THE UNIVERSITY
of EDINBURGH

Master's in Civil and Environmental Engineering

Detecting sockets and radiators in 360° images of indoor spaces using deep learning

MEng Dissertation

by

John Agyekum Kufuor

Abstract Scan-to-BIM systems convert image and point cloud data into accurate 3D models of buildings. Research on Scan-to-BIM has largely focused on the automated identification of structural components. However, design and maintenance projects require information on a range of other assets including mechanical, electrical and plumbing (MEP) components. This dissertation presents a deep learning solution that locates and labels MEP components in photos. The classification and location data generated by this solution could add useful context to BIM models. The system developed for this project uses a Faster Region-based Convolutional Neural Network (Faster R-CNN) to detect single sockets, double sockets and radiators in 360° images. The effects of data augmentation policies and model parameter adjustments on the accuracy of the neural network's predictions are studied. A dataset of 249 labelled images was built to train the deep learning model. The Faster R-CNN achieved a mean average precision (mAP) of 0.951 and an average recall (AR) of 0.779 when trained with benchmark parameters on the augmented dataset.

s1512601

April 2020

Supervisor: Dr Frédéric Bosché

COVID-19 Impact

The COVID-19 pandemic obstructed the planned schedule of this dissertation. January 2020 was dedicated to completing the Interim report and MEng Bridge Design project. February was spent reconstructing the 360° image dataset based on lessons learned in the previous semester. Therefore, experimentation could not begin until March. Many early experiments failed due to a parameter configuration error and once progress commenced it was cut short by the Project Status Declaration. Ideally, the rest of March would have been spent running new experiments every day and writing up the results. In this section, the impact that the early end to practical work had on this project will be discussed.

Cross-Validation and Testing: One of the effects that the practical work hard stop had on this dissertation was that cross-validation work vital to authenticating the experimental results could not be done. In this project, prior to experimentation, the labelled images were assigned to be used for training or validation at an 80%/20% split. Applying K-fold validation, multiple 80%/20% splits would have been extracted from the data and used to develop the deep learning models as opposed to just one. Then the performance metrics for each model would have been averaged over all K trials to get a more reliable measure of effectiveness. This strategy is standard practice in leading-edge object detection research (Turcsany et al., 2013)(Maji & Malik, 2009)(Gould et al., 2008).

In addition, there was not sufficient time to build a test dataset to test models after training was complete. Those results would have been used to further verify the validation results. Developing a test set would have involved capturing and labelling entirely new images not seen in the validation or training data.

Further Dataset Augmentation: Another impact of the hard stop was that some promising dataset augmentations could not be explored. One potential strategy that was being researched was the random transformation of brightness and contrast in the images. This would teach the model about important invariances in the colour space of the photos (Shorten & Khoshgoftaar, 2019). For example, a socket is still a socket regardless of whether it is a dull grey or bright white. This is useful knowledge for the neural network to learn because in practical application photos may not always be taken under the best lighting conditions.

There were also plans to experiment with AutoAugment, an adaptive augmentation technology developed by Google Brain. It automatically determines the optimal augmentation policy for a target dataset. In this context, a policy represents a set

of transformations such as rotation and cutout performed in a specific order. A Reinforcement Learning search algorithm is used to find the policy that maximises validation accuracy (Cubuk et al., 2018). Ideally, AutoAugment would have been directly applied to the MEP dataset to detect the best augmentations for this use case. If that proved too computationally expensive a policy developed for a similar dataset could have been transferred over.

Further Parametric Study: Experiments that would have offered additional insight into the optimisation of Faster R-CNN for MEP asset detection could not be carried out. There had been plans to experiment with model parameters not covered in this report. One example is the learning rate. This setting controls the magnitude with which internal variables are adjusted after each training step (Smith, 2017).

If the learning rate is too high, the model's neuron weights will oscillate violently and accuracy will plummet. However, if the rate is too slow, it will take more training time and computational resources for model performance to peak (Bengio, 2012). Usually, an initial learning rate is set that then monotonically decreases over the training period. The Faster R-CNN would have been trained with different initial learning rates to find the optimal setting for the MEP case.

In addition, a cyclical learning rate would have been tested. Smith (2017) proved that having the learning rate fluctuate between appropriate boundaries increases the classification accuracy of neural networks. This variation facilitates the rapid traversal of saddle points in the loss function topology where the learning process stagnates (Smith, 2017).

Loss weights are another model setting that would have been analysed. When training the Regional Proposal Network (RPN) of a Faster R-CNN, the final loss calculated for each detection is a function of the classification and localisation losses (Ren et al., 2015). The weightings assigned to localisation and classification influence their impact on the learning progress. Those parameters would have been modified in an experiment to observe how they affect the properties of a model.

Multiparameter experiments were also planned where multiple settings in the same neural network would be modified based on the sensitivity analysis data. The results would then reveal how the effects of these parameters compounded each other. An optimised model would have been developed, demonstrating the highest accuracy that could be achieved by adjusting these parameters and augmentations within the available computational resources.

Additional Goals: Finally, the hard stop prevented some interesting supplementary work from being done. Further customisation of the Tensorflow Object Detection API could have been done to produce a framework more suited to academic research. For example, one customisation that could have been pursued was adding the functionality to generate confusion matrices automatically. A confusion matrix is a table that visualises performance in terms of true positives, false positives, true negatives and false negatives segmented by class (Deng et al., 2016). This information would have yielded deeper insight into how different parameters and data augmentations impacted model performance.

School of Engineering – Incident Management

Project Status Declaration

This form is to be used in unforeseen circumstances necessitating the immediate cessation of practical project work during semester. It acts as a record of the current status of practical work (whether it be laboratory based, computational, or fieldwork). Due to circumstances, ***no further practical work is to be continued, regardless of the type of work or current status***. This ensures equality of opportunity for all students, regardless of the type of work being undertaken.

The form must be completed during a meeting with the student, and verified by the student, supervisor, and either the thesis examiner, or a second supervisor. **Any practical work beyond that stated in this form will not be considered in the final project assessment.**

A copy of the signed form must be included in the final project submission.

Name: John Kufuor Student number: s1512601

Work completed

All items of wholly, or partially completed work must be listed, indicating the percentage completion for each task. Reference can be made to an attached project plan if appropriate. **Please take care to provide a full detailed list of all work done.**

- Before Christmas I built a dataset of 185 labelled images. In February, I built a new and improved dataset of 249 labelled images. 100%
- I have done a literature review of research on BIM, Machine learning and Scan-to-BIM for MEP components. 100%
- I wrote a Python program in Google Colab to train models using the Tensorflow Object Detection API. 100%
- I wrote a Python program to analyse the makeup of my dataset in terms of size distribution, class distribution and other variables. 100%
- I retrained a Faster R-CNN NAS model that was previously trained on the COCO dataset with my MEP dataset both with and without augmentation, specifically horizontal flip. 100%
- I then retrained the model multiple times with different hyperparameter and training settings to create data for sensitivity analysis. Values of mAP, AR and loss from the validation process were recorded. Validation results were generated for a range of IOUs and for predefined object size ranges [small, medium, large]. The object size definitions were derived from my Python analysis of the dataset and then defined in the COCOAPI.
- Hyperparameters that have been tested include training scale, number of proposals, sliding window stride, dropout probability and gaussian patch size. Not all of these experiments have yielded conclusive results that are useful for the dissertation. 75%
- I retrained an SSD MobileNet model that was previously trained on the COCO dataset with my MEP dataset to compare the two models. 100%
- I wrote a Python program in Jupyter notebooks to tile and filter the images and labels of my dataset. 100%
- I wrote a Python program in Jupyter notebooks to plot graphs of the results of my sensitivity analysis testing. 75%

- I have written a draft of my Thesis Introduction, Literature Review, Dataset Methodology and Model Training Methodology chapters. 70%

Work not commenced

Any items of outstanding work that have not been started should be listed here.

- Experiment with a wider range of dataset augmentations.
- Implement a hard example mining algorithm to maximise the difficulty of augmented data and thereby make training more effective.
- Do more sensitivity analysis experiments on parameters like learning rate, maximum detections and loss weight.
- Repeat my sensitivity analysis experiments with different validation-training data splits to cross-validate and improve test reliability.
- Make a python program for generating confusion matrices using Tensorflow.
- My experimental results are currently based on the performance of the model on a validation dataset over the course of training. I was planning to build a new testing dataset and Python program to verify my validation results.
- Experiment with deploying my model as part of an existing Scan-to-BIM workflow.
- Writing my Data Analysis, Conclusion and Covid-19 disruption chapters.

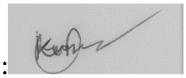
Plans for completing project submission

State revised plans for producing the final project submission in the absence of any additional practical work beyond that already listed. For example, this may include literature based research, or more in-depth analysis of results already obtained. Dates for completion of each element should be given.

- Further analysis of existing test results will be carried out that may not have been done if a greater quantity of experiments could have been conducted. [till mid-April]
- I must write my Data Analysis and Conclusion chapters as well as the new chapter on how COVID-19 disruption has affected my project. [till mid-April]
- Then I will edit and format a final draft of my dissertation. [till submission]

Declaration

To the best of our knowledge, this form is an accurate record of the project status and revised completion plans on 18/03/2020

Student: 

Supervisor: Frédéric Bosché 

Second sup./Thesis examiner: Antonis Giannopoulos  22/04/2020

Declaration

I declare that this thesis has not been submitted, in whole or in part, for any other degree or professional qualification. The work presented is entirely my own except where stated otherwise by reference or acknowledgement.

Word Count: 11,122 words

Acknowledgements

First, I would like to thank my supervisor, Dr Frédéric Bosché, for guiding me throughout this project and providing the opportunity to work on such a fascinating problem. I appreciated your encouragement and detailed feedback.

I am thankful to my parents for always supporting me. Without them, none of this would have been possible.

Finally, I am grateful to all the classmates and faculty members who made my degree such a fulfilling and transformative experience.

Contents

List of Figures

1

Introduction

1.1 Background

Building Information Modelling (BIM) is a process for creating and managing accurate virtual models of buildings. These models contain 3D geometry, cost information, asset inventories and a range of other datasets that support design, construction and facility management (Xiong et al., 2013) (Bassier et al., 2017) (Azhar, 2011).

Laser scanning and photogrammetry are surveying techniques used to acquire the 3D geometry needed for BIM models. Photogrammetry involves extracting three-dimensional measurements from two-dimensional images (El-Omari & Moselhi, 2008). Laser scanning records a collection of three-dimensional coordinates, known as a point cloud, from which the site can be reconstructed (Xiong et al., 2013).

These technologies are best suited to different use cases. Laser scanning can produce highly accurate results, but it is expensive and does not generate colour data. Photogrammetry uses equipment that is cheap and easily portable, but identifying common reference points between multiple images is a challenge and may require the use of targets (Faltýnová et al., 2016) (El-Omari & Moselhi, 2008). The potential of 360° cameras in photogrammetry is particularly interesting because they have a wider field of view than pinhole cameras. As a result, they can capture an interior space using fewer images (Barazzetti et al., 2018).

At present, the process of converting the data generated by laser scanning and photogrammetry into a BIM model is manual. The operator reconstructs the building envelope, i.e. walls, floors, and ceilings, using the point cloud as a guide. Then assets such as columns and windows are identified and recreated as BIM model

elements. These BIM elements contain material property data, energy performance data and a range of other information (Valero et al., 2016). The final product is an as-is 3D representation of the building that combines geometric information and semantic information (Laing et al., 2014). This approach is labour intensive and prone to error, so research is underway to automate it in a field of study typically referred to as Scan-to-BIM (Bassier et al., 2017).

Scan-to-BIM systems generally have three main stages as seen in Figure 1.1. First, the input data is segmented into regions of interest. These segments may be replaced by primitive objects, for example, cubes and cylinders, or boundary representations (B-Reps), that include both geometry and connectivity, to reduce computational load (Valero et al., 2016)(Adán et al., 2020). Second, using geometric and contextual information, objects of interest are detected and labelled. Object detection is a multivariate problem that involves determining both the location of an object in the scene and its classification (Huang et al., 2017). Finally, the assets are defined in BIM software (Bassier et al., 2017) (Ma & Liu, 2018). Various research projects have developed hard-coded mathematical algorithms and machine learning models to carry out these tasks (Wang & Kim, 2019).

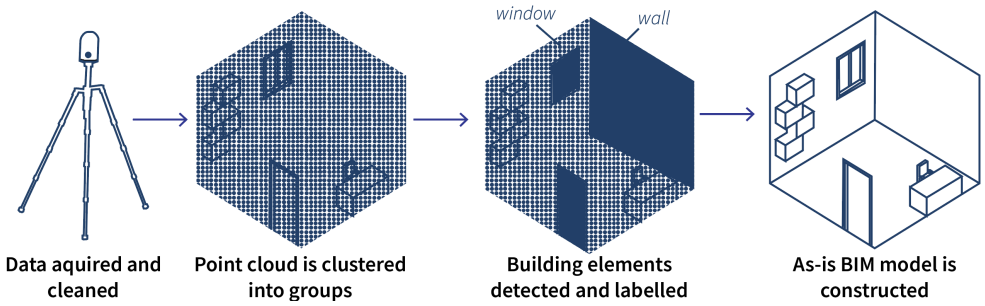


Figure 1.1: Automated Scan-to-BIM flow diagram

For Scan-to-BIM technology to be widely adopted, the computer models used to detect objects in photogrammetric and laser scan data need to be highly accurate and capable of classifying a wide range of building elements. The goal of this dissertation is to develop an object detection model that identifies sockets and radiators in 360° RGB photos of interior spaces using deep learning.

1.2 Problem analysis and motivation

In Scan-to-BIM research, work on automating object detection has mostly focused on large structural components such as floors, ceilings, and walls or openings such as doors and windows (Maalek et al., 2019)(Bassier et al., 2017)(Quintana et al.,

2018). However, the effective maintenance of buildings and other structures requires BIM models that contain many other details including mechanical, electrical and plumbing (MEP) components such as sockets and radiators. According to Adán et al. (2018), MEP assets account for a large share of building maintenance costs (Adán et al., 2018). Therefore, there is a clear need to develop Scan-to-BIM technology that focuses on MEP components.

Detecting MEP components presents a set of unique challenges. They are generally much smaller than structural components which makes it difficult for object detection models to identify them. Items that are small with regards to the image or point cloud do not offer a good depth of features for a model to interrogate (Li et al., 2017). MEP assets also have a greater range of variation within classes than structural components do; therefore, an MEP detector must learn more feature patterns. For example, different brands of radiators will have slightly different markings, valve designs and other characteristics.

Recent developments in deep learning have led to impressive results in the simultaneous detection of many classes of small objects (Li et al., 2017)(Kampffmeyer et al., 2016)(Liang et al., 2018). If successful, the application of deep learning to detecting MEP components in photographic and laser scan data will support their integration into Scan-to-BIM frameworks and ultimately deliver more detailed BIM models. These improved models will support more effective renovation and maintenance projects.

1.3 Aims and objectives

1.3.1 Aim

Deep learning models achieve state-of-the-art performance when applied to object detection problems. This project aims to determine if deep learning models can deliver high accuracy when used to detect MEP components in 360° images, specifically sockets and radiators.

1.3.2 Objectives

Objective 1: Identify and reveiw state-of-the-art research on;

- Scan-to-BIM systems used to detect MEP components
- Advancements in object detection deep learning model architectures
- Deep learning frameworks used to train and test object detection models
- Deep learning model parameters that impact small object detection performance

Objective 2: Build a dataset for training the deep learning model by;

- Collecting 360° images of sockets and radiators in a range of interior spaces
- Labelling these images in a format compatible for model training and testing

Objective 3: Develop the deep learning model by;

- Applying transfer learning to an appropriate model architecture using the 360° image dataset
- Analysing the performance of the model using data from the training and validation process
- Performing sensitivity analysis on model parameters
- Using the results of sensitivity analysis to optimise performance

1.4 Thesis Structure

The structure of this dissertation is arranged as follows. First, published research on MEP asset detection systems and deep learning is explored. Literature is sourced from the University of Edinburgh Library, Google Scholar and a range of academic archives. Then the process of building the 360 image dataset is explained and the dataset's size and class distribution are analysed. Next, the methodology used to train the deep learning model is detailed, including the technologies that were used and their limitations. Subsequently, the results of model experimentation are presented. These experiments offer insight into how the selected neural network should be optimised for the MEP use case. The deep learning model is trained on the 360 image dataset with its default settings to establish a performance benchmark. Dataset augmentations are implemented and model parameters are modified to evaluate their impact on performance. Finally, in conclusion, the practical implications of the experimental results are summarised and avenues for future research are detailed.

2

Literature Review

The chapter details a summary of the systems used to detect MEP components in images and point clouds and an overview of deep learning model architectures and frameworks that can be applied to this use case.

2.1 Detecting MEP components

2.1.1 Mathematical algorithms

The automatic identification and positioning of MEP components in interior spaces is a field that has received little research attention. Methods that have been developed usually only detect a single class of object such as pipes.

