Image Instance Segmentation: Using the CIRSY System to Identify Small Objects in Low Resolution Images

Orghomisan William Omatsone

A dissertation submitted in partial fulfilment of the requirements of
Dublin Institute of Technology for the degree of
M.Sc. in Computing (Data Analytics)

January 2020

Declaration

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed:

Date:

Abstract

The CIRSY system (or Chick Instance Recognition System) is am image processing system developed as part of this research to detect images of chicks in highly-populated images that uses the leading algorithm in instance segmentation tasks, called the Mask R-CNN. It extends on the Faster R-CNN framework used in object detection tasks, and this extension adds a branch to predict the mask of an object along with the bounding box prediction.

Mask R-CNN has proven to be effective in instance segmentation and object detection tasks after outperforming all existing models on evaluation of the Microsoft Common Objects in Context (MS COCO) dataset (He, Gkioxari, Dollár, & Girshick, 2017). However, this research explores to what extent the Mask R-CNN framework can perform in instance level recognition of small objects in poorly lit images.

By leveraging on the benefits of transfer learning in training deep neural networks, this research further explores if fine tuning the Mask R-CNN algorithm can significantly improve the models performance after it has been trained after applying the weights from the implementation of the model trained on the MS COCO dataset.

The CIRSY system was trained on various synthetic datasets with varying degrees of transformation and noise applied. These datasets were built from a collection of CCTV footage of chicks in a poultry farm. The experiments conducted showed that although there were slight improvements in the model performance, these improvements were not statistically significant.

Keywords: Mask R-CNN, Instance Segmentation, MS COCO, Transfer Learning, Chicks

Acknowledgments

I would like to express my sincere thanks to Damian Gordon for his constant patience, guidance and encouragement while undertaking this project.

I would like to thank Dr. Robert Ross for providing me with the data used for this project and his encouragement to allow me pursue this topic.

I would also like to acknowledge and convey my thanks to the rest of the Technological University Dublin staff that have assisted and supported me throughout my time in the university.

Finally, I would like to thank my family who provided me with unconditional support throughout this my studies.

Contents

List of Figures

List of Tables

List of Acronyms

MS COCO	Microsoft Common Objects in Context
CV	Computer Vision
NN	Neural Network
MLP	Multi Layer Perceptrons
CNN	Convolutional Neural Networks
RCNN	Regional Convolutional Neural Networks
FCN	Fully Convolutional Network
mAP	Mean Average Precision
RPN	Region Proposal Network
IoU	Intersection over Union
LoG	Laplacian of Guassian
SIFT	Scale-invariant feature transformation
HOG	Histogram of oriented gradients
SVM	Support Vector Machine
CRISP-DM	CRoss-Industry Standard Process for Data Mining

Chapter 1

Introduction

1.1 Project Background

Computer vision (CV) is a field of study that deals with the challenge of helping computers see, and understand what they are seeing. Developments in the architecture of Neural Network (NN) algorithms have greatly contributed to the developments in the field of CV. With the developments in CV, there has been an increased importance in the use of image processing techniques for the extraction of information from images (Abdel-Maksoud, Elmogy, & Al-Awadi, 2015). Three popular image processing techniques are Object Detection, Semantic Segmentation and Instance Segmentation.

Object Detection involves identifying different objects in an image. An object refers to identifiable and standalone things in an image, as opposed to the backgrounds of an image(Alexe, Deselaers, & Ferrari, 2010). Semantic Segmentation involves classifying each pixel in an image into a class (Dhanachandra, Manglem, & Chanu, 2015; Sridevi & Mala, 2012; Abdel-Maksoud et al., 2015).Instance Segmentation is a combination of Object Detection and Semantic Segmentation. Where rather than classifying each pixel on the image into different classes, it works by detecting an object in the image and classifying it. A bounding box is the generated around the object detected and a mask is generated over the detected object to segment it from the image.

The rise of NN models led to Convolutional Neural Networks (CNN) being used as the foundation for building image processing algorithms and the more successful

algorithms were based of the Fully Convolutional Network (FCN) architecture (Long, Shelhamer, & Darrell, 2015). These milestones were the precursors to the current state-of-the-art algorithm used in Instance Segmentation called Mask R-CNN. Given the success of CNN in object detection, Mask R-CNN is an extension as it predicts if a pixel belongs to the detected object in order to generate a Mask over the object.

1.2 Project Description

Extensive work has been done on instance segmentation with well lit or high quality/resolution images where the objects in them can be easily identified, but research into poorly lit low, resolution images with small objects that are difficult to detect, even by the human eye, is lacking.

Although using just high resolution images is more beneficial than low resolution images, various limitations currently prevent abandoning low resolution images, such as the size of the images will be larger which would lead to more storage cost, as well as improved hardware cost to ensure the hardware being used can easily handle streaming of the higher quality images. Hence, low resolution images cannot be ignored. Given that majority of CCTV footage tends to be poorly lit and in a lower resolution as it is more affordable. Being able to detect objects confidently in such conditions can lead to groundbreaking work such as identifying ill animals in farms, detecting criminal activity in late night CCTV footage and much more.

In low resolution images, contrast between the edges of different objects and the background is difficult to detect even for the human eye. This makes it much harder to determine the region covered by an object. This project will investigate how the Mask R-CNN algorithm is affected by these limitations by developing the CIRSY system.

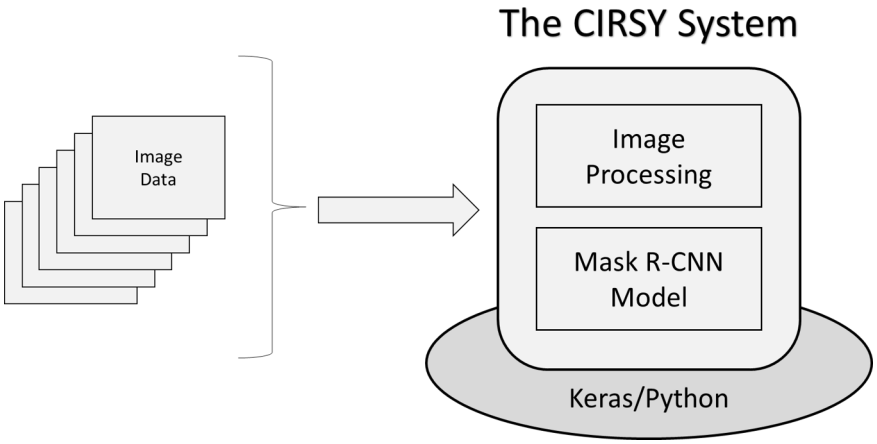


Figure 1.1: The CIRSY Overview

In figure 1.1 above, the overview of the CIRSY system is given. It takes in the Original CCTV image data, applies various image processing techniques to generate training, validation and test data that will be fed into the Mask R-CNN Keras implementation for training and testing.

