 71.5% | 64.3% |
| Toilet | 87.2% | 81.7% |
| Oven | 74.4% | 75.0% |
| Stapler | 85.6% | 76.1% |
| Phone | 92.3% | 88.9% |
| Trash Can | 84.7% | 77.1% |
| Scissors | 81.0% | 78.8% |
| Dish Washer | 52.2% | 0.0% |
| Lamp | 88.2% | 81.8% |
| Chair | 65.0% | 71.4% |
| Table | 62.9% | 62.6% |
| Storage Furniture | 85.4% | 90.3% |
| Pot | 87.1% | 77.9% |
| Mean | 81.3% | 75.6% |
| Stdev | 10.2% | 16.5% |

Table A.4: Multi-view Comparison Network Results per Object Category Fine-Tune

| | Metrics | |
|---|---|---|
| Category | Accuracy | Precision |
| Kitchen Pot | 93.3% | 87.5% |
| USB | 93.7% | 91.8% |
| Cart | 60.0% | 65.7% |
| Box | 85.0% | 76.1% |
| Pliers | 96.3% | 95.4% |
| WashingMachine | 76.0% | 87.1% |
| Lighter | 77.0% | 75.6% |
| Switch | 65.9% | 66.7% |
| Laptop | 95.0% | 94.2% |
| Bucket | 87.0% | 85.4% |
| Globe | 96.3% | 93.6% |
| Trashcan | 86.4% | 87.7% |
| Luggage | 52.3% | 50.0% |
| Window | 75.4% | 75.7% |
| Faucet | 77.6% | 77.8% |
| Eyeglasses | 91.0% | 91.7% |
| Kettle | 79.5% | 78.9% |
| Toilet | 83.6% | 94.4% |
| Oven | 59.5% | 64.0% |
| Stapler | 93.3% | 100.0% |
| Phone | 81.0% | 80.8% |
| Trash Can | 98.5% | 96.8% |
| Scissors | 82.2% | 81.2% |
| Dish Washer | 78.0% | 89.2% |
| Lamp | 89.4% | 88.2% |
| Chair | 62.8% | 71.0% |
| Table | 63.4% | 67.0% |
| Storage Furniture | 81.4% | 87.2% |
| Pot | 94.1% | 90.2% |
| Mean | 81.2% | 82.4% |
| Stdev | 12.5% | 11.7% |

Table A.5: Multi-view Comparison Network Results per Object Category Single model trained on all categories

| | Metrics | |
|---|---|---|
| Category | Accuracy | Precision |
| Kitchen Pot | 100.0% | 100.0% |
| USB | 95.9% | 96.4% |
| Cart | 51.3% | 53.1% |
| Box | 64.6% | 69.7% |
| Pliers | 98.7% | 98.3% |
| WashingMachine | 45.7% | 45.9% |
| Lighter | 50.8% | 50.0% |
| Switch | 50.2% | 51.0% |
| Laptop | 99.7% | 99.7% |
| Bucket | 51.4% | 50.8% |
| Globe | 100.0% | 100.0% |
| Trashcan | 81.2% | 79.7% |
| Luggage | 48.3% | 47.1% |
| Window | 62.2% | 63.8% |
| Faucet | 75.0% | 97.2% |
| Eyeglasses | 50.7% | 52.1% |
| Kettle | 82.3% | 97.6% |
| Toilet | 77.1% | 83.7% |
| Oven | 46.8% | 47.6% |
| Stapler | 96.4% | 97.4% |
| Phone | 61.0% | 59.6% |
| Trash Can | 100.0% | 100.0% |
| Scissors | 76.5% | 96.8% |
| Dish Washer | 50.2% | 49.9% |
| Lamp | 96.2% | 97.2% |
| Chair | 54.2% | 80.4% |
| Table | 49.6% | 0.0% |
| Storage Furniture | 49.9% | 54.0% |
| Pot | 97.8% | 99.1% |
| Mean | 71.2% | 73.0% |
| Stdev | 20.9% | 25.5% |

Table A.6: Multi-view Comparison Network Results per Object Category Difference Learning