Regarding electrical components, researchers have experimented with various techniques for detecting sockets. Eruhimov et al. (2011) proposed a method that found the holes of sockets in input images using a feature detector algorithm, then applied a geometric equation to group them into coherent sets. A 3D position of the outlet was then determined using a mathematical algorithm called a planar Perspective-n-Point solver (Eruhimov & Meeussen, 2011). Meeussen et al. (2010) detected a specific variant of orange on white electrical outlets in images using a colour thresholding technique and geometric filtering (Meeussen et al., 2010). Other research identified the features of a socket in an image using Gaussian filters and contrast limited adaptive histogram equalisation (CLAHE), then applied thresholding to extract the outlet boxes (Hamledari et al., 2017).

Díaz-Vilariño et al. (2015) developed a system for detecting ceiling lights. The

ceiling of the interior space was segmented from laser scan data using a Random Sample Consensus (RANSAC) algorithm. Then the ceiling point cloud was converted into an image by the application of nearest neighbour rasterisation. A Harris corner detector function found the fluorescent lights in the image and a Hough transform algorithm identified the circular form of the standard light bulbs (Díaz-Vilariño et al., 2015).

In the field of plumbing, much of the object detection research has focused on pipes. Czerniawski et al. (2016) developed a method for identifying pipes in cluttered 3D point clouds. Assuming that the curvature of a pipe spool would differ from the surrounding clutter, they filtered the point cloud for clusters of points that had a cylindrical shape. Then, using the Bag-of-Features method, a computer vision technique where features are aggregated into a histogram, they compared each potential pipe object to what was present in a 3D CAD file of the scene. The clusters with the highest similarity were registered as being accurate representations of as-built pipe spools (Czerniawski et al., 2016). Son et al. (2015) also applied a curvature-based algorithm to point cloud data in order to detect pipelines. Their system used region growth to segment the point cloud. Then segments were classified as pipelines based on whether 30 randomly selected points fit a curvature requirement (Son et al., 2015). Alternative research detected pipes in laser scans of interior spaces by applying the Hough transform algorithm to slices of the point cloud (Bosché et al., 2015) (Ahmed et al., 2014).

Adán et al. (2018) built a system that detected multiple classes of MEP components including radiators, sockets and switches. First, the system extracted orthoimages of walls from point clouds of interiors spaces. Then these orthoimages were separated into their geometric and colour components. Colour-based detection algorithms were applied to the colour images, and geometric detection algorithms were applied to the depth images. Finally, objects were recognised and positioned based on the consensus between the two results (Adán et al., 2018).

2.1.2 Machine learning

Machine learning, where computer models learn how to perform tasks from experience, has also been employed in research on MEP detection (Michie, 1968).

Krispel et al. (2015) used a random forest classifier, which is a type of machine learning classifier, together with a sliding window, on orthophotos of walls to detect sockets and light switches (Krispel et al., 2015). Huang et al. (2013) developed

a framework for detecting a range of objects in point cloud data, including MEP assets such as valves and spotlights. Primitive shapes, such as pipes and planes, were identified using a support vector machine, which is another type of machine learning classifier. Large primitives were assumed to be background elements such as walls and discarded. Then, the remaining points were clustered using their Euclidean distance. Next, clusters that passed a linearity filter underwent a detailed matching process comparing them to components in a premade 3D object library. If the alignment between a cluster and a target component exceeded a threshold, the cluster was deemed to be a detected instance of the target (Huang & You, 2013).

Alternative data sources, such as thermal imaging have also been used. Chen et al. (2017) identified light fixtures and air-conditioners by applying a machine learning clustering model to a thermal point cloud (Chen et al., 2017).

2.2 Deep learning

Deep learning is a subset of machine learning where models, known as neural networks, that are made up of multiple computational layers learn representations of data using numerous abstraction levels. In conventional machine learning, systems are carefully designed to convert raw data into a useful set of features and then apply those features to a learning subsystem, such as a classifier. In deep learning, the model automatically discovers the features needed to make predictions (Lecun et al., 2015).

2.2.1 Object detection

Deep learning models have outperformed conventional computer vision systems when applied to industry-standard object classification and detection benchmarks such as the Pascal VOC challenge and the MS COCO challenge (Fan et al., 2016). In these challenges, models are tasked to detect a wide array of common objects such as cars, humans, chairs and clocks in 100,000s of images (Lin et al., 2014). Examples of the detection results generated for these challenges are shown in Figure 2.1. Location is presented as a bounding box and classification is presented as a label with a degree of certainty. The models built for this research project output bounding box detections similar to those featured in Figure 2.1. Based on the widely documented success of deep learning in computer vision solutions, applying deep learning technology to the MEP use case should produce significantly better

results than prior attempts.

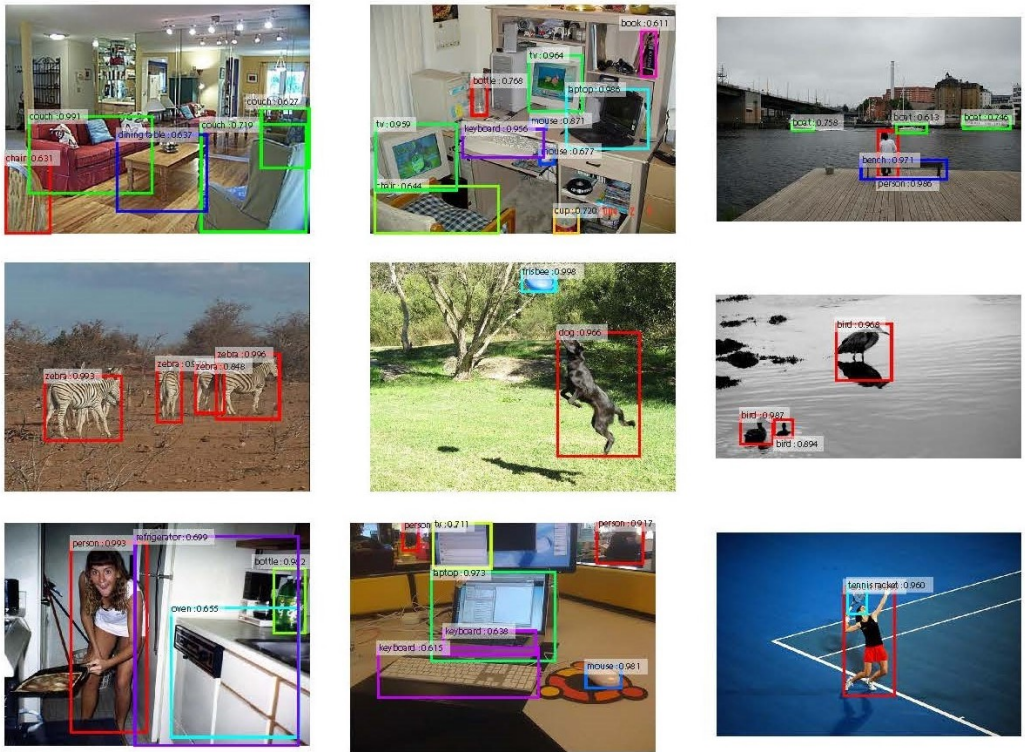


Figure 2.1: Sample object detection results from the PASCAL VOC 2007 challenge (Ren et al., 2015)

There are already many positive case studies of the application of object detection deep learning models to construction and asset management problems. Alipour et al. (2019) trained a fully convolutional neural network to detect cracks in pictures of concrete infrastructure. Their CrackPix network accurately identified over 92% of the crack pixels in the validation data (Alipour et al., 2019). Babacan et al. (2017) developed a deep learning model that semantically segmented furniture in laser scans of interior spaces (Babacan et al., 2017). Chen et al. (2019) used a neural network to detect structural elements such as beams and columns in the S3DIS point cloud data set (Chen et al., 2019). These successes offer further motivation to investigate the application of deep learning to MEP asset detection.

The type of deep learning used in this project is known as supervised learning. In supervised learning, a large set of data is collected and labelled with the outputs the model is intended to predict. In image based object detection, the intended output could be a labelled bounding box. Then, during training, the model is presented with these samples and is made to predict the appropriate output. A loss function calculates the difference between the model's prediction and the labelled reality and modifies the internal weights of the model to reduce that error (Lecun et al., 2015).

2.2.2 Faster R-CNN

The Faster Region-based Convolutional Neural Network (Faster R-CNN) object detector was chosen as the deep learning model architecture for this project. Faster R-CNN was developed by the Microsoft AI research team and has achieved state-of-the-art accuracy on industry standard benchmarks (Ren et al., 2015). As seen in Figure 2.2, the Faster R-CNN is composed of three main elements. First, a fully convolutional network extracts relevant features from the image (Fan et al., 2016). The feature map it generates is a scaled-down, convolutional representation of the original image (Ren et al., 2015). The Regional Proposal Network (RPN) then uses these features to suggest areas where the objects of interest are likely to be present. Finally, a Fast Region-based Convolutional Network (Fast R-CNN), the predecessor to the Faster R-CNN, uses a combination of classification and regression to detect objects based on the RPN proposals and feature map (Fan et al., 2016).

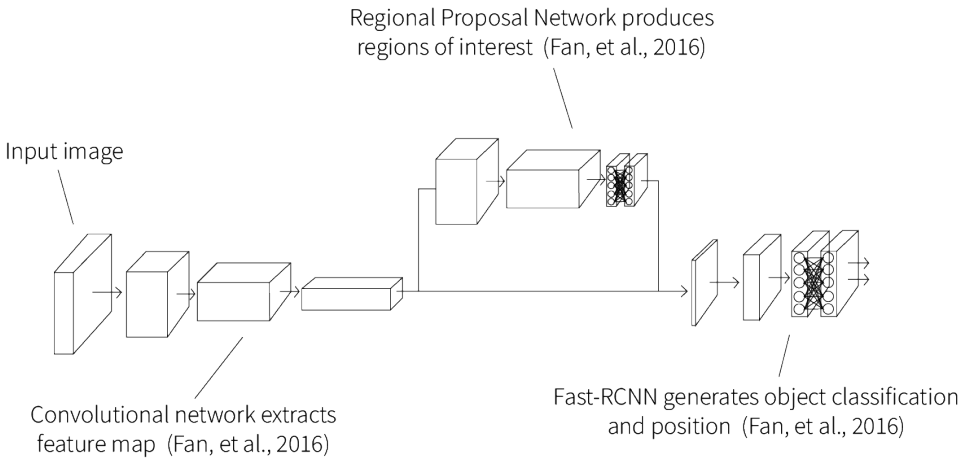


Figure 2.2: Faster R-CNN model architecture

2.2.3 Tensorflow

In this dissertation, models are developed using the Tensorflow machine learning system built by Google Brain. Tensorflow is a software library that uses dataflow graphs built from tensors (multi-dimensional arrays) to represent computations. It manages the distribution of workload across multiple computational devices, including CPUs, GPUs and TPUs. Using GPUs and TPUs during neural network development accelerates the training process because they are better suited to processing the necessary matrix calulations than CPUs (Zhang et al., 2018)(Lawrence et al., 2017). Tensorflow supports a range of deep learning experimentation with a focus on training and deploying neural networks (Abadi et al., 2016).

Tensorflow is more flexible than older, 'parameter server' designs, such as DistBelief, where the management of the training process was hard-coded into the framework. It allows users to experiment with different model architectures without having to modify the core system implementation (Abadi et al., 2016).

Research comparing Tensorflow to other modern deep learning frameworks, such as Torch, Theano and Caffe, shows that it has many advantages over its peers. Tensorflow offers extensive visualisation tools for observing graphs of the model structure, model predictions sampled from the training process and other information. Unlike most systems, Tensorflow also has in-built algorithms that automate the optimised scaling of computations from small single-machine setups to large server clusters. In addition, it implements strategies such as quantisation when running tasks so that Convolutional Neural Networks (CNNs) deliver faster prediction times when trained and deployed on Tensorflow than they would on other frameworks (Kovalev et al., 2016)(Goldsborough, 2016)(Abadi et al., 2016).

3

Dataset

3.1 Methodology

Building a large, diverse dataset is the most critical step in any successful deep learning project. By providing a large quantity and variety of examples for the model to learn from, a robust dataset ensures the model will perform well when applied to a range of real-world cases (Perez & Wang, 2017). Typically, in the field of image-based object detection, researchers use vast, pre-labelled image databases to train models, such as the Open Images detection dataset which has 9 million images and 600 classes of objects (Pont-Tuset, 2020). Currently, there are no open-source image databases focused on sockets and radiators in interior spaces. Therefore, one of the main objectives of this project was to build one. A high-level overview of the process is detailed in Figure 3.1.

A Ricoh Theta V 360° camera was used to capture RGB images. The majority of these pictures were taken in University of Edinburgh buildings because they vary in layout, contain a range of MEP assets and are freely accessible throughout the week. As shown in Figure 3.2, a variety of lighting conditions, camera angles and levels of occlusion are represented in the data. This diversity ensures that models trained on the dataset will generalise well to a range of realistic cases.

The 360° photos taken by the Theta V are large (5376 × 2688) which presented a problem. Training the deep learning model on such massive images would be very computationally expensive. Scaling down the 360° photos would make the sockets and radiators in them more difficult for a model to detect (Fan et al., 2016). It would reduce the amount of detail in the images that the model could use to make decisions.

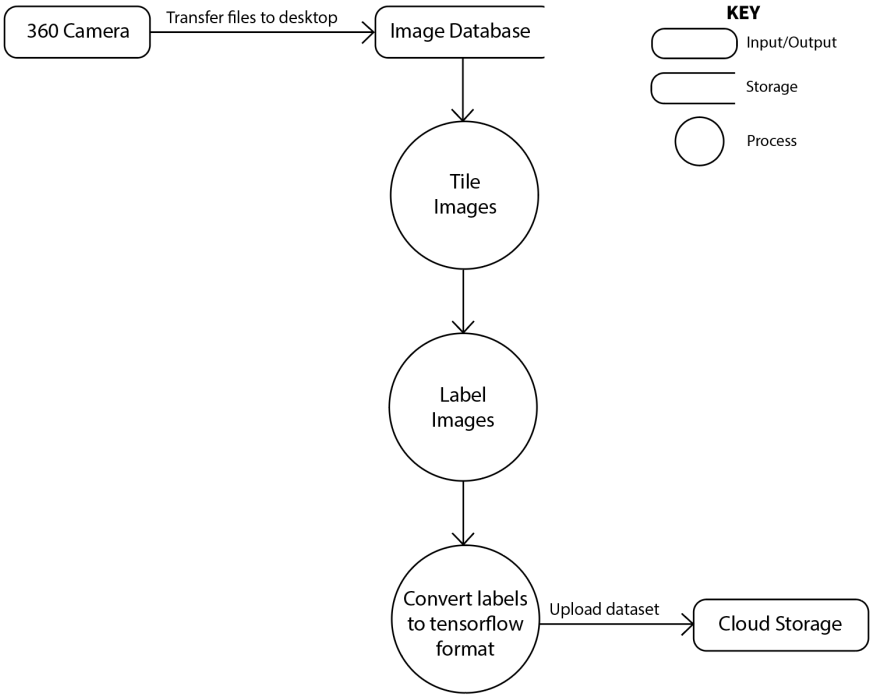


Figure 3.1: Data flow diagram of dataset building process

Figure 3.2: Diversity of dataset

To address this issue, instead of scaling down the images, they were segmented into 'tiles'. These sections were small enough to reduce the computational load to manageable levels but kept the MEP objects the same size as they were in the original image. As shown in Figure 3.3, each image was divided into six 1792 × 1344 tiles, then the tiles that contained sockets and radiators were selected for use in the dataset.

Using a Python program, 80% of images were randomly assigned to the training portion of the dataset, and the remaining 20% were assigned to validation.

Figure 3.3: Image tiling process

The sockets and radiators in the images were then labelled using the open-source tool LabelImg. Single sockets and double sockets were labelled as separate classes. LabelImg saved the classification labels and bounding box coordinates for each labelled image to an XML file as shown in Figure 3.4. Then all of these XML files were compiled into two CSV files, one for the training data and one for the validation data. Those CSV files were ported into TensorFlow compatible records files.

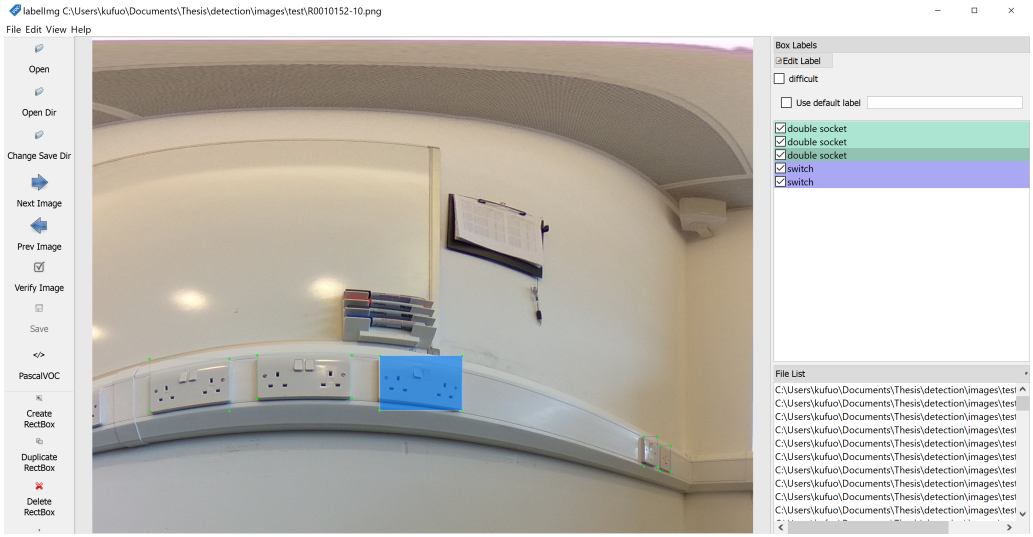


Figure 3.4: LabelImg interface

Finally, the completed dataset of images and labels was uploaded to Google Drive where it could be accessed from a remote server and used to train models.