1.3 Project Aims and Objectives

For this research, the research question is defined as:

- *"To what extent can the mean Average Precision for a Mask R-CNN Model which has been applied on low resolution images to identify small be improved after fine tuning the model?"*

The hypothesis are defined as:

- Null Hypothesis (H0): There will be no statistically significant improvement in the mean average precision of the Mask R-CNN algorithm applied to a low resolution image after the parameters of the algorithm have been fine tuned compared to using the baseline model of the algorithm just of the shelf without fine tuning the parameters.
- Alternate Hypothesis (H1): There will be a statistically significant improvement in the mean average precision of the Mask R-CNN algorithm applied to a low

resolution image after the parameters of the algorithm have been fine tuned compared to using the baseline model of the algorithm just of the shelf without fine tuning the parameters.

The objective of this study is to determine the effect of low resolution images with small objects on the Mask R-CNN algorithm and whether fine tuning the algorithm can lead to significant changes in the mean Average Precision (mAP) of the algorithm.

- Null Hypothesis (H0): There will be no statistically significant improvement in the mean average precision of the Mask R-CNN algorithm applied to a low resolution image after the parameters of the algorithm have been fine tuned compared to using the baseline model of the algorithm just of the shelf without fine tuning the parameters.
- Alternate Hypothesis (H1): There will be a statistically significant improvement in the mean average precision of the Mask R-CNN algorithm applied to a low resolution image after the parameters of the algorithm have been fine tuned compared to using the baseline model of the algorithm just of the shelf without fine tuning the parameters.

In order to conduct this experiment, activities that will be conducted include:

- Generate train and test data from a collection of CCTV footage of chicks in a poultry farm
- Develop the the CIRSY system to apply the Mask R-CNN model to the data and record the observed mAP.
- Fine tune the system parameters and apply the fine tuned model to the data and record the observed mAP.
- Statistically compare observed results from both models.

1.4 Research Methodologies

This project combines Primary and Secondary research, and involves building on data and techniques from existing research which in this case are low resolution images and the Mask R-CNN architecture for image segmentation. The objectives are principally quantitative as the mean average precision will be quantified and the difference measured to test for an improvement from the baseline model. The form is empirical as the null hypothesis will be tested to show there is no statistically significant improvement on the mean average precision on the fine tuned model compared to the baseline model. A deductive reasoning approach will be used as the hypothesis will be tested and the results observed and the hypothesis can then be confirmed based on the observations.

1.5 Project Scope and Limitations

This scope of this research is focused on instance segmentation of small objects from low resolution images using the Mask R-CNN algorithm. To accomplish that, a collection of CCTV images of chicks in a poultry farm will be used. Given the number of chicks in an image, it will not be feasible to annotate all chicks to build a dataset for this research. Hence various chicks will be cropped out and portions of the background will be extracted and used to generate synthetic data for training and testing.

The focus of this research is limited to the Mask R-CNN model with various data preparation techniques and hyperparameter configurations to investigate how to optimize the model performance on the dataset.

The limitation of this project is time as with more time, future work can be done to build on the project such as addition of more classes building of more test datasets and exploring the impact of more hyperparameter configurations.

1.6 Thesis Roadmap

Presented below are the details of the upcoming chapters in this dissertation:

- **"Chapter 2: Computer Vision and Neural Networks: A Roadmap"** This chapter explores the rise of Neural Networks to the state-of-the-art approach to tackle various computer vision problems. A review of techniques which have helped further develop the field of deep learning and a highlight of existing gaps in the research which serve as the roadmap for this project

- **"Chapter 3: The CIRSY system: Development and Experiments"** This chapter describes the development of the CIRSY system as well as the experiment to be conducted in order to explore the research question and fill in the gaps highlighted in Chapter 2. This chapter gives comprehensive details of the research design from data preparation, model building and evaluation.

- **"Chapter 4: The CIRSY System: Evaluation and Discussion"** This chapter reviews results obtained from the experiments carried out from this research and a discussion of the impact of the observed results on the hypothesis of this research.

- **"Chapter 5: Conclusions and Future Work"** This chapter gives an overview of the experiment conducted in this project as well as a discussion of discovered insights during the process. It concludes with a discussion of potential areas of exploration for further development of the work.

Chapter 2

Computer Vision and Neural Networks: A Roadmap

2.1 Introduction

This chapter reviews research in the field of Computer Vision with a focus on Image Segmentation and the techniques applied to its implementation. The first part of the chapter begins with understanding the difference between Image processing and computer vision, then an overview of various computer vision applications and a review of their earlier techniques. The second part of this chapter focuses on the rise of neural networks and their developments to being used in various computer vision applications. Finally, the chapter concludes with a discussion of the gaps not yet deeply explored in image segmentation.

2.2 Computer Vision

2.2.1 Image Processing and Computer Vision

In its simplest form, Image processing can be described as a process which takes an Image and transforms it into an enhanced Image (Vernon, 1991). Some Image processing techniques include Image enhancement, restoration, compression, segmentation,

recognizing and smoothing (Sumithra, Buvana, & Somasundaram, 2015). Various applications where image processing has been used include Character recognition, Face detection, Fingerprint detection (Kiran, Kumar, Prabha, & Kavya, 2015; Saha, Basu, & Nasipuri, 2014) While Computer Vision is the automated extraction of information from images (Solem, 2012). Image processing and Computer Vision are similar as they both take in an Image as their input, they however differ in the output as unlike Image processing which has an output of just an Image, the output of Computer Vision also includes qualitative or quantitative data. An application of Computer Vision could involve a system taking an image, applying Image processing and pattern recognition techniques to the image, and providing inferences about the image.

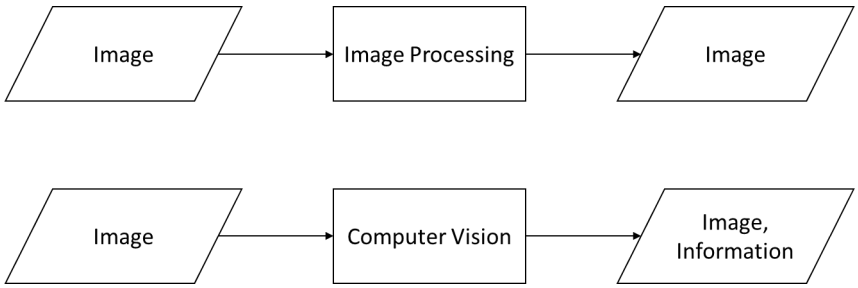


Figure 2.1: Input and Output for Image Processing and Computer Vision

2.2.2 Image Processing Techniques

Some common Image Processing techniques that are used to include: Colour Conversion (Grundland & Dodgson, 2007) or various Image Enhancement and Sharpening approaches (Russo, 2002)i.e. Noise Reduction and Image Normalization (Shi & Govindaraju, 2004) .

Colour conversion in image processing refers to the transformation of colour images to grayscale images to aid perception of images by various applications (Saravanan, 2010). Although there is a lot of information in the colour of images, some tasks won't require the extra dimension of information provided from coloured images and the applications can benefit from the performance enhancement provided by working on grayscale images. Grundland and Dodgson (2007) notes that using grayscale images proves to be more computational efficient with Saravanan (2010) proving that use of

grayscale images can reduce cost.

Image normalization is aimed at enhancing the color contrast of an image and it is also known as contrast stretching (Bazeille, Quidu, & Jaulin, 2006). Shi, Setlur, and Govindaraju (2004) used normalization to enhance the background of a grayscale image in order to improve clarity without changing the original image as much as possible. A major benefit of image normalization is in adjusting the light properties of an image. Shi and Govindaraju (2004) used image normalization on grayscale and coloured images to adjust discolourations in images caused by background light.

Another image processing technique is Noise reduction. Noise refers to noise as frequencies which blur the image or reduce the resolution of an image (Du, 2004). Du (2004) warns that the presence of noise in an image not only degrades the quality of the image but also lessens the accuracy of information extracted from the images. Two popular forms of noise are Gaussian noise and Salt & Pepper noise.

Figure 2.2: Lenna Image with Gaussian Noise

Figure 2.3: Lenna Image with Salt & Pepper Noise

Figure 2.4: Lenna Image Denoised

Kanan and Cottrell (2012) observed how few research completely declared all details of all image processing techniques but proved that all image processing techniques applied should be thoroughly declared as even using different color to grayscale techniques resulted in significant differences in the results when the grayscaled images had been applied to various applications (Kanan & Cottrell, 2012).

2.2.3 Computer Vision Techniques

Three popular applications of Image processing and Computer Vision are; Image Classification, Object Detection and Segmentation.

Image Classification is a technique in which an image is classified based on the content of image (Bosch, Zisserman, & Munoz, 2007). Consider the images below from the Caltech-256 dataset (Bosch et al., 2007) and their classes. Image classification can be used to predict the object in the given image.

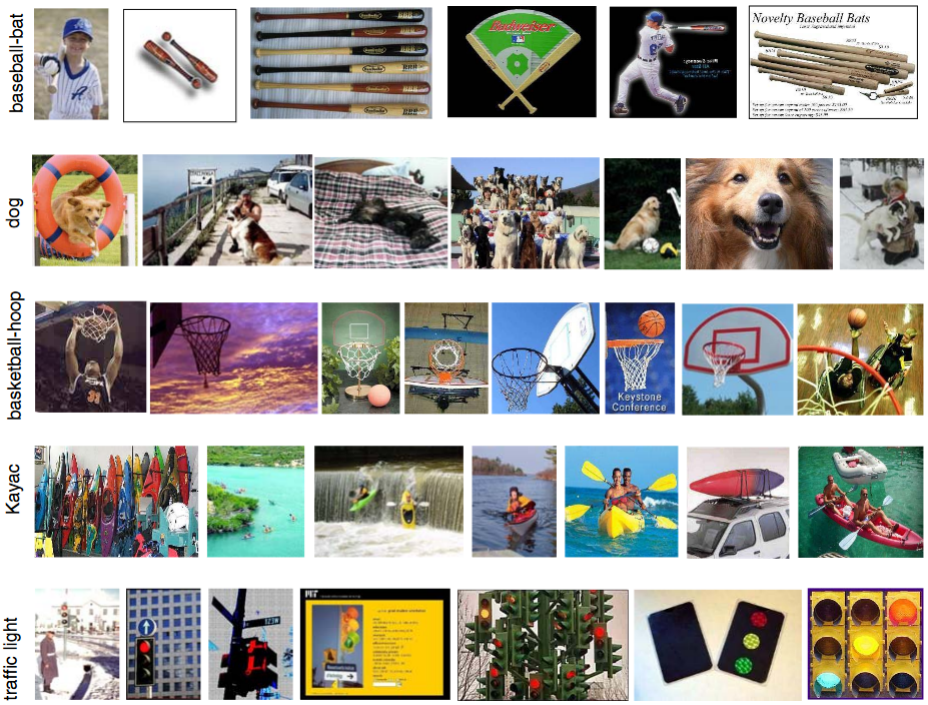


Figure 2.5: Some Images from the Caltech-256 dataset.

The Object detection technique does not just focus on classifying the image based on its content. Rather it is combined with Image Localization, which is used to identify the location of the object in the image. In Object detection, the location and class of the object of interest are predicted.

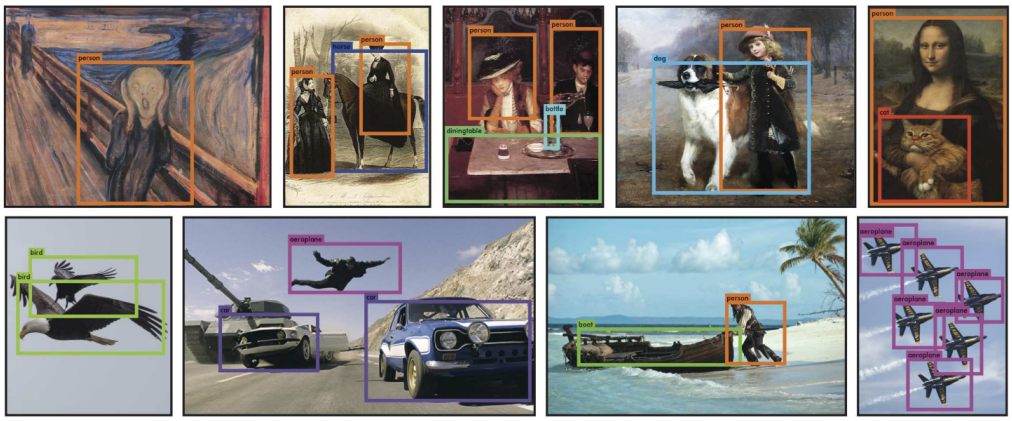


Figure 2.6: Object detection on Picasso and People-Art Datasets.
(Redmon, Divvala, Girshick, & Farhadi, 2016)

Image segmentation is a technique which involves identifying different regions or areas in an image (Dhanachandra et al., 2015; Sridevi & Mala, 2012; Abdel-Maksoud et al., 2015). Images are made up of pixels and image segmentation involves the grouping of these pixels based on how similar they are. Two approaches to Image Segmentation are Semantic Segmentation and Instance Segmentation.