The following Python programs were written to automate the data management process:

- *Prep.ipynb* - This Jupyter notebook randomly splits the images into a validation set and a train set (Appendix A).

- *Tile_img.py* - This script loops through the training and validation images and segments each of them into six new images (Appendix A).
- *xml_to_csv.py* - This script, which was adapted from (Tran, 2017), converts the XML label data generated by LabelImg to CSV format (Appendix A).
- *generate_tfrecords.py* - This script, which was adapted from (Tran, 2017), converts the CSV label data to TFRECORD format. This is the format accepted by Tensorflow (Appendix A).
- *Dataset_analysis.ipynb* - This Jupyter notebook analyses and generates plots on the the size distribution and category distribution of the dataset (Appendix A).

3.2 Dataset Analysis

A total of 249 images containing 438 objects were collected and labelled. These images contained 236 double sockets, 87 sockets, and 115 radiators. The percentage distribution is visualised in Figure 3.5.

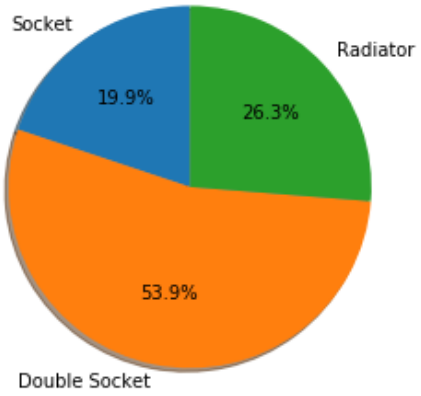


Figure 3.5: Distribution of classes

When segmented into six tiles as described in Section 3.1, this image base expanded to 1494 images, 300 of which contained objects and were selected for the dataset. The 80/20 split resulted in 240 training examples and 60 validation images.

The size distribution of the sockets and radiators in these images is plotted in Figure 3.6. The lower tertile range of labelled objects was 0 - 14892 pixels^2, the

middle tertile range 14892 - 42917 pixels^2, and the upper tertile 42917 - 804138 pixels^2. These size ranges were defined as small, medium and large and used to inform the results of validation as discussed in Section 3.

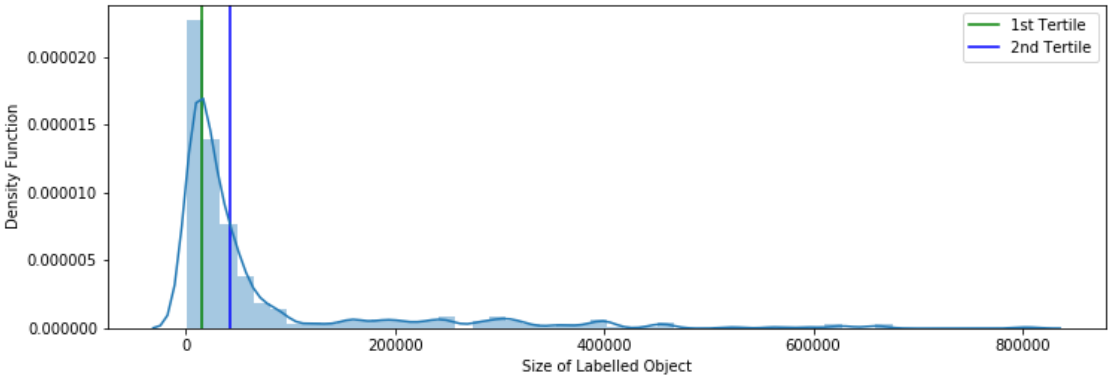


Figure 3.6: Size Distribution of Dataset

4

Model Training and Validation

4.1 Selection of Model

The deep learning model chosen for this research is one with the Faster R-CNN meta-architecture that uses Neural Architecture Search Net (NASNet) as a feature extractor and was previously trained on the Microsoft Common Objects in Context (COCO) detection dataset.

The Faster R-CNN model was obtained from the TensorFlow object detection Model Zoo, a repository of open-source models. As discussed in section 1.2, this model architecture was developed by the Microsoft AI research team and has been proven to achieve high accuracy on object detection challenges when compared to other architectures (Huang et al., 2017)(Ren et al., 2015).

The Faster R-CNN had already been trained on the COCO detection dataset which contains more than 200,000 labelled images with 80 object classes. The COCO images present objects in a variety of realistic settings - partly obscured, amongst clutter and in an appropriate context (Lin et al., 2014). Models trained on COCO generalise better to everyday scenes and performance on this dataset is used as a benchmark for object detection in many state-of-the-art research projects (Huang et al., 2017).

The object detection knowledge the Faster R-CNN gained from that COCO training allowed the model to achieve higher accuracy with less data when retrained on the MEP dataset than would otherwise have been possible. This technique is known as transfer learning. It has been investigated in much of the published literature on deep learning and proven to improve model performance in a range of

fields (Weiss et al., 2016)(Wang & Mahadevan, 2011)(Farhadi et al., 2007). Building an extensive dataset is a time-intensive process, so transfer learning, which reduces the necessary size of the dataset, was key to the success of this project. COCO was an ideal dataset to transfer-learn from for this use case because in the COCO data objects are presented in messy, realistic environments.

Every object detection architecture uses a feature extractor on the input image to extract high-level features. Selecting the right one has a significant impact on performance. The Google Brain Team developed the NASNet feature extractor in an innovative semi-automated process. A neural network searched for the optimal convolutional layer structure that, when stacked into a new neural network, would deliver the highest classification performance on the CIFAR-10 image dataset. Then by stacking more of these optimal layers together with different internal weights, they built a family of networks called NASNets that could be applied to different datasets (Zoph et al., 2018). The NASNet tailored to the ImageNet dataset delivered higher accuracy than any human-designed model. When the ImageNet NASNet was applied as a feature extractor in the Faster R-CNN it broke records on the COCO dataset (Zoph et al., 2018). Based on these results, the NASNet is likely to achieve better performance on this use case, which is limited by a small dataset with small objects, than other feature extractors.

4.2 Methodology

4.2.1 Training and Validation Process

As discussed in Section 2.2.1, during supervised learning, a deep learning model is presented with many sets of inputs and their corresponding outputs. If the model fails to produce the right output for an input, its internal variables are edited by a loss function in such a manner as to improve future accuracy (Mohri et al., 2018). As training progresses, the model is periodically tested using validation data that it cannot' learn' i.e. which is not used to update its internal variables. If the performance of the model on the validation data begins to fall, the model is overfitting, and the training process should be stopped. Overfitting occurs when the model learns to identify objects using patterns that apply to the training data but do not generalise to new examples (Srivastava et al., 2014). A high-level overview of the training and validation workflow used for this project can be seen in Figure 4.1.

Model parameters were defined in a configuration file before the training process

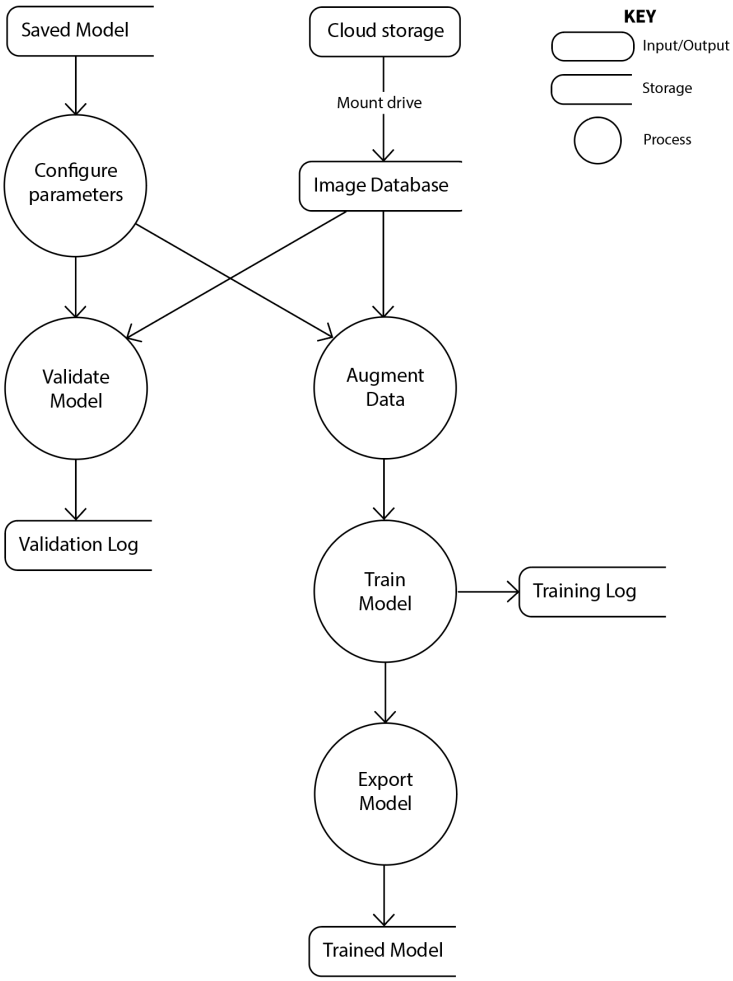


Figure 4.1: Data flow diagram of the training process

began. These included settings such as the input image size the model would accept. Then variables regulating the training and validation process were defined in the configuration file, such as the number of training steps that should occur and the location of the dataset files.

During each validation trial, the values of loss, average recall (AR) and mean average precision (mAP) were saved in log files. Loss is a function of the difference between the object classification and location predicted by the model and the ground-truth (Mohri et al., 2018). The mAP achieved during validation was used as the primary performance metric.

Once training was complete, the log data was used to analyse the model's learning progress, as shown in Section 5.

Finally, the trained model was exported in a frozen graph format that can be deployed to mobile applications, web servers, embedded systems and a range of other platforms for real-world use.

4.2.2 Calculating Performance Metrics

Measuring mAP and AR is a common strategy for evaluating the performance of object detection neural networks (Huang et al., 2017)(Fan et al., 2016)(Ren et al., 2018). During each validation period, these metrics are calculated by comparing the detections the model made on the validation images to the labelled reality. In this section, the process of calculating these metrics is explained.

First, the Intersection over Union (IoU) of the detections is determined. As seen in Figure 4.2, IoU is the ratio of the area of overlap and area of union between the region where the model has detected an object, known as a prediction bounding box, and the pre-labelled region where that object actually is, known as a ground truth box. A detection is true positive if it has an IoU with the ground-truth that is greater than some predefined threshold (Rezatofighi et al., 2019).

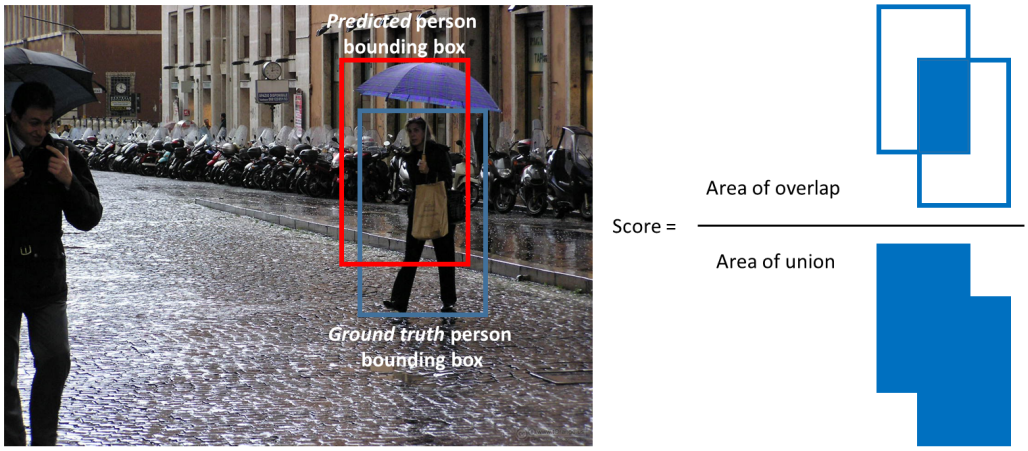


Figure 4.2: Definition of IOU (Hulstaert, 2018)

Precision is a measure of how accurate the model predictions are, as shown in Eq. 4.1. Recall is the fraction of the objects of interest that were detected, as seen in Eq. 4.2 (Henderson & Ferrari, 2017).

$$\text{Precision} = \frac{\text{No of true positive detections}}{\text{true positive detections} + \text{false positive detections}} \tag{4.1}$$

$$\text{Recall} = \frac{\text{No of true positive detections}}{\text{true positive detections} + \text{false negative detections}} \tag{4.2}$$

The average precision (AP) of an object class is the area under the precision-recall curve for that category. Using radiators as an example, the precision-recall curve is obtained by varying the probability threshold at which the neural network decides a bounding box contains a radiator (Hui, 2018) (Henderson & Ferrari, 2017). This curve has a zigzag shape, as seen in Figure 4.3. When the probability requirement is

lowered the model makes more detections. Then recall increases as more radiators are detected in regions that the model had previously determined too risky. Precision falls as the probability threshold decreases because detections that the model has low confidence in are more likely to be false positives. It then rises at points where a true positive that had not previously been detected is found.

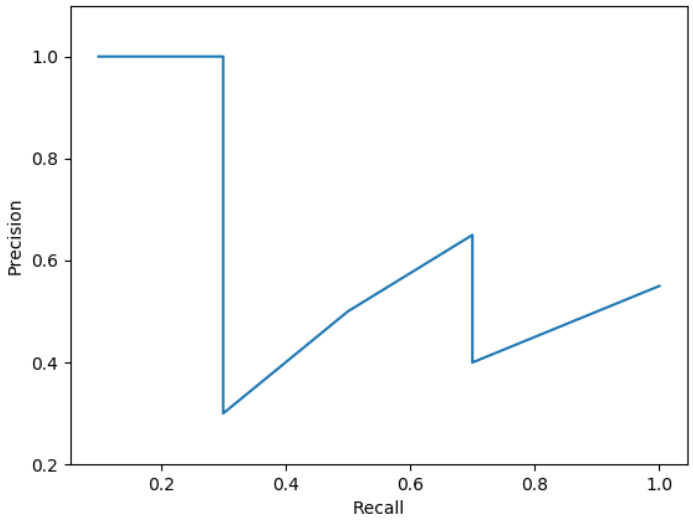


Figure 4.3: Precision-Recall Curve Example

Many object detection frameworks use an interpolated AP. First the curve is smoothed by redefining the precision function: $p_{interp}(r) = \underset{r' \geq r}{max}\, p(r')$. A precision value at recall [r] is set to the maximum precision at recall greater than [r] as seen in Figure 4.4. Then the AP is approximated by averaging the maximum precision value at several recall points as seen in Eq. 4.3 (Simonelli et al., 2019).

$$AP|_R = \frac{1}{|R|} \sum_{r \in R} p_{interp}(r) \tag{4.3}$$

For example, if an eleven-point interpolation is used, the precision values are taken at eleven equally spaced recall points on the smoothed curve as seen in Figure 4.5. Then the AP is calculated as seen in Eq. 4.4.

$$AP = \frac{1}{11} \sum_{r \in [0.0 \ldots 1.0]} p_{interp}(r) \tag{4.4}$$

The benefit of this interpolation procedure is that it smooths out random fluctuations in the precision-recall curve. The downside is that the resulting AP is less precise because it is an approximation (Li et al., 2019).

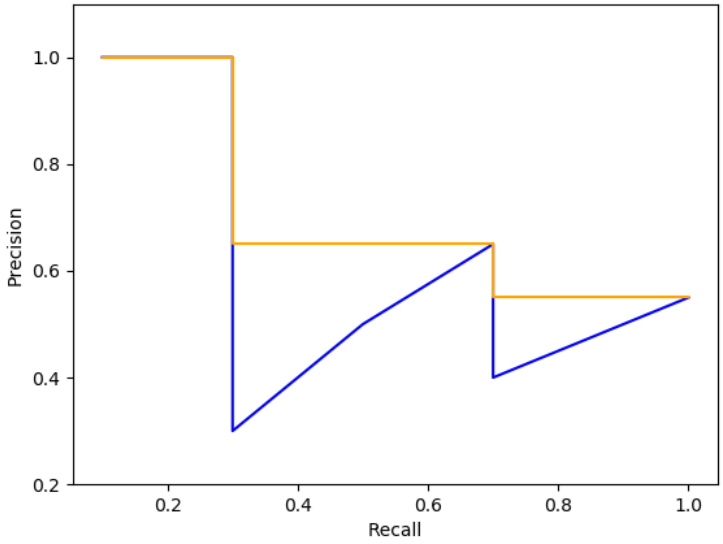


Figure 4.4: Smoothed Precision-Recall Curve Example

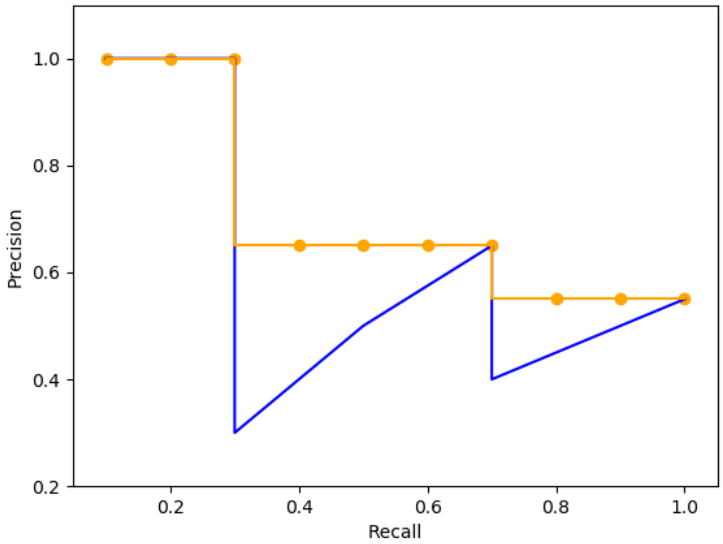


Figure 4.5: Eleven-point Interpolation Example

The mAP is calculated by finding the average of AP across all object categories. The mAP is defined according to the IoU threshold that was used. For example, mAP@0.5 is the mAP when 0.5 is the IoU threshold for a positive detection.

AR is the maximum recall given a defined number of detections per image, averaged over all the categories and a range of IoU thresholds, specifically IoU=0.5 to IoU=0.95 with a step of 0.5 (Hosang et al., 2015) (COCO Dataset, 2019). For example, AR@100 is the average recall at a maximum of 100 detections per image. All images in this dataset have more than one and less than ten sockets and radiators

in them so AR@10 will be used in the experimental analysis.

4.2.3 Dataset Augmentation

Dataset augmentation was used to improve model performance. The majority of published research on deep learning makes use of augmentation techniques. Augmentation involves increasing the size of a dataset by performing colour alterations, geometric modifications and other transformations on the images, for example, reflecting the images, cutting out patches, and changing the brightness (Perez & Wang, 2017). Neural networks have millions of internal variables that must be tuned to make accurate predictions. Therefore a proportional number of examples are needed for neural networks to achieve their full potential (Krizhevsky et al., 2012). Augmentation serves to make this scale of dataset more achievable.

The two most popular methods of delivering image transformations are known as online and offline augmentation. Offline augmentation transforms the data before training begins and stores it in memory. Online augmentation, which was implemented in this project, generates the new data during the training process. Using online augmentation significantly reduced the necessary data storage space. Unfortunately, it also had a higher computational cost and slowed down the training process (Shorten & Khoshgoftaar, 2019). The impact of dataset augmentation on performance is explored in Section 5.2.

4.2.4 Technology Stack

Cloud computing services are commonly used to host deep learning development because of the ease with which resource usage can be scaled to meet high computational and storage demand (Li et al., 2017). As seen in Figure 4.6, Google Colab, which offers cloud-based Jupyter Notebooks, was the chosen service for this project. Jupyter Notebook is an open-source tool for creating documents that contain both live code and descriptive text (Perkel, 2018). Jupyter supports a wide range of programming languages. In this project, the code for setting up the Tensorflow environment, training models using the Object Detection API, and analysing the developed models was all written in Python. Colab provides the use of a single NVIDIA Tesla K80 GPU for up to 12 hours at a time to accelerate the training process (Carneiro et al., 2018).

The Tensorflow Object Detection API is a deep learning framework built on top

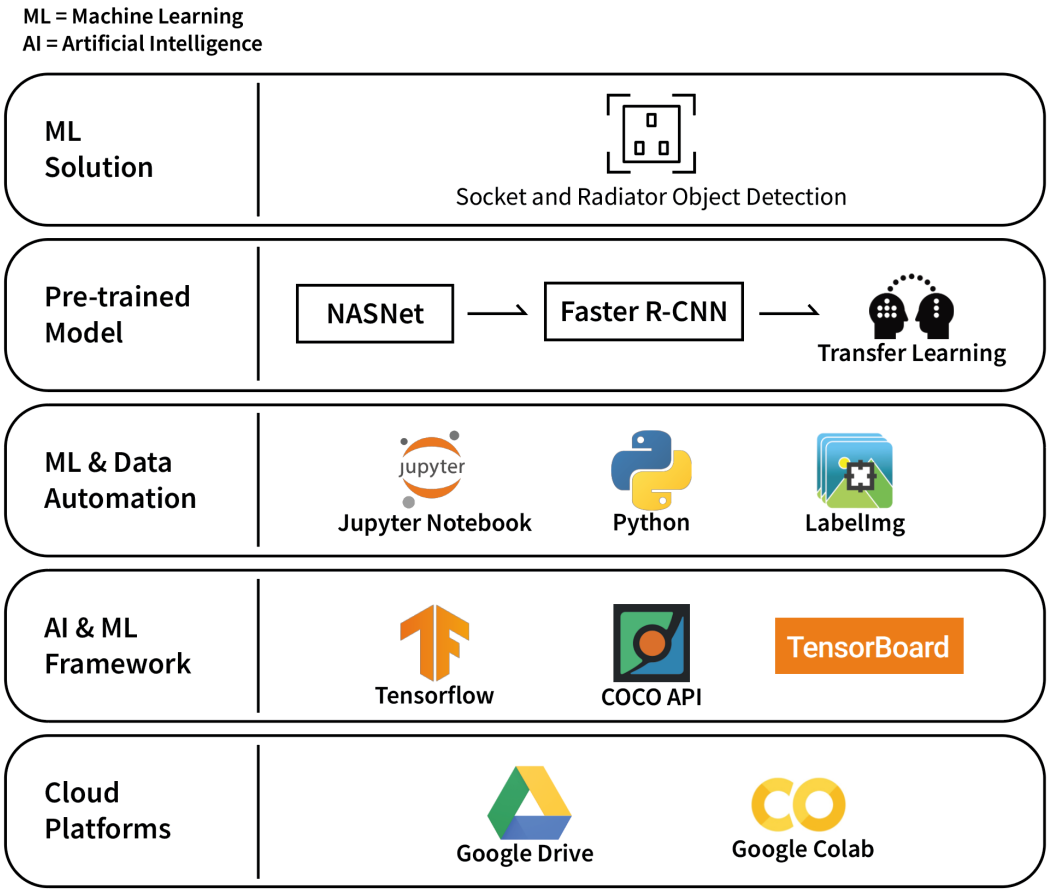


Figure 4.6: Technology Stack

of Tensorflow that was used in this project. It provides interfaces that simplify the process of training and deploying object detection models (Huang et al., 2017).

The COCO Python API is a tool designed to parse object detection test results into the COCO format when training models on the COCO dataset (COCO Dataset, 2019). It was modified and integrated with the Tensorflow Object Detection API to expand the evaluation functionality. The Object Detection API evaluates AP for each class, mAP@0.5 and loss. The COCO API extends this by calculating mAP and AR at different IOUs and size ranges as seen in Figure 4.7. The COCO default size definitions of small, medium and large were replaced with the tertile size ranges found in Section 3.2. The COCO API uses a 101-point interpolation to calculate AP (Oksuz et al., 2018).

Tensorboard is a web application built to help users inspect and understand Tensorflow runs (Tensorflow, 2015). It takes Object Detection API log files as an input. In this project, Tensorboard was used to download the tracked progress of each experiment in CSV format. It also provided visualizations of validation detections in PNG format.

```
I0324 16:41:58.304629 139669053331328 eval_util.py:80] Writing metrics to tf summary.
INFO:tensorflow:Losses/Loss/BoxClassifierLoss/classification_loss: 0.059049
I0324 16:41:58.304984 139669053331328 eval_util.py:87] Losses/Loss/BoxClassifierLoss/classification_loss: 0.059049
INFO:tensorflow:Losses/Loss/BoxClassifierLoss/localization_loss: 0.025804
I0324 16:41:58.305194 139669053331328 eval_util.py:87] Losses/Loss/BoxClassifierLoss/localization_loss: 0.025804
INFO:tensorflow:Losses/Loss/RPNLoss/localization_loss: 0.157345
I0324 16:41:58.305331 139669053331328 eval_util.py:87] Losses/Loss/RPNLoss/localization_loss: 0.157345
INFO:tensorflow:Losses/Loss/RPNLoss/objectness_loss: 0.061741
I0324 16:41:58.305536 139669053331328 eval_util.py:87] Losses/Loss/RPNLoss/objectness_loss: 0.061741
INFO:tensorflow:PascalBoxes_PerformanceByCategory/AP@0.5IOU/double socket: 0.914295
I0324 16:41:58.305681 139669053331328 eval_util.py:87] PascalBoxes_PerformanceByCategory/AP@0.5IOU/double socket: 0.914295
INFO:tensorflow:PascalBoxes_PerformanceByCategory/AP@0.5IOU/radiator: 0.840406
I0324 16:41:58.305822 139669053331328 eval_util.py:87] PascalBoxes_PerformanceByCategory/AP@0.5IOU/radiator: 0.840406
INFO:tensorflow:PascalBoxes_PerformanceByCategory/AP@0.5IOU/socket: 0.913889
I0324 16:41:58.305959 139669053331328 eval_util.py:87] PascalBoxes_PerformanceByCategory/AP@0.5IOU/socket: 0.913889
INFO:tensorflow:PascalBoxes_Precision/mAP@0.5IOU: 0.889530
I0324 16:41:58.306080 139669053331328 eval_util.py:87] PascalBoxes_Precision/mAP@0.5IOU: 0.889530
INFO:tensorflow:Metrics written to tf summary.
```

(a) Default Validation

```
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=0.45s).
Accumulating evaluation results...
DONE (t=0.14s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.617
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.897
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.763
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.416
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.631
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.716
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.549
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.709
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.719
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.600
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.744
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.789
```

(b) COCO API Validation

Figure 4.7: COCO API Validation vs Default Validation

The following Jupyter Notebooks were developed to manage the training and validation process:

- *Model_gym.ipynb* - This notebook mounts the Google Drive where the dataset is stored, installs the Tensorflow Object Detection API, integrates the COCO API, and trains and validates models (Appendix A). Adapted from (Kinsley, 2017)(Solomon, 2019)(Tensorflow, 2020).
- *Model_testbed.ipynb* - This notebook compares and contrasts the developed models according to key metrics and generates plots (Appendix A).

4.3 Limitations

The primary limitations faced in the training and validation process have been high computational demand, a limited dataset and difficulty of model interpretation. The

Faster R-CNN with NASnet model is relatively slow and had an average speed of 2.56s per inference on the MEP dataset. This means that 20,000 training steps took more than 14 hours to complete. With the limits on GPU availability in Colab, training multiple models to conduct a sensitivity analysis experiment was a time-intensive process. In further research that builds upon this dissertation, the existing setup could be scaled to make use of a cluster of dedicated GPU or TPU servers. This would facilitate the execution of more detailed experiments exploring a wider range of model configurations.

As previously discussed, the size and diversity of the dataset is a crucial driver of deep learning model performance. The dataset for this project did not achieve the scale of industry-standard datasets which usually have millions of object instances, labelled over 10,000s worker hours (Lin et al., 2014). Therefore, this project can only give a limited view of the accuracy that could be achieved in the practical application of Faster R-CNN to detect MEP components. However, the data on the optimal model parameters for this use case that was gathered through sensitivity analysis can be applied to future work backed by those resources.

Deep learning is often referred to as a 'black box' technology because it is difficult to explain the root causes of a model's predictions (Koh & Liang, 2017). In this dissertation, interpretations of why the model makes specific predictions are based on knowledge of the model structure and theories regarding how neural networks internally organize. Ideally, these explanations would be based on comprehensive data of the internal optimisation process. Research is underway to address this knowledge gap which should serve to provide a greater understanding of the application of deep learning to MEP detection in the future (Koh & Liang, 2017)(Castelvecchi, 2016)(Schwartz-Ziv & Tishby, 2017)(Samek et al., 2017).

5

Experimental Results

5.1 Benchmark Test

The Faster R-CNN model was trained in its default configuration on the non-augmented MEP dataset to establish a performance benchmark. As shown in Figure 5.1, the mAP@0.5 of the model on the validation data rose to a maximum of 0.917 after 16,387 training steps. This means that at peak precision an average of 91.7% of the benchmark model's detections were accurate. By 20,000 steps, it was clear that growth in performance had stagnated, as seen in Figure 5.2. This signalled that overfitting was occurring and therefore, the training process was stopped. When the model's performance on the validation data begins to deteriorate, this is an indication that the model is learning feature patterns that are valid when applied to the training data but do not generalise to new data.

In Figure 5.3, the improvement in the model's accuracy during training is shown using validation trial visualisations. At 1,109 training steps the model was not able to differentiate between a white radiator and a white wall. This may have been because it learned the expected colour range but not the more complex combination of edge features that identify a radiator. By step 5,404 the model could correctly detect the heater in the image.

As shown in Figure 5.4, AP varied greatly between object classes. Double sockets exhibited the highest peak AP at 0.941, followed by single sockets at 0.934, and radiators at 0.893.

There are many possible explanations for this skewed performance distribution. As detailed in Section 3.2, there are more double sockets in the dataset than there

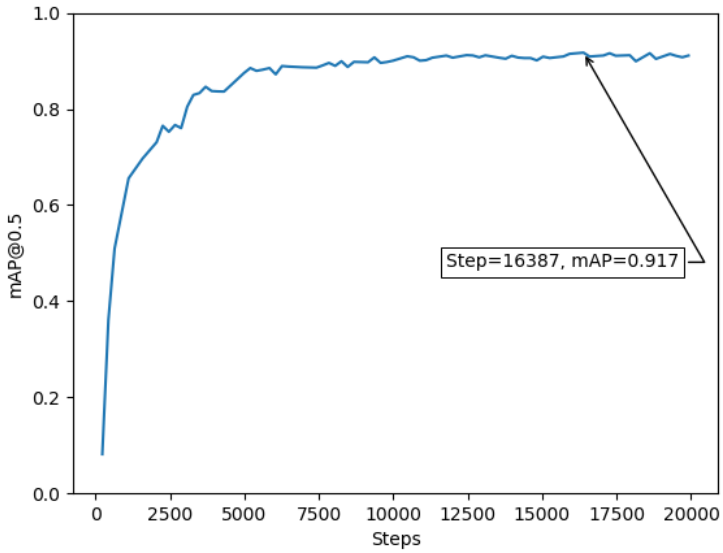
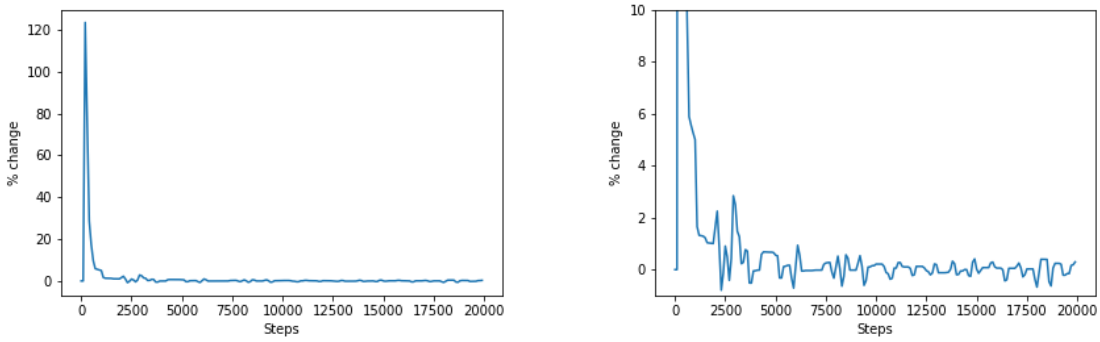


Figure 5.1: mAP vs Training Steps



(a) Full scale (b) Zoomed in

Figure 5.2: % change in mAP over the next 100 steps

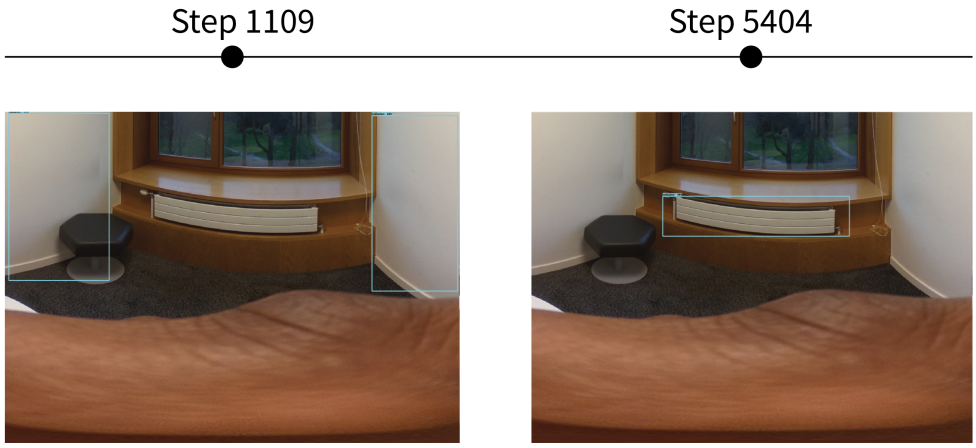


Figure 5.3: Improvement of model on validation data

are single sockets and radiators. Therefore, during training, the model will encounter examples with double sockets more frequently than the other classes, and the loss

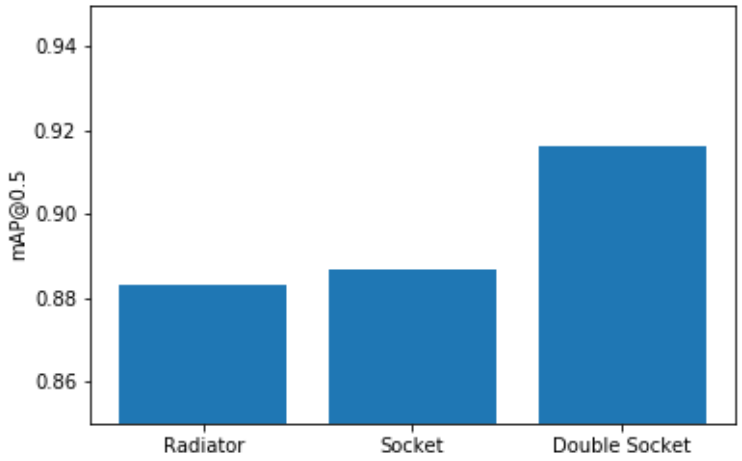


Figure 5.4: mAP by Object Class

function will adjust internal variables accordingly. This imbalance results in the Faster R-CNN being heavily biased towards the double socket class.