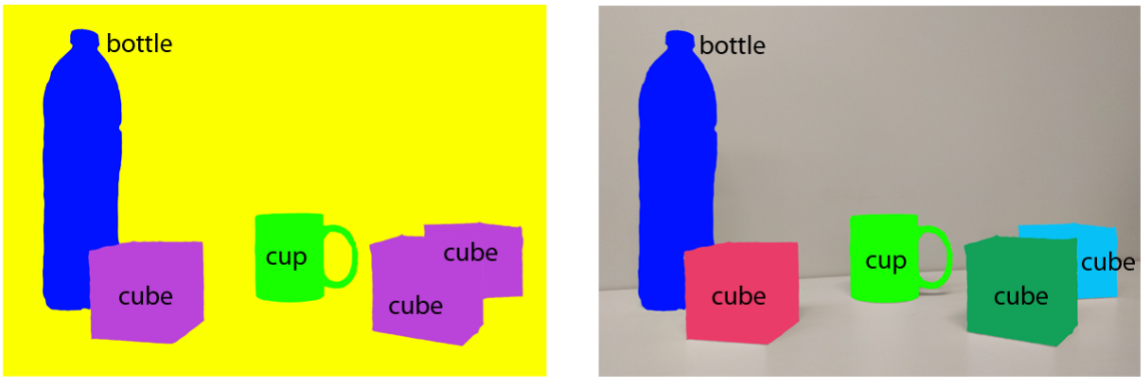


Figure 2.7: Semantic Segmentation (left) and Instance Segmentation (right).
(Garcia-Garcia, Orts-Escolano, Oprea, Villena-Martinez, & Garcia-Rodriguez, 2017)

In Semantic Segmentation, the pixels of objects that represent the same class are not differentiated and given the same mask color. While in Instance Segmentation, different objects for the same class are separately identified and given a different mask color where possible.

Instance Segmentation can be seen as combination of Object Detection and Image Segmentation. Unlike Object detection which generates only a boundary box over the object of interest in the image, Instance Segmentation also generates a mask over the region covered by the object and segments it from the rest of the image.

Image Segmentation is important as it can be used to examine the contents of an image, divide it into regions and allow focus on regions of importance and ignore regions that do not provide useful information.

2.3 Neural Networks

2.3.1 Perceptrons

Rosenblatt (1958) developed the Perceptron as a mathematical model based on the work of McCulloch and Pitts (1943) which aimed to replicate how the neurons in the human brain operate. Perceptrons work by taking a series of binary inputs and producing a single binary output.

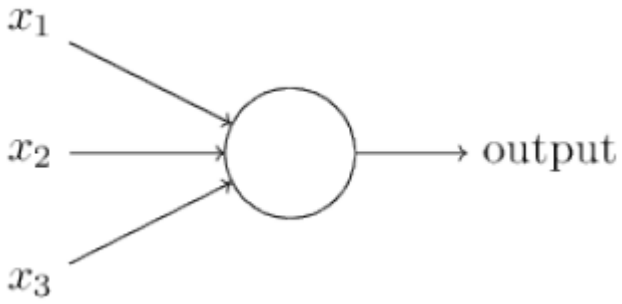


Figure 2.8: A Perceptron.
(Nielsen, 2015)

Nielsen (2015) stated that Rosenblatt introduced the notion of weights (w) as real numbers to express the importance of a specific input to the output, and explains that the output is determined by the weighted sum of the inputs being less than or greater

than a specified threshold value. It can be expressed as follows:

$$output = \begin{cases} 0, & \text{if } \sum_j w_j x_j \leq threshold. \\ 1, & \text{if } \sum_j w_j x_j > threshold. \end{cases} \tag{2.1}$$

Backpropagation

Nievergelt (1969) noted that two key researchers in the neural network field, Marvin Minksy and Seymour Papert, noted in 1969 that there are limitations of single neuron models ("perceptrons") and they showed evidence of the need for multi-layer perceptrons, with each layer performing well defined functions, for complex operations such as feature extraction.

This led to a lull in researching this topic until Rumelhart, Hinton, Williams, et al. (1988) proposed back-propagating errors in which based on the error of the output layer, the weights of the units in the hidden layers are adjusted to account for the error. The figure below visualises the idea of backpropagation.

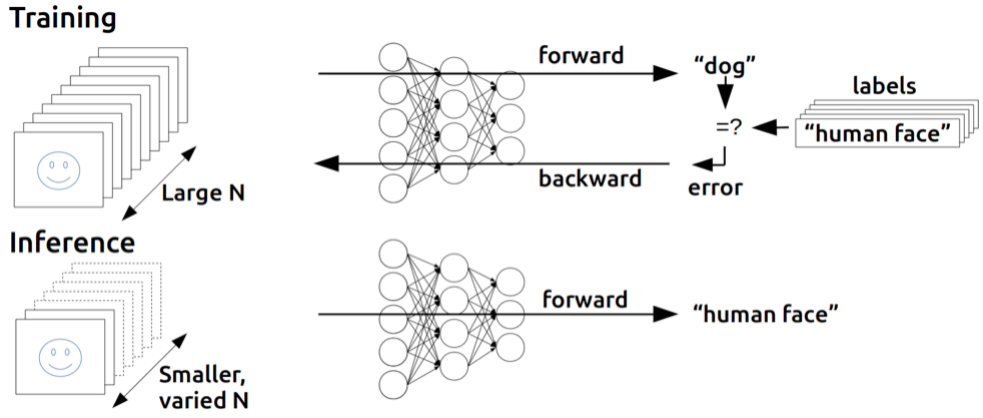


Figure 2.9: The concept of backpropagation.

(Inference, 2015)

2.3.2 Convolutional Neural Networks (CNN)

Given the breakthrough brought on by backpropagation, LeCun et al. (1989) presented a groundbreaking application of it on the recognition of handwritten zip code digits.

In this work, rather than feed in feature vectors, the network was fed directly with the images which showed that backpropagation networks could deal with large amounts of data (LeCun et al., 1989). The network used a weight sharing technique, which is now known as convolutions. Feature extraction was done in the first hidden layer which composed of planes known as feature maps. The second hidden layer also had feature maps, it took in the features extracted from the first hidden layer. The third layer had 30 units which were fully connected to the second hidden layer while the output layer contained 10 units, one for each digit from 0 to 9.