The sockets in the data also have a simpler set of identifying features than the radiators. The radiators vary in the shape of the valve, the ridging pattern on the panel, aspect ratio and a range of other characteristics. Due to UK regulation the sockets usually only differ in colour, as seen in Figure 5.5.

The AR@10 of the model reached a maximum of 0.731, as seen in Figure 5.6. This metric indicates that, on average, 73.1% of the sockets and radiators in the validation images were detected. Comparing this recall with the relatively high precision, it is evident that although most of the Faster R-CNN's predictions were accurate there were a high number of false negatives where objects of interest were missed entirely.

These results prove that the Faster R-CNN achieves a high level of accuracy when transfer learning is used to retrain it for socket and radiator detection. Therefore, Objective 3.1, to retrain a deep learning model to detect MEP components in 360 images, has been fulfilled. In the following sections, data augmentation and sensitivity analysis of the model parameters will be used to optimise the Faster R-CNN to this use case.

Figure 5.5: Differentiation in Radiators vs Sockets

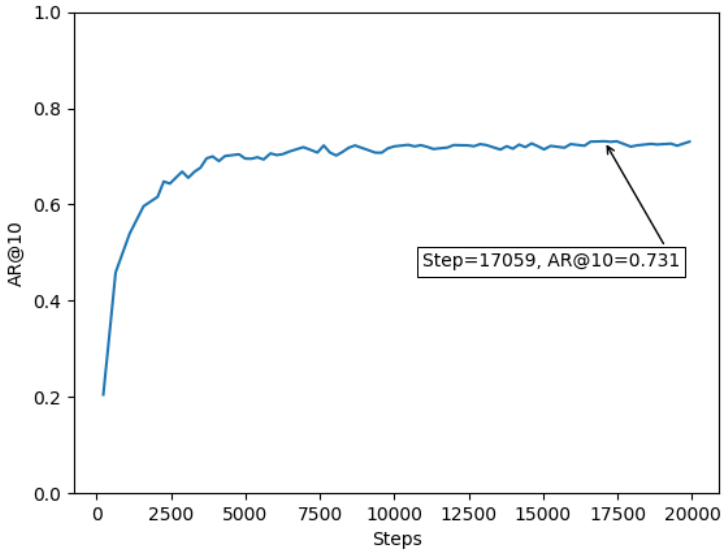


Figure 5.6: AR vs Training Steps

5.2 Impact of Data Augmentation

5.2.1 Horizontal Flip

Data augmentation creates new images by performing transformations on existing images. The horizontal flip is a common augmentation technique where new pictures are made by mirroring existing ones, as seen in Figure 5.7. Only the training data was augmented to ensure that validation results from this experiment could be equitably compared to the benchmark.

Figure 5.7: Application of Horizontal Flip

In this experiment, the horizontal flip was randomly applied to the images with a 50% probability. Effectively, this increased the number of potential training examples from 240 to 360. Randomising the transformation prevents the model from learning the pattern of data augmentation and accounting for that when making predictions. Memorising this pattern would improve model accuracy on this dataset, but that knowledge would hinder performance when applying the model to new data (Srivastava et al., 2014).

Training the Faster R-CNN with an augmented dataset significantly improved accuracy. Peak mAP rose from the 0.917 benchmark to 0.943 as shown in Figure 5.8. Therefore, the fraction of the detections the model made that were accurate increased. This could be due to a growth in true-positive detections, a fall in false-positive detections or both. It is evident that the model had a more diverse range of examples to study and learnt patterns it may not have discovered otherwise, which enhanced its performance (Shorten & Khoshgoftaar, 2019). The maximum AR exhibited during validation also rose from 0.731 to 0.766 as seen in Figure 5.9. This is evidence that the Faster R-CNN was able to identify and locate sockets and radiators that it had previously been unable to.

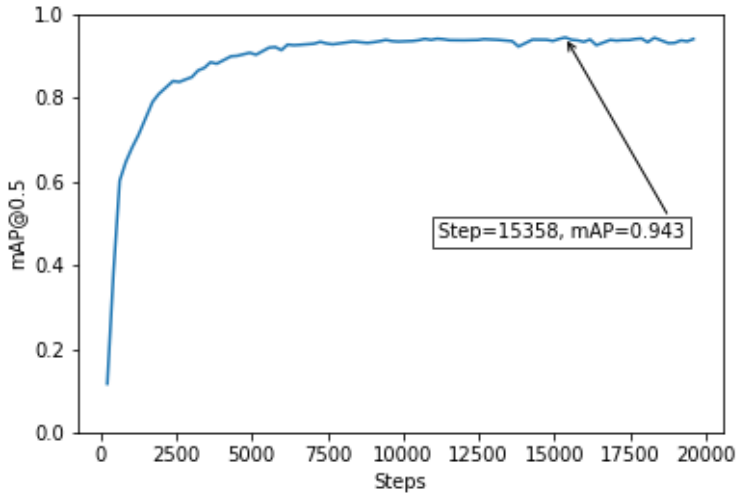


Figure 5.8: mAP with horizontal flip

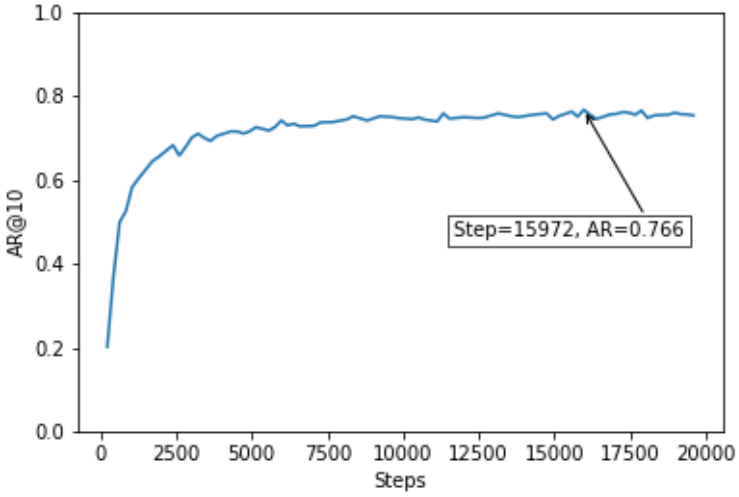


Figure 5.9: AR with horizontal flip

5.2.2 Patch Gaussian

Patch Gaussian is an augmentation strategy developed by the Google Brain team. Much of the research on advancing deep learning has focused on developing new architectures that deliver high accuracy on datasets with high-quality images; a trait referred to as 'clean accuracy'. For object detection models to perform well in realistic settings, they also need to be able to cope with corruption in the image data, a quality known as 'robustness' (Lopes et al., 2019). Patch Gaussian maximises both of these properties by combining two standard augmentation techniques.

Cutout removes square sections of the input image at randomised locations.

This technique pushes the model to make decisions based on the entire context of the picture instead of relying on a small set of visual features (DeVries & Taylor, 2017). Cutout improves the 'clean accuracy' of a neural network (Lopes et al., 2019).

Gaussian augmentation involves injecting Gaussian noise into the pixels of the input image. This type of noise occurs naturally when digital pictures are taken. It is simulated by adding an RGB value to each pixel that is sampled from a normal distribution function with a predefined standard deviation (Montserrat et al., 2017). Gaussian augmentation serves to bolster the neural network's robustness to noise (Lopes et al., 2019).

Patch Gaussian combines Cutout and Gaussian augmentation by adding a square of Gaussian noise to a random location on the input image, as seen in Figure 5.10 (Lopes et al., 2019). In this experiment, the square was applied at random sizes between $300 pixels^2$ and $600 pixels^2$ with a random standard deviation between 0.1 and 1. The probability that an image would be augmented was set to 0.5.

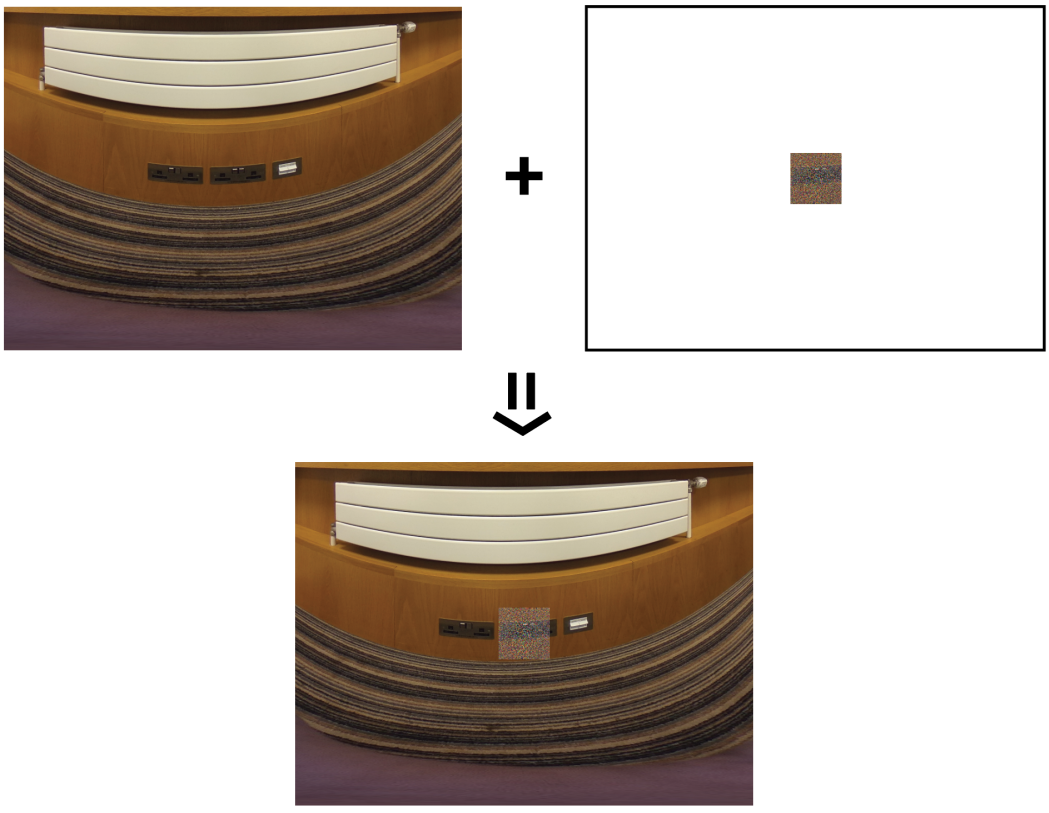


Figure 5.10: Application of Patch Gaussian

This augmentation had a significant positive effect on Faster R-CNN precision, raising peak mAP from the 0.917 benchmark to 0.941 as seen in Figure 5.11. Recall also improved. As shown in Figure 5.12 peak AR grew from 0.731 to 0.783. The horizontal flip and Patch Gaussian policies resulted in an almost equal boost in

precision, but Patch Gaussian had a much greater impact on AR. This leap in performance could be because Patch Gaussian generates new samples that are more challenging than those created by mirroring, and therefore, contributes more to the model's growth (Shrivastava et al., 2016). In addition, the neural networks's enhanced resistance to noise would have enabled it to better identify sockets and radiators that were affected by image corruption, thereby reducing the number of false negatives.

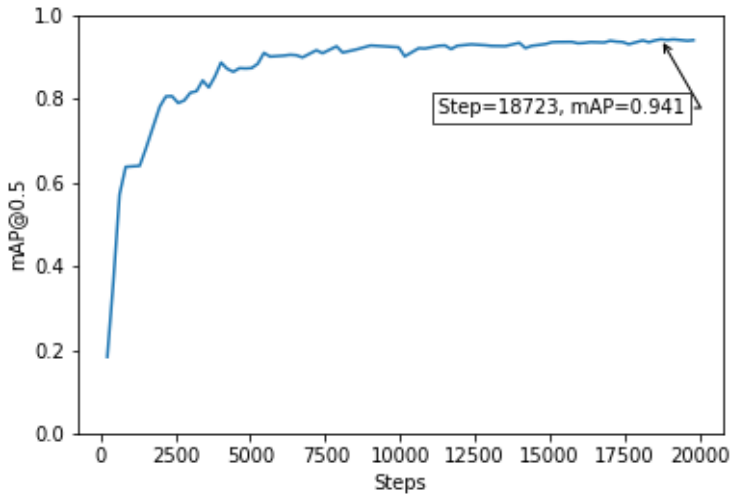


Figure 5.11: mAP with Patch Gaussian

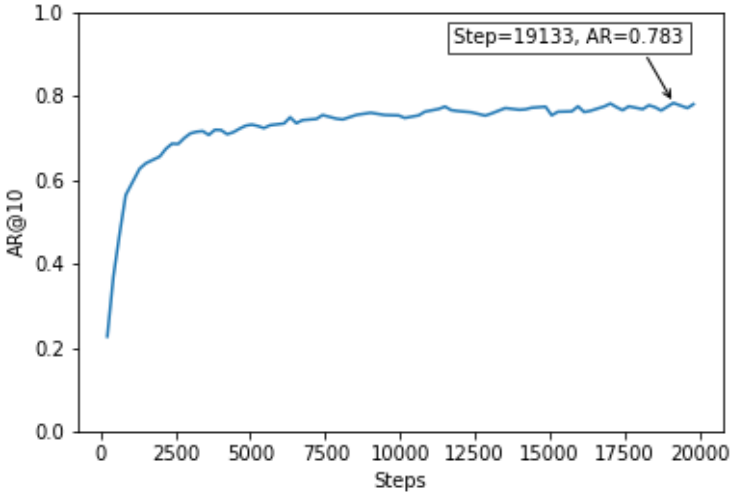


Figure 5.12: AR with Patch Gaussian

Interestingly, Patch Gaussian augmentation had the greatest positive impact on the performance on the hardest examples in the validation data. As seen in Figure 5.13, it increased precision on small objects by 8% and reduced precision on medium-sized objects by 6%, where the size ranges are as defined in Section 3.2. These results indicate that small objects were the worst affected by Gaussian noise,

and the model's improved noise resistance allowed it to detect them more accurately. As previously mentioned, this noise is naturally generated during image acquisition by photographic sensors such as the Complementary Metal Oxide Semiconductor (CMOS) (Boyat & Joshi, 2015).

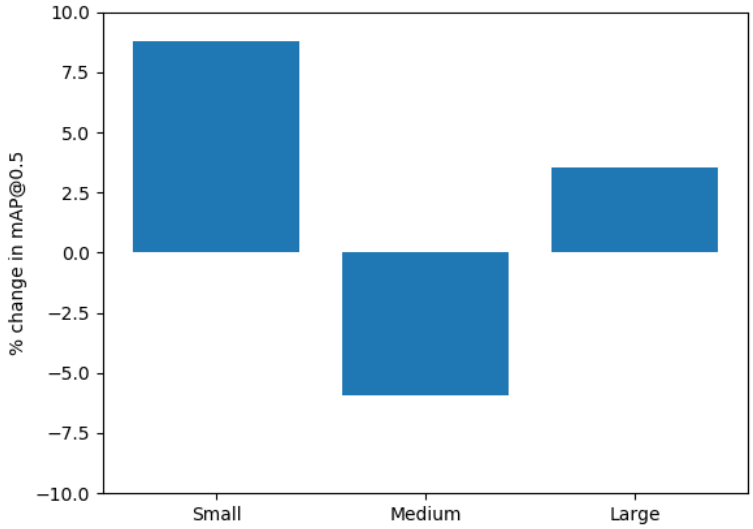


Figure 5.13: % change in peak mAP according to size of MEP object

5.2.3 Combined Augmentation

The horizontal flip and Patch Gaussian augmentation policies can be implemented together to positive effect. As seen in Figures 5.14 and 5.15, when the two are used in conjunction mAP rises to a peak of 0.951 and maximum AR is 0.779. The neural network achieves higher precision and recall than was possible when only one augmentation technique was used. They complement each other; the Patch Gaussian serves to boost the neural networks resilience to common corruption without compromising clean accuracy, while the horizontal flip further increases the size of the dataset.

Based on these results, it is clear that in future deep learning MEP detection solutions a combination of augmentation strategies should be used to achieve the best performance.