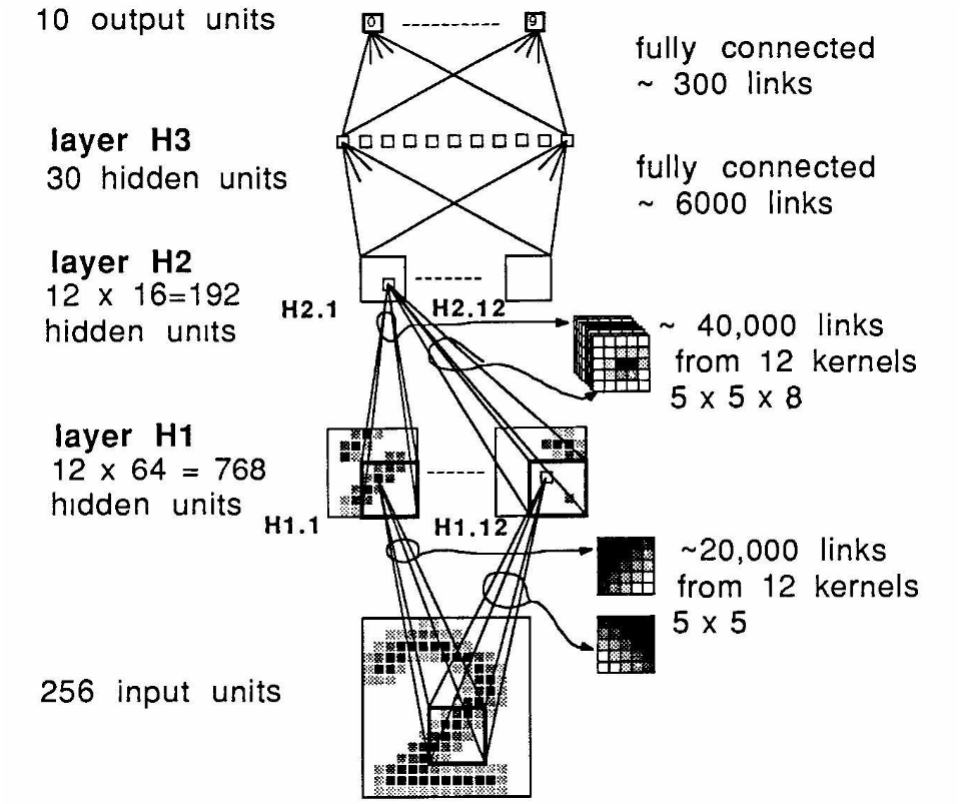


Figure 2.10: handwritten zip code digits backpropagation network design (LeCun et al., 1989)

The designed developed by LeCun et al. is very similar to current CNN design, however there was little popularity in Neural Nets and CNN at the time.

2.3.3 Deep Learning

in 2012, Krizhevsky, Sutskever, and Hinton (2012) noted that although the CNN architecture is attractive and efficient, it has been computationally expensive to implement in large scale high resolution images, however, with the current GPUs it is possible to train such large scale image datasets. With the proposal of AlexNet (Krizhevsky et al., 2012), Krizhevsky et al. showed the ability and ease to train CNNs with millions of high resolution images and improve the state-of-the-art in model accuracy (Krizhevsky et al., 2012).

Despite the many advances being made after the success of Krizhevsky et al., He and Sun (2014) noted that very deep networks were not necessarily better as accuracy stagnated and even reduced in very deep attempts. He, Zhang, Ren, and Sun (2015) went on to alleviate the problem with the proposal of ResNet (He et al., 2015) with the addition of the Residual block.

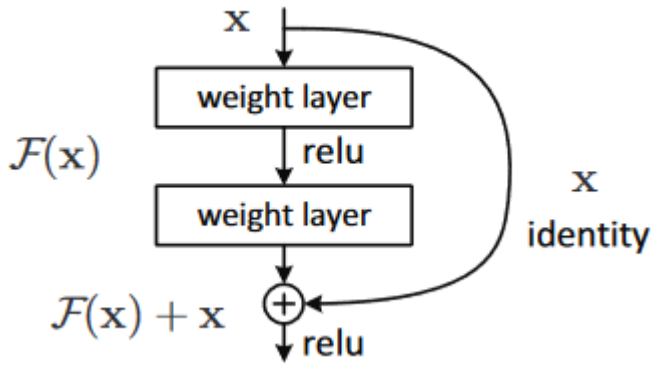


Figure 2.11: Residual learning: a building block
(He et al., 2015)

ResNet enjoyed greater accuracy than state-of-the-art approaches in the ImageNet dataset (He et al., 2015).

2.4 Traditional Approaches to Image Segmentation

Earlier approaches to image segmentation involved threshold, edge and region-based methods (Pal & Pal, 1993; Kaganami & Beiji, 2009; Senthilkumaran & Rajesh, 2009).

2.4.1 Threshold-Based

Thresholding is a simple image segmentation technique which uses a threshold value (T) in order to create a binary image from a grayscale image (Bhargavi & Jyothi, 2014). Image binarization separates the pixel values into two, black for the background and white for the foreground (Senthilkumaran & Vaithegi, 2016).

Senthilkumaran and Vaithegi (2016) states the threshold technique can be expressed as:

$$T = T[x, y, p(x, y), f(x, y)] \tag{2.2}$$

Where T is the threshold value. x, y are the coordinates of the threshold value point. p(x,y) is local property and f(x,y) is the gray level.

The threshold image is expressed as:

$$g(x, y) = \begin{cases} 1, & \text{if } f(x, y) \geq T. \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

2.4.2 Region-Based

The region-based approach groups pixels with similar values and separates pixels with different values as separate classes. In an image, regions are a group of connected pixels with similar properties (M. Kaur & Goyal, 2015). M. Kaur and Goyal (2015) states that in region-based approaches, a pixel is assigned to an object or region. Two main techniques to region-based segmentation are region growing techniques and region splitting and merging (D. Kaur & Kaur, 2014). Both techniques represent different opposite approaches to each other. Where region growing starts with an initial set of pixels and neighbouring pixels are added based on their similarity (M. Kaur & Goyal,

2015; D. Kaur & Kaur, 2014). while region splitting and merging divides starts with the whole image as a single region and divides it into sub regions.

2.4.3 Edge-Based

In the edge-based approach, an edge filter is used to classify a pixel as an edge or not an edge. The pixels not separated by edges are assigned to the same class. Edge-based techniques are divided into two groups; First and Second order derivatives (Sharifi, Fathy, & Mahmoudi, 2002; Sridevi & Mala, 2012).

First order derivative techniques include Gradient edge detectors such as Prewitt and Sobel (Sridevi & Mala, 2012; Sharifi et al., 2002; Saini & Arora, 2014), which were popular for their simplicity, but were however sensitive to noise (Sharifi et al., 2002).

Second order derivative techniques such as Zero crossings (Sharifi et al., 2002; Saini & Arora, 2014) and the Laplacian of Gaussian (LoG) (Sridevi & Mala, 2012; Sharifi et al., 2002). The LoG technique was invented by Marr and Hildreth (Marr & Hildreth, 1980) where a Gaussian is used in blurring the images and a Laplacian is used in the enhancing of the edges.

M. Kaur and Goyal (2015) noted that compared to edge based segmentation techniques, region-based techniques are more handle noisy images better.

2.4.4 Clustering

Clustering techniques such as K-means clustering have also been used in image segmentation (Abdel-Maksoud et al., 2015; Tatiraju & Mehta, 2008; Ray & Turi, 1999; Ng, Ong, Foong, Goh, & Nowinski, 2006), where the images are converted to 2-dimensional arrays and the cluster algorithm such as K-means is applied to assign pixels of the image to a cluster. However studies show that the algorithm is not suitable for large image sizes and the Euclidean distance metric is not good for segmentation (Tatiraju & Mehta, 2008).