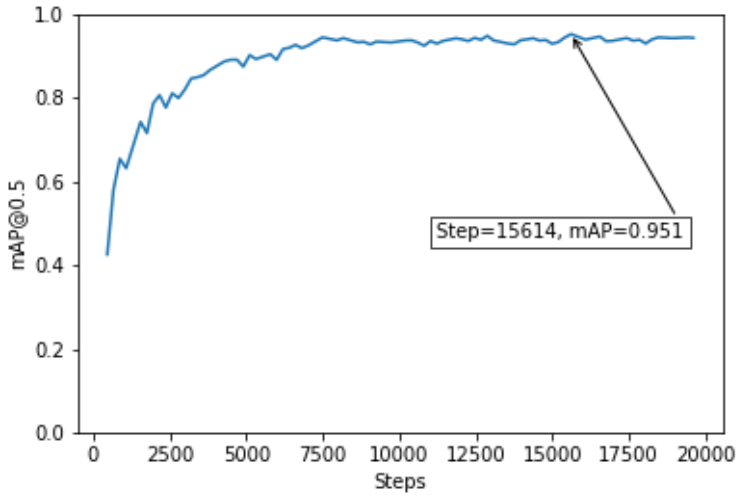


Figure 5.14: mAP with Patch Gaussian and horizontal flip

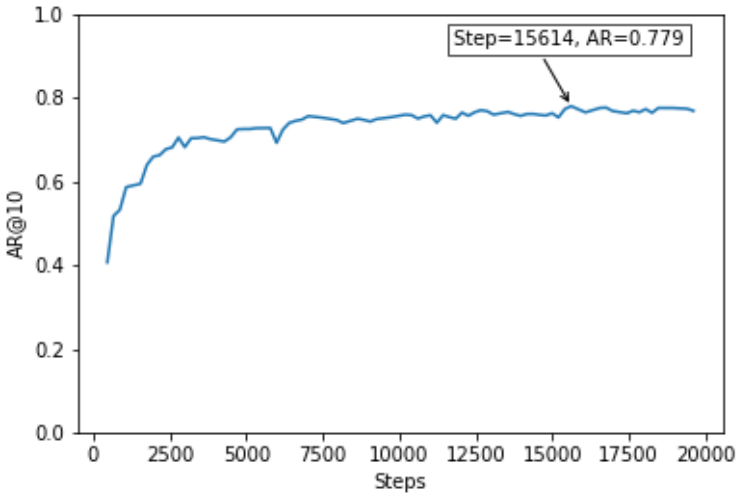


Figure 5.15: AR with Patch Gaussian and horizontal flip

5.3 Sensitivity Analysis

In this section, the influence of a range of model parameters on performance will be interrogated. The different model configurations are compared based on their peak mAP@0.5 and AR@10. Horizontal flip augmentation was used in all of the experiments.

In the benchmark test, growth stagnated well before 20,000 steps, as shown in Figure 5.3. Therefore, in all of the experiments, training was stopped at 20,000 steps. Continuing to develop the models past this point would cost time and computational resources without a significant return in performance.

5.3.1 Training Scale

During training, the Faster R-CNN resizes all the input images to the same square area, and this size is known as the training scale. In this experiment, the models were developed using a range of training scales from 800×800 pixels to 1300×1300 pixels. As shown in Figure 5.16, this parameter proved to have a significant impact on model performance. The Tesla K80 GPU available for training models on Colab was not sufficient to test training scales above 1300×1300 pixels. Attempting to do so resulted in Out of Memory Errors.

Plotting the peak mAP against the training scale revealed a directly proportional relationship as seen in Figure 5.16. This indicates that higher performance would be achieved above 1300×1300 pixels.

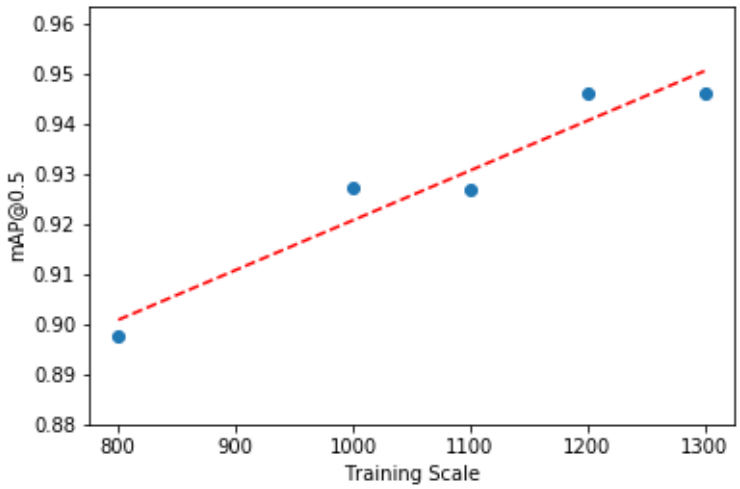


Figure 5.16: mAP vs Training Scale

This trend can be rationalised based on the model structure. In the benchmark test, the 1792×1344 images from the dataset were resized to the default training scale of 1200×1200. Therefore, the size of the MEP objects in the photos decreased. Lowering the training scale, for example, from 1200×1200 to 800×800, further reduces the size of these objects, as seen in Figure 5.17. Objects with less pixel information are harder for a model to recognise because they offer fewer visual features (Liang et al., 2018).

The design of the Faster R-CNN compounds this effect. A Faster R-CNN proposes potential object locations and classifications based on a feature map projected on a deep convolutional layer (Ren et al., 2015). This feature map has been scaled down significantly from the size of the input image by the feature extractor, which in this case is a NASnet. As shown in Figure 5.18, there are several Reduction Cells

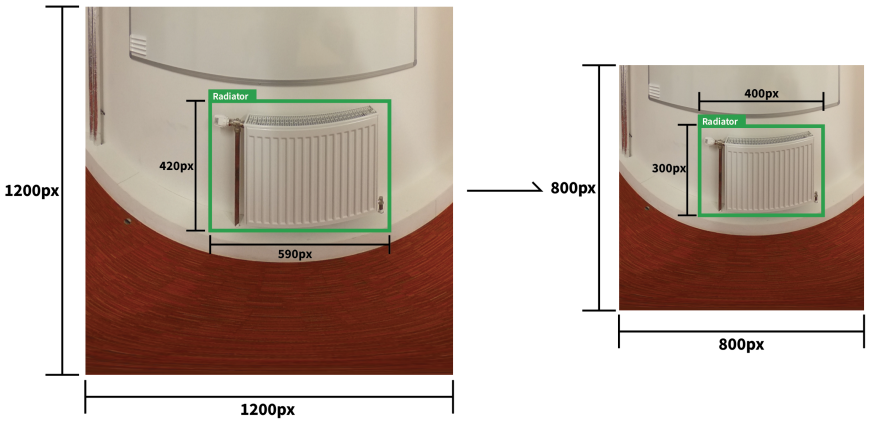


Figure 5.17: Effect of downscaling on MEP object size

in the NASnet architecture which reduce the feature map height and width by a factor of two (Zoph et al., 2018). In the NASNet output map, objects that were already small are unlikely to exhibit features that are useful for classification.

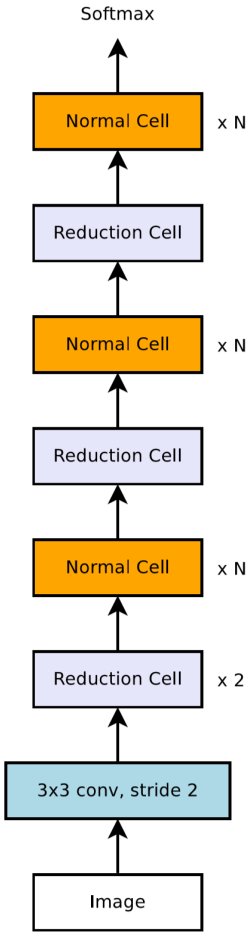


Figure 5.18: Structure of NASnet (Zoph et al., 2018)

Raising the training scale increases the size of the objects in the final feature map used by the Faster R-CNN. This has a positive effect on model accuracy. Based on these results, it can be concluded that in future research on the application of Faster

R-CNN to MEP detection, high performance can be achieved by using the largest possible training scale. The limitation is the quality and quantity of GPUs that are available for training. The directly proportional relationship between training scale and mAP shown in Figure 5.16 aligns with observations made in published research applying Faster R-CNN to vehicle detection (Fan et al., 2016) and crack detection (Alipour et al., 2019).

The training scale also had a positive effect on AR as shown in Figure 5.19. At higher training scales the percentage of sockets and radiators that were accurately identified by the model increased. The 1000×1000 training scale model did not match this trend. This could be the result of a mistake made in the configuration process.

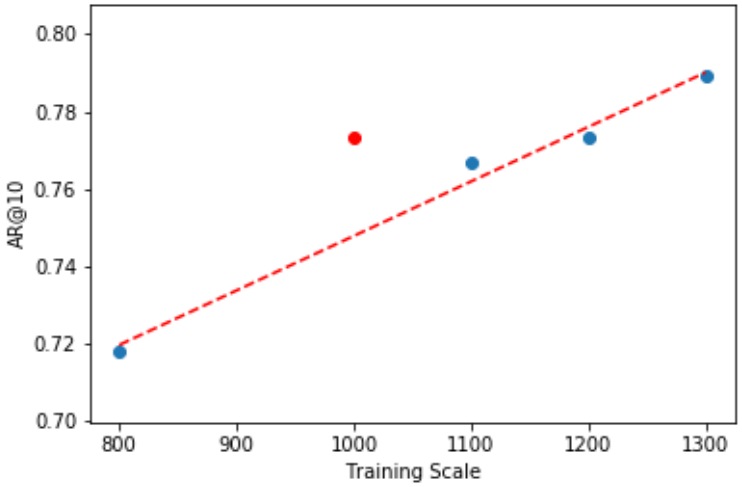


Figure 5.19: AR vs Training Scale

In the following experiments, the default training scale of 1200 was used for the sake of computational efficiency.

5.3.2 Sliding Window Stride

The Regional Proposal Network (RPN) of the Faster R-CNN generates proposals by applying a sliding window to the feature map output by the feature extractor. As seen in Figure 5.20, the sliding window is a small network that moves across the feature map n steps at a time, where n is described as the stride of the window, and takes as input an $x \times x$ sized region of the map. At each step of the sliding window, the RPN simultaneously evaluates several possible regional proposals at different sizes and aspect ratios. These rectangular reference regions are known as anchors (Ren et al., 2015).

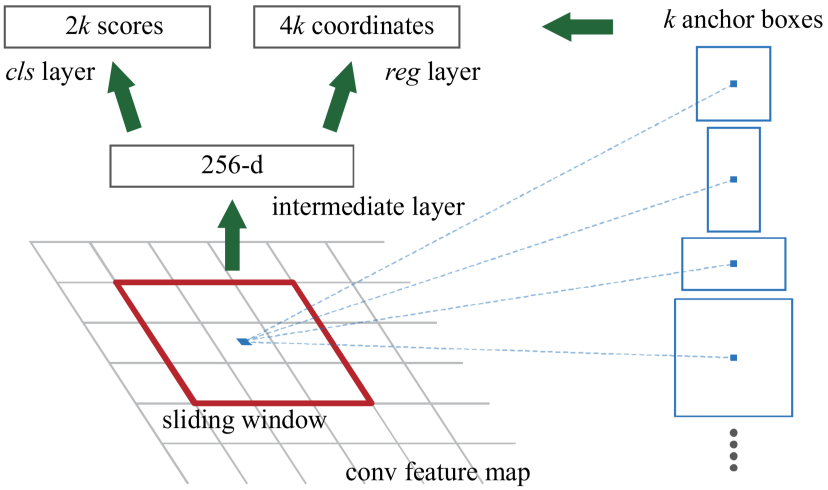


Figure 5.20: RPN Sliding Window (Ren et al., 2015)

As shown in Figure 5.21, increasing the stride of the sliding window over the feature map improved the mAP. However, published research has demonstrated that decreasing sliding window stride should have a positive effect on precision which is in direct contradiction with these results (Huang et al., 2017)(Zhang et al., 2016).

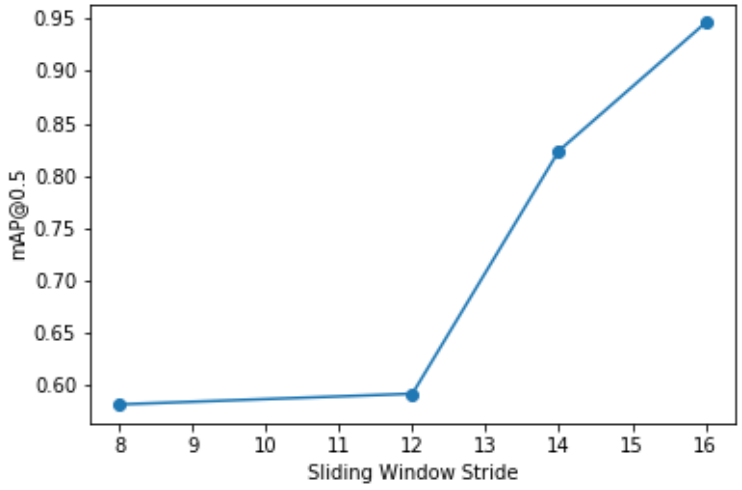


Figure 5.21: mAP vs Sliding Window Stride

Decreasing the stride results in more anchors being generated on the feature map. This effect usually increases mAP because more regional proposals are evaluated, and therefore small objects are less likely to be missed or skipped over entirely (Ren et al., 2018). However, decreasing stride can also have a negative impact because the increased quantity of anchors being evaluated can result in more false positives. A smaller stride can also contribute to overfitting as the model memorises more of the visual features in the training data, some of which are just noise created by errors in data capture. In this use case, the objects of interest are few, large with respect to the image and far apart. Therefore, when the stride was decreased the improvement

in mAP due to a lower small object 'miss rate' may have been overpowered by overfitting and the false positives detected in the background (Bastian & Jiji, 2017).

As shown in Figure 5.22, when the Faster R-CNN was configured with the benchmark stride of 16, total loss fell as training progressed. When the stride was decreased, the total loss was higher and adopted an upwards trend. Total loss is a function of the difference between the detection results and the ground truth (Ren et al., 2018). Therefore, rising total loss indicates a divergence between the model output and reality. These results support the theory that overfitting occured early in the training progress of low stride models and negatively impacted performance.

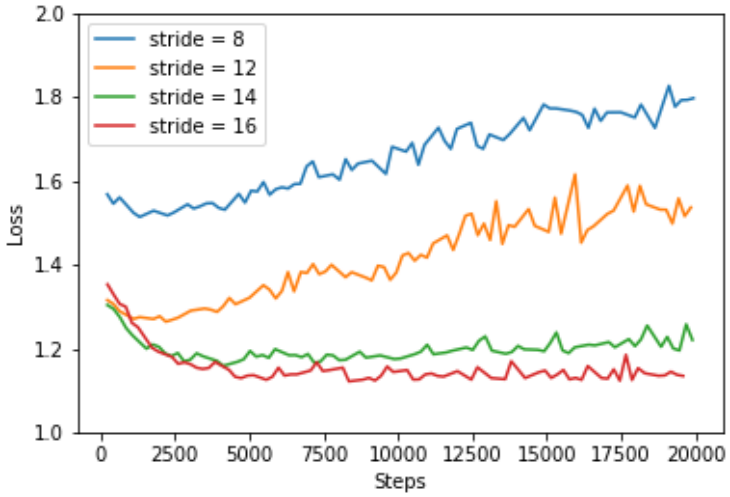


Figure 5.22: Impact of sliding window stride on loss

The AR of the Faster R-CNN was directly proportional to the sliding window stride setting, as shown in Figure 5.23. This trend is further proof that using a low stride harmed accuracy because the models learnt feature patterns that did not generalise to new data. The Faster R-CNN with an 8 step sliding window assessed more anchor proposals and identified fewer sockets and radiators than one with a 16 step sliding window.

It is evident that customising the RPN sliding window to each dataset is essential to improving performance. Sliding window characteristics that work best for a new use case may not reflect the results of published research articles, many of which focus on the same industry-standard datasets.

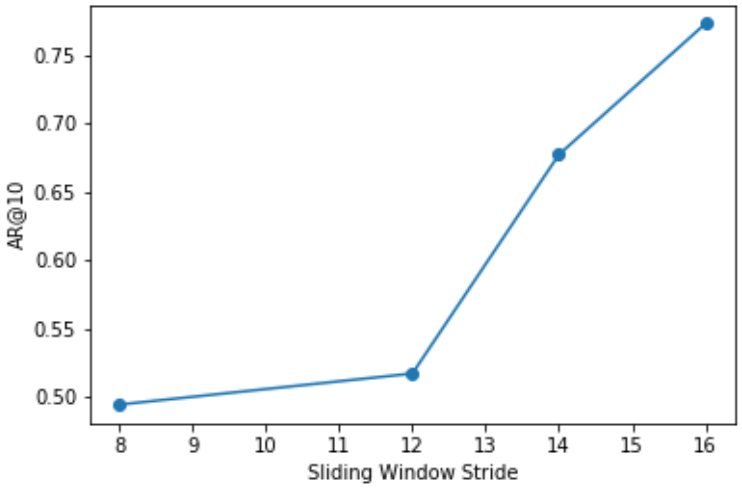


Figure 5.23: AR vs Sliding Window Stride

5.3.3 Number of RPN Proposals

As previously mentioned, the RPN component of the Faster R-CNN suggests regions of interest that inform the model's detections. In this experiment, the Faster R-CNN was trained using different settings for the maximum number of proposals the RPN could make. Evaluating the impact of this parameter on AR, as shown in Figure 5.24, revealed that raising the number of suggestions from the RPN marginally increased the proportion of sockets and radiators that were accurately detected.

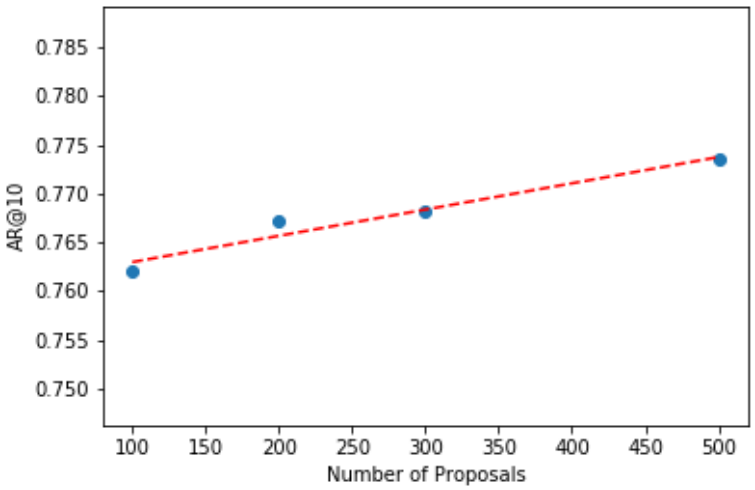


Figure 5.24: AR vs Number of Proposals

It was expected that mAP would have a directly proportional relationship with the number of proposals, similar to the AR. This was the outcome, as seen in Figure 5.25, but the model trained with the maximum RPN proposal setting of 100

did not fit the trend. The anomalous result had a higher performance as measured by mAP than other models that generated more proposals.

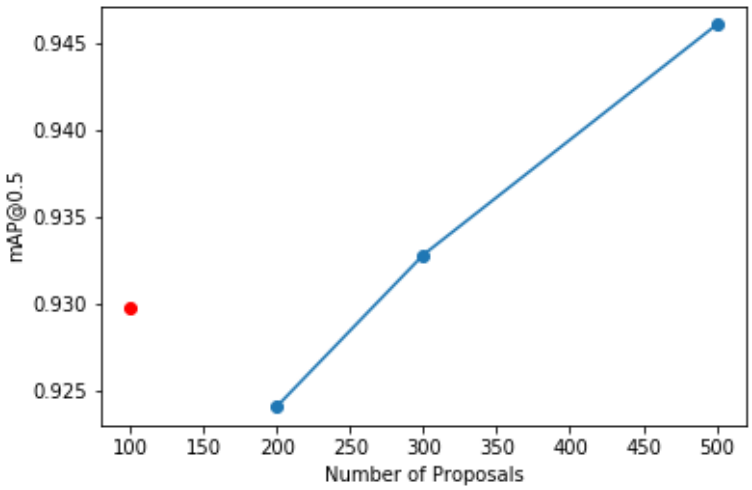


Figure 5.25: mAP vs Number of Proposals

As shown in Eq. 5.6, the fact that the 100 proposal model had a lower AR than the model with a 200 proposal RPN indicates that it generated fewer true positive results.

$$\text{Recall} = \frac{\text{No of true positive detections (TP)}}{\text{true positive detections (TP)} + \text{false negative detections (FN)}} \tag{5.1}$$

$$\text{AR} = \frac{\sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i}}{n} \tag{5.2}$$

The total number of sockets and radiators in the validation data (TSR) is constant in all experiments,

$$\text{AR} = \frac{\sum_{i=1}^{n} \frac{TP_i}{TSR}}{n} = \frac{\frac{\sum_{i=1}^{n} TP_i}{TSR}}{n} \tag{5.3}$$

$$\text{AR}_{100p} < \text{AR}_{200p} \tag{5.4}$$

$$\frac{\frac{\sum_{i=1}^{n} TP_{100pi}}{TSR}}{n} < \frac{\frac{\sum_{i=1}^{n} TP_{200pi}}{TSR}}{n} \tag{5.5}$$

$$\sum_{i=1}^{n} TP_{100pi} < \sum_{i=1}^{n} TP_{200pi} \tag{5.6}$$

Precision is the fraction of true positives to number of detections. Therefore, if the 100 proposal model had a smaller number of true positives than the 200 proposal model, but a higher mAP then it must have made fewer detections. The

lower number of detections may have been a direct result of the 100 proposal model having fewer RPN suggestions to base decisions on.

Despite the anomaly, this experiment indicates that the performance of a Faster R-CNN model in identifying MEP assets can be improved by implementing a high maximum proposal RPN. However, this improvement is minuscule. Increasing the proposal rate from 300 to 500 only raised mAP by 1.42% and AR by 0.7%. Segmenting the precision data by size in Figure 5.26 offers evidence as to why the change is so small. For large and medium objects, high accuracy is achieved within the first 100 proposals. At this point, performance is similar or even marginally higher than it would be with a larger proposal rate; for example, 300. Only performance on the smallest, hardest examples consistently rises with the number of proposals, possibly because they are less likely to be overlooked by the RPN.

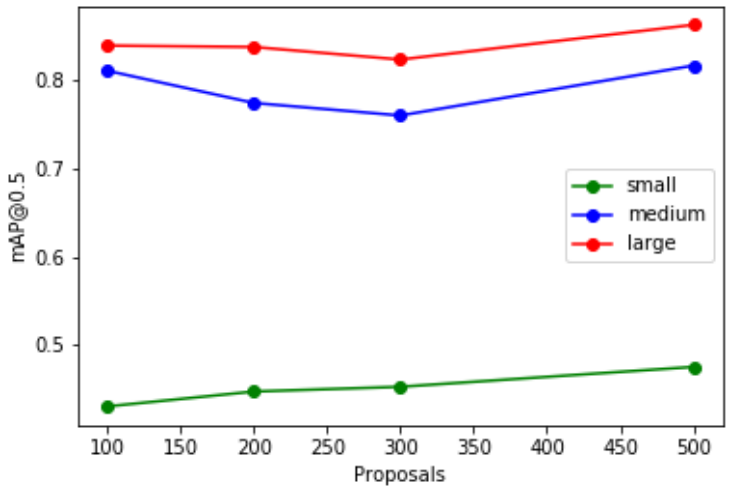
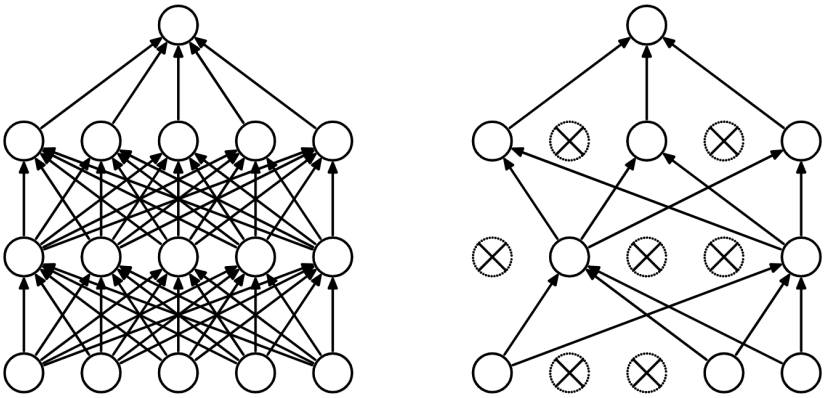


Figure 5.26: Impact of Number of Proposals on mAP according to MEP object size

5.3.4 Dropout

Dropout is a strategy for preventing overfitting that works by temporarily disconnecting internal variables from the neural network at random points in the training process as shown in Figure 5.27. During training, closely related neurons can become heavily co-adapted and, as a result, play a more active role in the detection process than others. The result is a network that is too reliant on a small set of feature patterns specific to the training data. Dropout forces internal variables that may not have actively participated otherwise to develop and thereby ensures a diversity of feature relationships are employed in the detection process (Srivastava et al., 2014)(Hinton et al., 2012).



(a) Standard Neural Net (b) After applying dropout.

Figure 5.27: Application of dropout (Srivastava et al., 2014)

The Faster R-CNN was trained using different keep probabilities for neurons in the final layers of the model, the box-classification and box-regression layers that predict the class of each detection and regress the bounding box (Ren et al., 2018). A keep probability of 1.0 meant no dropped neurons and 0.0 indicated no output from the layer. As shown in Figure 5.28, applying dropout harmed peak performance and models with low keep probabilities had smaller mAP. Dropout also had a negative impact on recall as seen in Figure 5.29. There are many potential explanations for this phenomenon.

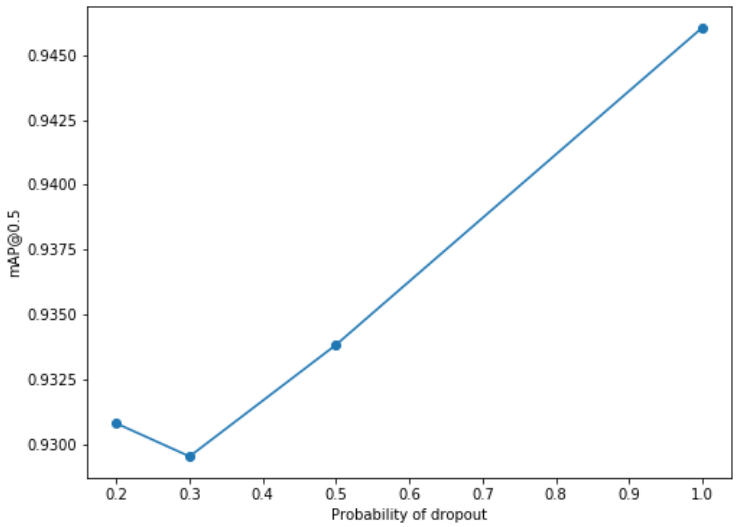


Figure 5.28: mAP vs Keep Probability

Dropout naturally reduces model capacity in the short term. It is based on the principle that the best way to prevent overfitting is to average the predictions from diverse configurations of the same model (Xiong et al., 2011). Combining results from multiple models is not feasible for large neural networks that have a

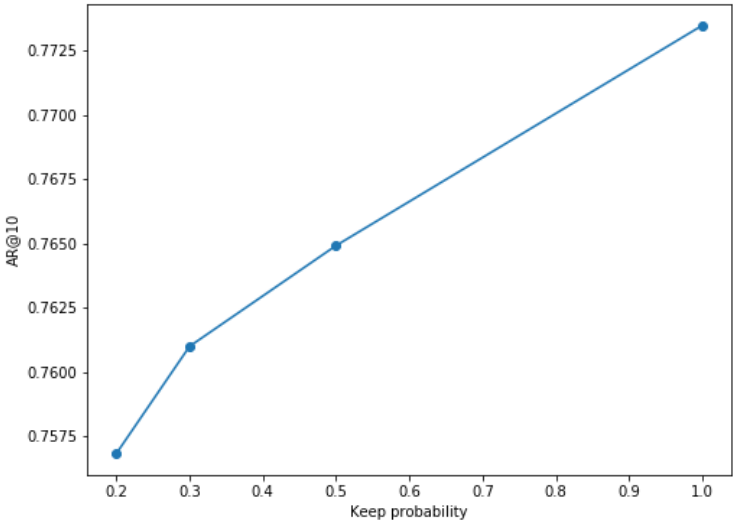


Figure 5.29: AR vs Keep Probability

high computational demand. Dropout simulates this 'Bayesian gold standard' by training a new 'thinned' network at every training step, consisting of all the retained neurons. When the network is validated without dropout, it is effectively an average of all the previous thinned networks. Each thinned network is quite weak so vast quantities of them need to be trained to generate a robust model with improved peak performance (Srivastava et al., 2014). Therefore, the 20,000 steps of training time used in this experiment may not have been sufficient to earn the positive long-term effects of dropout.

Another potential explanation for the negative impact of dropout is that the keep probabilities used in this experiment were too low. At a probability of 0.3 only 30% of the neurons in the box classifier layer would be retained at every training step. This could have resulted in underfitting meaning that the thinned models did not retain enough internal variables to learn critical visual features. Underfitting is a common side effect of overusing regularisation techniques such as DisturbLabel and dropout (Bell et al., 2015)(Xie et al., 2016). A model with a higher keep probability, for example, 0.95, may have produced better results than the other dropout models and even achieved higher performance than the benchmark 1.0.

Based on these results, it can be concluded that in future research on the application of dropout to MEP object detection, further investigation should be carried out with longer training schedules and higher keep probabilities.

5.4 Multiparameter Optimisation

This section was intended to explore how multiple parameters and data augmentations could be adjusted in combination, using the data from sensitivity analysis, to deliver an optimised model and fulfil Objective 3.4. As a result of the early suspension of practical work, the models for this experiment could not be developed as mentioned in the COVID-19 impact declaration.

It was expected that the final optimised model would use the largest training scale and highest number of RPN proposals that could be supported by the available computational resources. The sliding window stride would have been increased to its critical peak. Beyond this peak, a bigger stride would have negatively impacted performance because the RPN would have skipped over sections of the images that contained sockets and radiators. Also, dropout would have been applied in the training process if a keep probability that positively impacted mAP and AR was derived from further experimentation. Other parameters such as the learning rate and loss weights that could not be investigated due to the hard stop in practical work would also have been edited in the optimised model. A cyclical learning rate would likely have been the best learning rate to apply based on promising results in published research (Smith, 2017).

It was expected that most of these modifications would not negatively impact each other because they affect different elements of the model operation. For example, increasing the number of RPN proposals would reduce the probability of the model overlooking a small socket or radiator regardless of the training scale. The only potential adverse interaction would be between the training scale and sliding window stride. A higher training scale would result in a larger feature map being output by the NASNet. Therefore, the sliding window would need to skip a larger number of pixels in order to traverse the same proportion of the image's height or width. In other words, increasing the training scale may have negated the impact of increasing the sliding window stride.

The optimised model would also have been trained with a combination of dataset augmentations. Patch Gaussian and horizontal flip, which have been proven to work well in combination, would have been implemented together with additional strategies such as random brightness adjustment. It is also possible that applying an adaptive augmentation technology such as AutoAugment would have generated a selection and order of transformations that delivered better performance than what could be manually designed (Cubuk et al., 2018).

Building on the results presented in this dissertation, multiparameter optimisation experiments that would verify these predicted outcomes can be carried out in future research on the use of Faster R-CNN for MEP object detection.

6

Conclusion

6.1 Summary and Practical Implications

This dissertation aimed to investigate the use of deep learning neural networks to detect sockets and radiators in 360° images. A dataset of 360° images was built and used to retrain an existing Faster R-CNN model. Then, a sensitivity analysis of the deep learning model parameters was carried out to explore how best to optimise the Faster R-CNN to this use case. The results proved that neural networks could be an effective tool for detecting MEP assets and thereby add value to Scan-to-BIM frameworks.

When transfer learning was applied to the Faster R-CNN, it achieved a peak mAP of 0.917 and a peak AR of 0.731 on the MEP dataset. The sensitivity analysis revealed that increasing training scale, sliding window stride and the number of proposals generated by the RPN improved the accuracy of the neural network while low keep probability dropout harmed precision.

Implementing horizontal flip and Patch Gaussian online augmentations enhanced performance. A combination of these augmentation strategies delivered a model with a maximum mAP of 0.951, a 3.7% improvement in precision from the benchmark. This model also achieved a peak AR of 0.779, which was 6.6% higher than the benchmark.

As discussed in Section 5.1, the high precision and comparatively low recall of the model indicate that most of the Faster R-CNN's predictions were accurate, but there were many false negatives i.e. sockets and radiators that were overlooked. It is evident that in the practical application of this deep learning model, multiple images

should be captured at each location in the interior space to ensure that every MEP component is detected at least once. The operator should take photos close to the walls where the sockets, radiators and other assets are mounted to maximise the size of the devices in the images, and therefore, make it easier for the model to detect them. This strategy will lower the probability that the neural network overlooks an MEP asset.

6.2 Future Research

The experiments carried out in this project yielded promising results in the automated classification and localisation of MEP assets. However, this dissertation solely focused on the use of image data and, due to time constraints, could not explore a number of promising technologies. There are many additional areas of research that could be investigated to support the integration of MEP object detection into BIM systems, building on the work of this project.

A particularly exciting development is the use of synthetic data generation to train object detection models. Building a diverse dataset of labelled images or point clouds is a labour-intensive and challenging process. Synthetic data generation addresses this issue by creating massive amounts of artificial training examples. Published research has demonstrated the effectiveness of models trained on artificially generated data in detecting words, cars, and a range of everyday objects (Jaderberg et al., 2014)(Tremblay et al., 2018)(Peng et al., 2015). Randomly generated 3D models could serve as datasets for teaching deep learning models to detect building assets. This strategy is already being explored in Scan-to-BIM research. Quintana et al. (2018) successfully trained a mathematical algorithm to detect doors in point clouds using a combination of simulated and real interior spaces (Quintana et al., 2018).