2.5 Traditional Approaches to Object Detection

Earlier techniques to object detection involved a combination of using techniques such as Haar-like features, Scale-invariant feature transform (SIFT) or Histogram of oriented gradients (HOG) features to extract features before feeding them into a classifier like a Support Vector Machine (SVM) to perform object detection tasks (Zafeiriou, Zhang, & Zhang, 2015).

2.5.1 Viola and Jones Object Detection Framework

Viola and Jones (2001) proposed this framework, which used Haar features extracted from the eye and nose regions of the face and feeding to an AdaBoost classifier to perform face detection tasks. This was a robust and real time framework which excelled for its efficiency and ease of use (Jones & Viola, 2003).

Haar features are those that hold key information about the object to be detected. In facial recognition, Haar features are used to detect key facial regions like eyes and nose. The three main types of Haar features are edge features, line features and centre surrounding features (Guennouni, Ahaitouf, & Mansouri, 2015).

2.5.2 Histogram of Oriented Gradients

Dalal and Triggs (2005) proposed a framework which used HOG for feature extraction with an SVM classifier for detection of people in an image. This proposal was shown to outperform existing feature extraction techniques (Dalal & Triggs, 2005).

HOG is a feature descriptor which not only extracts the edge features, it also gets the direction and orientation of these features. This is done for different regions in the image and a Histogram of the gradients is then constructed. This feature map is then fed into classifiers such as a linear support vector machine.

2.6 Deep Learning in Computer Vision

Unlike the earlier traditional approaches which first required separate feature extraction/definition techniques before feeding these features into a machine learning classifier, deep learning techniques did not need separate feature definition stages before classification. Constant developments in CNNs and Deep learning techniques led to continuous improvement in deep learning approaches to object detection.

2.6.1 Region Proposals

Girshick, Donahue, Darrell, and Malik (2014) proposed a novel approach to object detection called Regions with CNN (R-CNN). The algorithm uses selective search to extract 2000 region proposals. This region proposals are regions with a high probability of having an object (Girshick et al., 2014). The extracted regions of interest (RoI) are then warped and fed into a CNN with the AlexNet (Krizhevsky et al., 2012) architecture before classification of the extracted feature vectors using an SVM (Girshick et al., 2014).

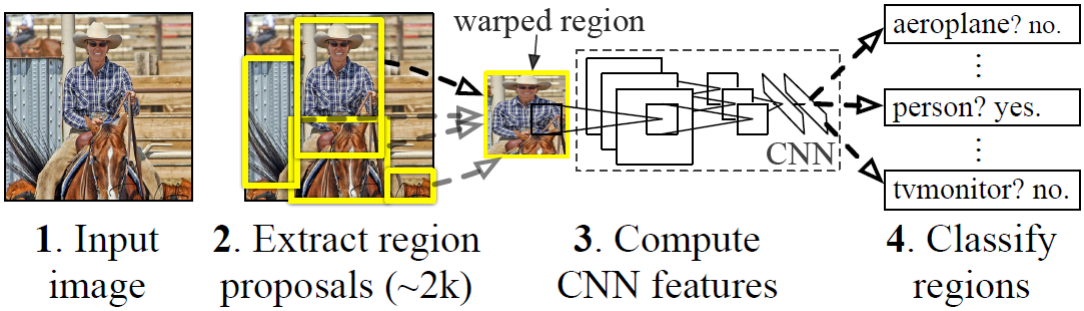


Figure 2.12: R-CNN Work Flow
(Girshick et al., 2014)

Unfortunately, object detection was slow in R-CNN (Girshick et al., 2014; Girshick, 2015). Girshick (2015) was proposed to improve on the drawbacks and accuracy of R-CNN. This was achieved by first processing the image producing a convolutional (conv) feature map before obtaining the region proposals (Girshick, 2015).

Although Fast R-CNN (Girshick, 2015) have drastically reduced the computational

cost and improved the speed of region-based CNNs,the region proposal was to current bottleneck in the system (Ren, He, Girshick, & Sun, 2015). Ren et al. (2015) proposed Faster R-CNN which combines the novel Region Proposal Networks (RPNs) with the convolutional layer. This resulted in a deep fully convolutional network used to propose regions and a Fast R-CNN detector to use the proposed regions (Ren et al., 2015).

Faster R-CNN was extended when He et al. (2017) proposed Mask R-CNN which added a branch to predict segmentation masks over each RoI. This was done by applying an FCN to each RoI (He et al., 2017). However, due to the Masks being slightly misaligned, the RoIPool layer was replaced with the RoIAlign to solve the problem. The generated masks are then combined with the object detection from the Faster R-CNN to generate instance segmentation.

2.7 Transfer Learning

Transfer learning is a technique that allows a model to be reused on a different task that it wasn't initially or directly trained for. This technique has been known to allow for improved training time and a reduction in the time cost of data gathering for training (Pan & Yang, 2010). Deep learning focus on learning abstract representations allows it to be well suited to transfer learning (Bengio, 2012).

2.8 Gaps in Research

Research on the impact of low resolution images on image segmentation algorithms has not been sufficiently explored. Although the image reconstruction technique Super Resolution has been used to improve the quality of low resolution images (Sarmadi & Shamsa, 2016; Zhang & He, 2014), however, given the advances in deep learning and the image segmentation algorithms most specifically the image instance segmentation algorithms, not enough research has been done to see how this algorithm performs when identifying objects in low resolution images without extensively modifying the resolution of the image.

2.9 Summary

Developments in Neural Networks and Deep Learning saw the emergence of Convolutional Neural Networks (CNN) and more Deep neural nets which were relatively easier to train become the go-to solutions for computer vision tasks. These developments led to shifts from threshold, region, edge based and clustering techniques in image segmentation problems to Neural net based approaches. A major advantage of CNN over traditional image processing approaches is the hierarchical feature representation (Zhao, Zheng, tao Xu, & Wu, 2018) and its efficiency has seen it used in image reconstruction (Zhang & He, 2014), segmentation (E.-U. Lin, McLaughlin, Alshehri, Ezekiel, & Farag, 2014), classification (H. Lin, 2008) and object detection (Chen, Lu, & Fan, 2017).

This research aims to explore instance level segmentation with the Mask R-CNN algorithm on poorly lit images where the objects of interest are difficult to differentiate from the background. The use of transfer learning will be used for this research as the experiments conducted will build on applying the weights from the model trained on the MS COCO dataset. Based on the review of the literature, this act should allow for an increase in training time and aid in good performing models without extensive modifications and training.

The next chapter describes the design of the experiment to be carried out to explore the proposed research gap.

Chapter 3

The CIRSY System: Design and Development

3.1 Introduction

This research aims to develop the CIRSY system to use the Mask R-CNN algorithm to perform instance segmentation on small, low resolution objects. In this chapter, the structure of the experiment to be performed in order to test the proposed hypothesis will be described.