Another area of research that requires more attention is the automatic identification of MEP assets in laser scanning data. Much like MEP detection in images, this is a field where not much work has been done. However, the application of machine learning to labelling 3D objects in point clouds has generated promising results in a variety of other use cases. Li (2017) trained a fully convolutional model to detect cars in the KITTI point cloud dataset (Li, 2017). Armeni et al. (2016) developed a learning algorithm that semantically parsed furniture and structural elements in point clouds of interior spaces (Armeni et al., 2016). Vetrivel et al. (2018) used a neural network and point clouds collected by unmanned aerial vehicles

to detect damaged buildings (Vetrivel et al., 2018). These case studies serve as an indication that machine learning may be effective in recognising MEP assets in laser scans. Entirely new datasets of MEP objects, labelled and segmented in point cloud data, will have to be built. It will be challenging because laser scanners are more expensive and difficult to use than 2D cameras (Faltýnová et al., 2016). Also, 3D laser scan data is more challenging to interpret than 2D images and may require trained technicians to annotate (Hackel et al., 2017).

The experiments carried out in this project evidenced the positive impact that effective data augmentation has on object detection model performance. Therefore, more advanced augmentation strategies should be explored in future research to increase model accuracy and prevent overfitting. There is a wealth of published research on systems that successfully increase the difficulty of training data by jointly training a 'generator' model, which aims to manufacture augmented images that induce maximal loss, and a target neural network which improves the harder the examples are (Peng et al., 2018)(Shrivastava et al., 2016)(Fawzi et al., 2016). These methods should be explored to find solutions that are best suited to the MEP use case.

Finally, research focused on the integration of detection systems into BIM software will have a significant impact on the future adoption of this technology in real facilities management and renovation projects. Specifically, work needs to be done on the automated conversion of asset classification and location data from common object detection formats into industry-standard BIM formats such as IFC (buildingSmart International, 2018).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. and Zheng, X., 2016. Tensorflow: A system for large-scale machine learning. In USENIX Association, *12th USENIX Symposium on Operating Systems Design and Implementation* (pp.265-283).

Adán, A., Quintana, B., Prieto, S. A. and Bosché, F., 2018. Scan-to-BIM for 'secondary' building components. *Advanced Engineering Informatics*, 37, pp.119–138.

Adán, A., Quintana, B., Prieto, S. A. and Bosché, F., 2020. An autonomous robotic platform for automatic extraction of detailed semantic models of buildings. *Automation in Construction*, 109, p.102963.

Ahmed, M. F., Haas, C. T. and Haas, R., 2014. Automatic Detection of Cylindrical Objects in Built Facilities. *Journal of Computing in Civil Engineering*, 28(3).

Alipour, M., Harris, D. K. and Miller, G. R., 2019. Robust Pixel-Level Crack Detection Using Deep Fully Convolutional Neural Networks. *Journal of Computing in Civil Engineering*, 33(6).

Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., fischer, M. and Savarese, S., 2016. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp.1534-1543).

Azhar, S., 2011. Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadership and management in engineering*, 11(3), pp.241-252.

Babacan, K., Chen, L. and Sohn, G., 2017. Semantic segmentation of indoor point clouds using convolutional neural network. *ISPRS Annals of the Photogrammetry,*

Remote Sensing and Spatial Information Sciences, 4, pp.101-108.

Barazzetti, L., Previtali, M. and Roncoroni, F., 2018. Can we use low-cost 360 degree cameras to create accurate 3D models?. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(2).

Bassier, M., Vergauwen, M. and Van Genechten, B., 2017. Automated classification of heritage buildings for as-built BIM using machine learning techniques. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4(2W2), pp.25-30.

Bastian, B.T. and Jiji, C.V., 2017, December. Aggregated channel features with optimum parameters for pedestrian detection. In *International Conference on Pattern Recognition and Machine Intelligence* (pp.155-161). Springer, Cham.

Bell, S., Lawrence Zitnick, C., Bala, K. and Girshick, R., 2016. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp.2874-2883).

Bengio, Y., 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade* (pp.437-478). Springer, Berlin, Heidelberg.

Bosché, F., Ahmed, M., Turkan, Y., Haas, C.T. and Haas, R., 2015. The value of integrating Scan-to-BIM and Scan-vs-BIM techniques for construction monitoring using laser scanning and BIM: The case of cylindrical MEP components. *Automation in Construction*, 49, pp.201-213.

Boyat, A.K. and Joshi, B.K., 2015. A review paper: noise models in digital image processing. *arXiv*.

buildingSmart International, 2018. *Industry Foundation Classes (IFC) - An Introduction*. [Online] Available at: https://technical.buildingsmart.org/standards/ifc/?fbclid=IwAR2IbOhdx7ww__7bbYRB1aStdpaRGxONjvrIpyvGVeJuaFACB2L-I_Q7mP4

Carneiro, T., Da Nobrega, R.V.M., Nepomuceno, T., Bian, G.B., De Albuquerque, V.H.C. and Reboucas filho, P.P., 2018. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, 6, pp.61677-61685.

Castelvecchi, D., 2016. Can we open the black box of AI?. *Nature*, 538(7623), p.20.

Chen, J., Kim, P. and Cho, Y.K., 2017. Building element recognition with thermal-mapped point clouds. In *34th International Symposium on Automation and Robotics in Construction (ISARC 2017)*.

Chen, J., Kira, Z. and Cho, Y.K., 2019. Deep learning approach to point cloud scene understanding for automated scan to 3D reconstruction. *Journal of Computing in Civil Engineering*, 33(4).

COCO Dataset, 2019. *Detection Evaluation*. [Online] Available at: http://cocodataset.org/#detection-eval

Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V. and Le, Q.V., 2019. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 113-123).

Czerniawski, T., Nahangi, M., Haas, C. and Walbridge, S., 2016. Pipe spool recognition in cluttered point clouds using a curvature-based shape descriptor. *Automation in Construction*, 71, pp.346-358.

Deng, X., Liu, Q., Deng, Y. and Mahadevan, S., 2016. An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, 340, pp.250-261.

DeVries, T. and Taylor, G.W., 2017. Improved regularization of convolutional neural networks with cutout. *arXiv*.

Díaz-Vilariño, L., González-Jorge, H., Martínez-Sánchez, J. and Lorenzo, H., 2015. Automatic LiDAR-based lighting inventory in buildings. *Measurement*, 73, pp.544-550.

El-Omari, S. and Moselhi, O., 2008. Integrating 3D laser scanning and photogrammetry for progress measurement of construction work. *Automation in construction*, 18(1), pp.1-9.

Eruhimov, V. and Meeussen, W., 2011, September. Outlet detection and pose estimation for robot continuous operation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2941-2946). IEEE.

Faltýnová, M., Matoušková, E., Šedina, J. and Pavelka, K., 2016. building facade documentation using laser scanning and photogrammetry and data implementation into BIM. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 41.

Fan, Q., Brown, L. and Smith, J., 2016, June. A closer look at Faster R-CNN for vehicle detection. In *2016 IEEE intelligent vehicles symposium (IV)* (pp. 124-129). IEEE.

Farhadi, A., Forsyth, D. and White, R., 2007, June. Transfer learning in sign language. In *2007 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-8). IEEE.

Fawzi, A., Samulowitz, H., Turaga, D. and Frossard, P., 2016, September. Adaptive data augmentation for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)* (pp. 3688-3692). IEEE.

Goldsborough, P., 2016. A tour of tensorflow. *arXiv*.

Gould, S., Baumstarck, P., Quigley, M., Ng, A.Y. and Koller, D., 2008, October. Integrating visual and range data for robotic object detection.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J.D., Schindler, K. and Pollefeys, M., 2017. Semantic3d. net: A new large-scale point cloud classification benchmark. *arXiv*.

Hamledari, H., McCabe, B. and Davari, S., 2017. Automated computer vision-based detection of components of under-construction indoor partitions. *Automation in Construction*, 74, pp.78-94.

Henderson, P. and Ferrari, V., 2016, November. End-to-end training of object class detectors for mean average precision. In *Asian Conference on Computer Vision* (pp. 198-213). Springer, Cham.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv*.

Hosang, J., Benenson, R., Dollár, P. and Schiele, B., 2015. What makes for effective detection proposals?. *IEEE transactions on pattern analysis and machine intelligence*, 38(4), pp.814-830.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., fischer, I., Wojna, Z., Song, Y., Guadarrama, S. and Murphy, K., 2017. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7310-7311).

Huang, J. and You, S., 2013, June. Detecting objects in scene point cloud: A

combinational approach. In *2013 International Conference on 3D Vision-3DV 2013* (pp. 175-182). IEEE.

Hui, J., 2018. *mAP (mean Average Precision) for Object Detection.* [online] Available at: https://medium.com/@jonathanhui/map-mean-average-precision-forobject-detection-4

Hulstaert, L., 2018. *Going deep into object detection.* [online] Available at: https://towardsdatascience.com/going-deep-into-object-detectionbed442d92b34

Jaderberg, M., Simonyan, K., Vedaldi, A. and Zisserman, A., 2014. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv.*

Kampffmeyer, M., Salberg, A.B. and Jenssen, R., 2016. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 1-9).

Kinsley, H., 2017, *Tensorflow Object Detection API Tutorial.* [online] Available at: https://pythonprogramming.net/introduction-use-tensorflow-objectdetection-api-tutor

Koh, P.W. and Liang, P., 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 1885-1894). JMLR.

Kovalev, V., Kalinovsky, A. and Kovalev, S., 2016. Deep learning with theano, torch, caffe, tensorflow, and deeplearning4j: Which one is the best in speed and accuracy?. In *XIII International Conference on Pattern Recognition and Information Processing*

Krispel, U., Evers, H.L., Tamke, M., Viehauser, R. and Fellner, D.W., 2015. Automatic texture and orthophoto generation from registered panoramic views. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5), p.131.

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Laing, R., Leon, M., Mahdjoubi, L. and Scott, J., 2014. Integrating rapid 3D data collection techniques to support BIM design decision making. *Procedia Environmental Sciences*, 22, pp.120-130.

Lawrence, J., Malmsten, J., Rybka, A., Sabol, D.A. and Triplin, K., 2017. Comparing tensorflow deep learning performance using cpus, gpus, local pcs and cloud.

LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *nature*, 521(7553), pp.436-444.

Li, B., 2017, September. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1513-1518). IEEE.

Li, J., Chen, K., Lin, W., See, J., Wang, J., Duan, L., Chen, Z., He, C. and Zou, J., 2019. Towards accurate one-stage object detection with AP-loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5119-5127).

Li, J., Liang, X., Wei, Y., Xu, T., Feng, J. and Yan, S., 2017. Perceptual generative adversarial networks for small object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1222-1230).

Liang, Z., Shao, J., Zhang, D. and Gao, L., 2018, September. Small object detection using deep feature pyramid networks. In *Pacific Rim Conference on Multimedia* (pp. 554-564). Springer, Cham.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

Lopes, R.G., Yin, D., Poole, B., Gilmer, J. and Cubuk, E.D., 2019. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv*.

Ma, Z. and Liu, S., 2018. A review of 3D reconstruction techniques in civil engineering and their applications. *Advanced Engineering Informatics*, 37, pp.163-174.

Maalek, R., Lichti, D.D. and Ruwanpura, J.Y., 2019. Automatic recognition of common structural elements from point clouds for automated progress monitoring and dimensional quality control in reinforced concrete construction. *Remote Sensing*, 11(9), p.1102.

Maji, S. and Malik, J., 2009, June. Object detection using a max-margin hough transform. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1038-1045). IEEE.

Meeussen, W., Wise, M., Glaser, S., Chitta, S., McGann, C., Mihelich, P., Marder-Eppstein, E., Muja, M., Eruhimov, V., Foote, T. and Hsu, J., 2010, May. Autonomous door opening and plugging in with a personal robot. In *2010 IEEE International Conference on Robotics and Automation* (pp. 729-736). IEEE.

Michie, Donald, 1968. "Memo" functions and machine learning." *Nature*, 218, 19-22.

Mohri, M., Rostamizadeh, A. and Talwalkar, A., 2018. *Foundations of machine learning*. MIT press.

Montserrat, D.M., Lin, Q., Allebach, J. and Delp, E.J., 2017. Training object detection and recognition CNN models using data augmentation. *Electronic Imaging*, 2017(10), pp.27-36.

Oksuz, K., Can Cam, B., Akbas, E. and Kalkan, S., 2018. Localization recall precision (LRP): A new performance metric for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 504-519).

Peng, X., Sun, B., Ali, K. and Saenko, K., 2015. Learning deep object detectors from 3d models. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1278-1286).

Peng, X., Tang, Z., Yang, F., Feris, R.S. and Metaxas, D., 2018. Jointly optimize data augmentation and network training: Adversarial data augmentation in human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2226-2234).

Perez, L. and Wang, J., 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv*.

Perkel, J.M., 2018. Why Jupyter is data scientists' computational notebook of choice. *Nature*, 563(7732), pp.145-147.

Pont-Tuset, J., 2020. *Open Images V6 — Now Featuring Localized Narratives*. [online] Available at: https://ai.googleblog.com/2020/02/open-images-v6-now-featuringloc
html

Quintana, B., Prieto, S.A., Adán, A. and Bosché, F., 2018. Door detection in 3D coloured point clouds of indoor environments. *Automation in Construction*, 85, pp.146-166.

Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

Ren, Y., Zhu, C. and Xiao, S., 2018. Small object detection in optical remote sensing images via modified faster R-CNN. *Applied Sciences*, 8(5), p.813.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. and Savarese, S., 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 658-666).

Samek, W., Wiegand, T. and Müller, K.R., 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv*.

Shwartz-Ziv, R. and Tishby, N., 2017. Opening the black box of deep neural networks via information. *arXiv*.

Shorten, C. and Khoshgoftaar, T.M., 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), p.60.

Shrivastava, A., Gupta, A. and Girshick, R., 2016. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 761-769).

Simonelli, A., Bulo, S.R., Porzi, L., López-Antequera, M. and Kontschieder, P., 2019. Disentangling monocular 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1991-1999).

Smith, L.N., 2017, March. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 464-472). IEEE.

Solomon, N., 2019. *Training an Object Detection Model with Tensorflow API using Google COLAB*. [online] Available at: https://medium.com/analytics-vidhya/training-an-object-detection-modelwith-tensorflow-api-using-google-colab-4f9a688d5

Son, H., Kim, C. and Kim, C., 2015. Fully automated as-built 3D pipeline extraction method from laser-scanned data based on curvature computation. *Journal of Computing in Civil Engineering*, 29(4), p.B4014003.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1), pp.1929-1958.

Tensorflow, 2015. *TensorBoard: Tensorflow's visualization toolkit*. [online] Available at: https://www.tensorflow.org/tensorboard

Tensorflow, 2020. *Object Detection API Demo*. [online] Available at: https://github.com/tensorflow/models/blob/master/research/objectdetection/objectdetection

Tran, D., 2017. *datitran/raccoon dataset*. [online] Available at: https://github.com/datitran/raccoondataset

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S. and Birchfield, S., 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 969-977).

Turcsany, D., Mouton, A. and Breckon, T.P., 2013, February. Improving feature-based object recognition for X-ray baggage security screening using primed visualwords. In *2013 IEEE International Conference on Industrial Technology (ICIT)* (pp. 1140-1145). IEEE.

Valero, E., Adán, A. and Bosché, F., 2016. Semantic 3D reconstruction of furnished interiors using laser scanning and RfiD technology. *Journal of Computing in Civil Engineering*, 30(4).

Vetrivel, A., Gerke, M., Kerle, N., Nex, F. and Vosselman, G., 2018. Disaster damage detection through synergistic use of deep learning and 3D point cloud features derived from very high resolution oblique aerial images, and multiple-kernel-learning. *ISPRS journal of photogrammetry and remote sensing*, 140, pp.45-59.

Wang, C. and Mahadevan, S., 2011, June. Heterogeneous domain adaptation using manifold alignment. In *Twenty-second international joint conference on artificial intelligence*.

Wang, Q. and Kim, M.K., 2019. Applications of 3D point cloud data in the construction industry: A fifteen-year review from 2004 to 2018. *Advanced Engineering Informatics*, 39, pp.306-319.

Weiss, K., Khoshgoftaar, T.M. and Wang, D., 2016. A survey of transfer learning. *Journal of Big data*, 3(1), p.9.

Xie, L., Wang, J., Wei, Z., Wang, M. and Tian, Q., 2016. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4753-4762).

Xiong, H.Y., Barash, Y. and Frey, B.J., 2011. Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context. *Bioinformatics*, 27(18), pp.2554-2562.

Xiong, X., Adan, A., Akinci, B. and Huber, D., 2013. Automatic creation of

semantically rich 3D building models from laser scanner data. *Automation in construction*, 31, pp.325-337.

Zhang, L., Lin, L., Liang, X. and He, K., 2016, October. Is faster r-cnn doing well for pedestrian detection?. In *European conference on computer vision* (pp. 443-457). Springer, Cham.

Zhang, Y.D., Muhammad, K. and Tang, C., 2018. Twelve-layer deep convolutional neural network with stochastic pooling for tea category classification on GPU platform. *Multimedia Tools and Applications*, 77(17), pp.22821-22839.

Zoph, B., Vasudevan, V., Shlens, J. and Le, Q.V., 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8697-8710).

Appendix A

Code Repository

The code used to automate dataset management, model training and the analysis of experimental results is stored in the following GitHub repository:

https://github.com/Jam516/MEP-object-detection