The methodology used in this research is the CRISP-DM (CRoss-Industry Standard Process for Data Mining) methodology. The CRISP-DM methodology is a framework used when working on data mining projects (Wirth & Hipp, 2000). It has six different phases which in its lifecycle, namely Business Understanding, Data understanding, Data preparation, Modeling, Evaluation and Deployment.

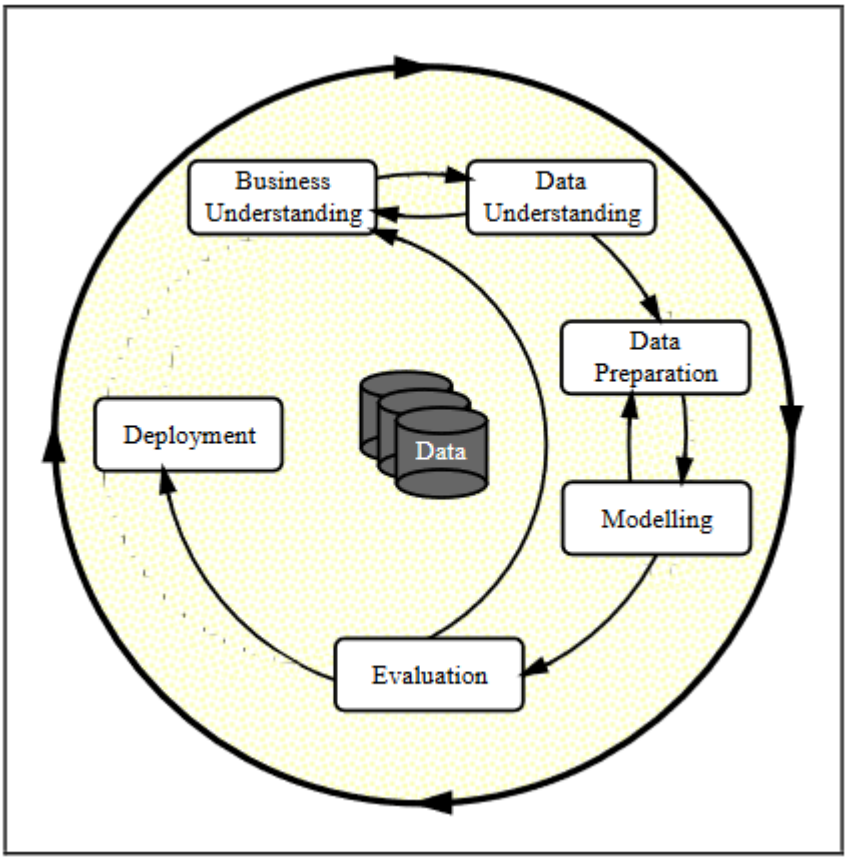


Figure 3.1: Crisp DM Lifecycle
(Wirth & Hipp, 2000)

From 3.1 it shows that even though each phase of the CRISP-DM lifecycle represents a different milestone in a data mining project, any stage may be revisited at any time due to new insights gained or project goal changes. This flexibility has made the CRISP-DM methodology the most popular today as it is widely used in majority of data mining research projects.

3.2 Business Understanding

This research focuses on examining the effect of Mask R-CNN on detecting small low resolution objects in images and if the performance can be improved. For this experiment, an increase in the mAP will be used to determine if the model has improved or worsened in performance. The hypothesis for this research can be stated as:

- Null Hypothesis (H0): There will be no statistically significant improvement in the mean average precision of the Mask R-CNN algorithm applied to a low resolution image after the parameters of the algorithm have been fine tuned compared to using the baseline model of the algorithm just of the shelf without fine tuning the parameters.
- Alternate Hypothesis (H1): There will be a statistically significant improvement in the mean average precision of the Mask R-CNN algorithm applied to a low resolution image after the parameters of the algorithm have been fine tuned compared to using the baseline model of the algorithm just of the shelf without fine tuning the parameters.

3.3 Data Understanding

3.3.1 Original Dataset

A collection of images from CCTV footage of free range chicks in a farm has been selected for use in this research. This is because the images presents challenging conditions which meet the requirements for the experiment. It has small objects to detect, it is difficult to differentiate the background and some objects and the images are generally poorly lit and in a low resolution.

Figure 3.2: Example of an Image from the dataset

The CCTV images used in this research were obtained from a collection from Dr. Robert Ross. The collection contained a total of 186 images with an average size of 1.27MB each, all with dimensions of 2560 X 1440 pixels, containing no less 200+ instances of chickens on each image.

3.3.2 Generated Dataset

In order to train the Mask R-CNN on images similar to that in figure 3.2, each object (chickens) would have to be annotated. Hence for feasibility reasons, synthetic images which have the objects composed on the backgrounds will be used for training and testing.

The images synthetically generated were an average of 50kb each, with no more than three chick instances in an image and at least one instance in any image. All dimensions were set at 512 X 512 pixels to save memory.

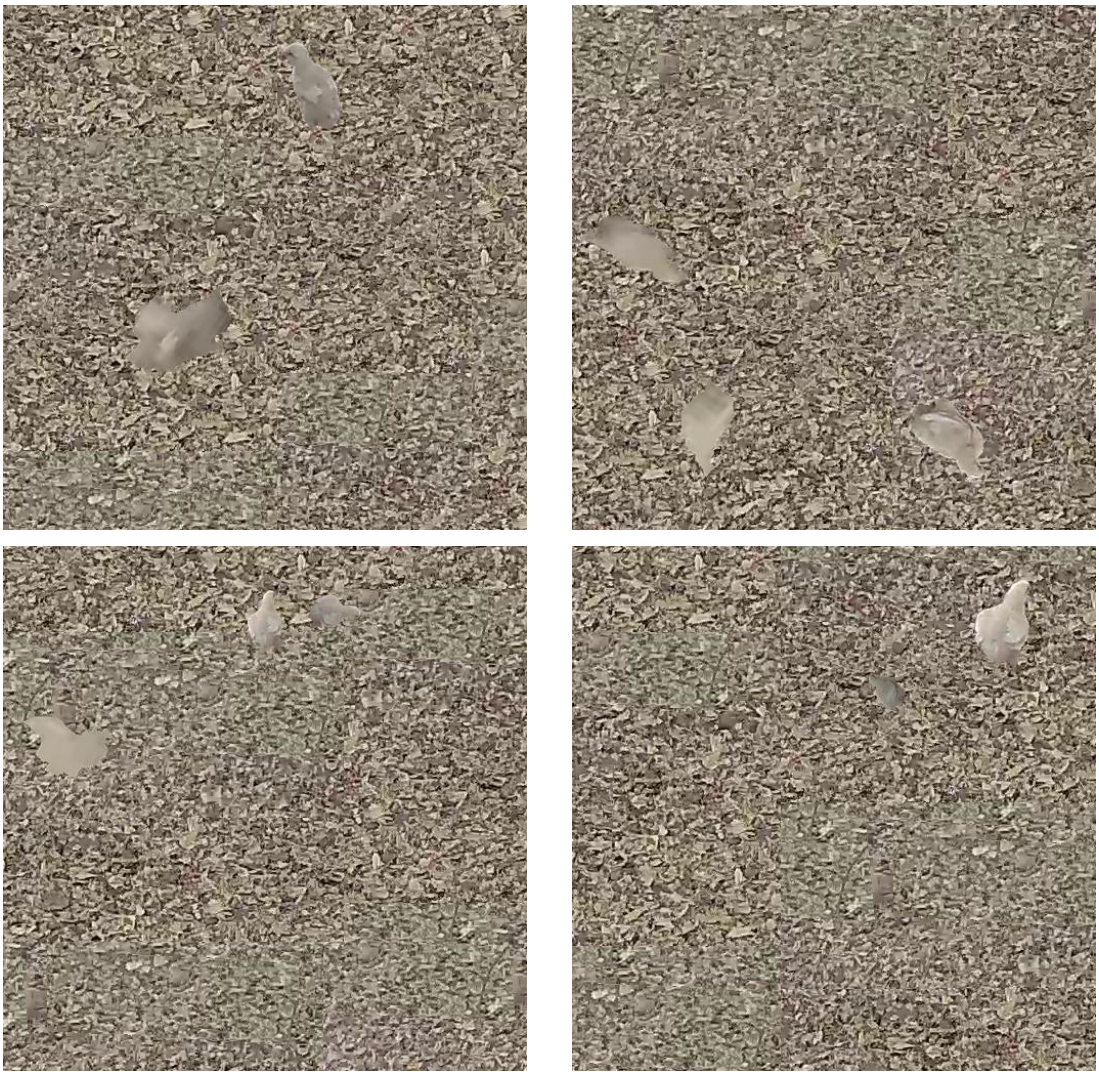


Figure 3.3: Example of Composed Images used for Training and Testing

The images shown in figure 3.3 were created by composing the foreground (i.e various instances of chickens cropped from the original image) with the background from the original dataset. Given that the background is the same, only one background image has been used for the image composition. This method allowed for the annotation of 10,000 training images, 1,000 validation images and 2,500 test images with masks over each instance of interest.

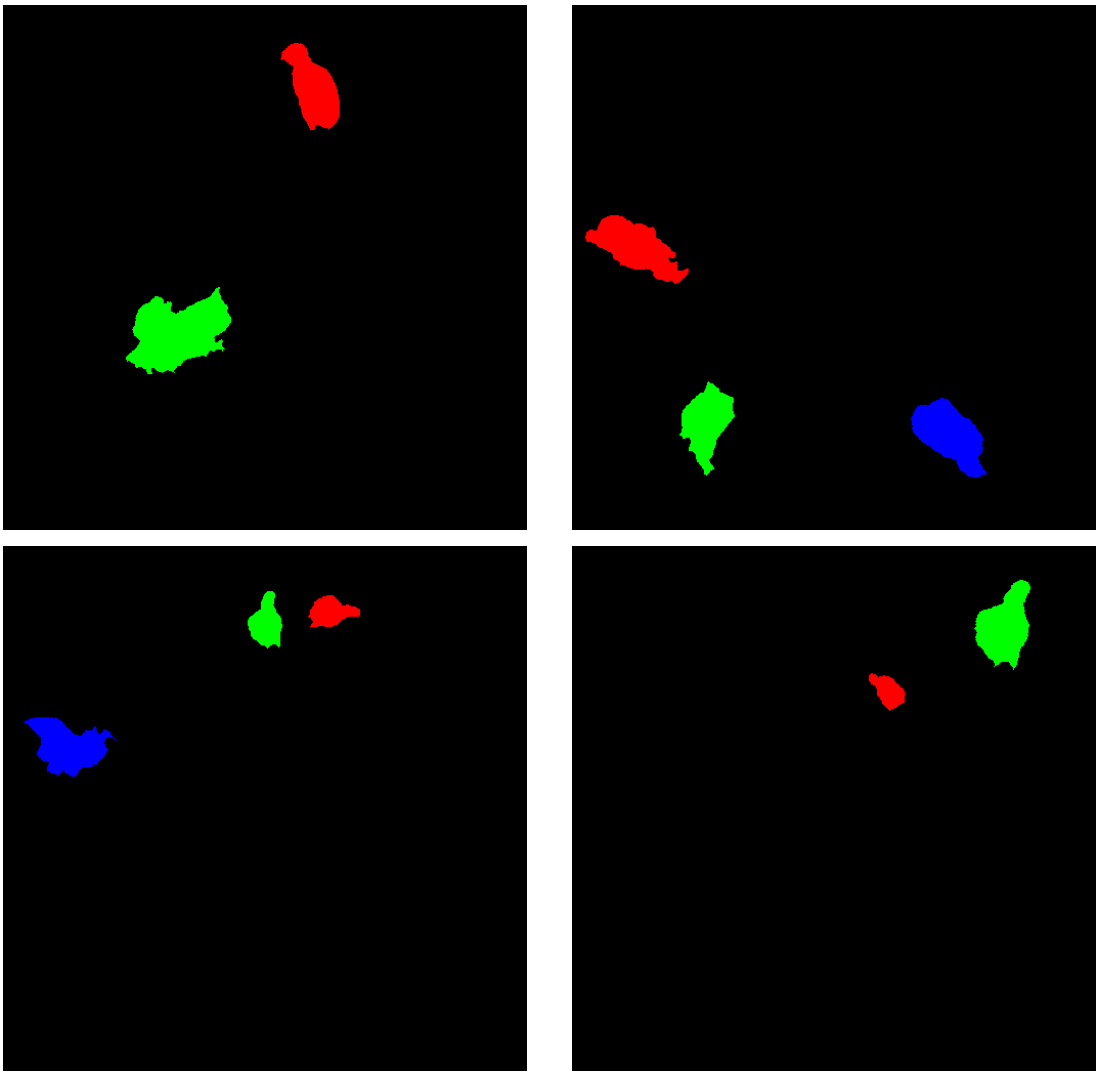


Figure 3.4: Example of Masks for generated images

3.4 Model Understanding

To implement the experiment, the CIRSY system was developed on a Keras (Python-based Neural Network library) implementation of the Mask R-CNN algorithm (Abdulla, 2017). The Mask R-CNN uses the ResNet 101 architecture as the backbone model to extract features from the image for input to the RPN. The RPN takes the extracted features (feature maps) and predicts if the region may or may not have an object. A pooling layer is applied and the regions are converted to the same shape and passed through an FCN in order to predict bounding boxes and the class labels. While they

are fed into the mask branch which predicts a mask over the predicted object.

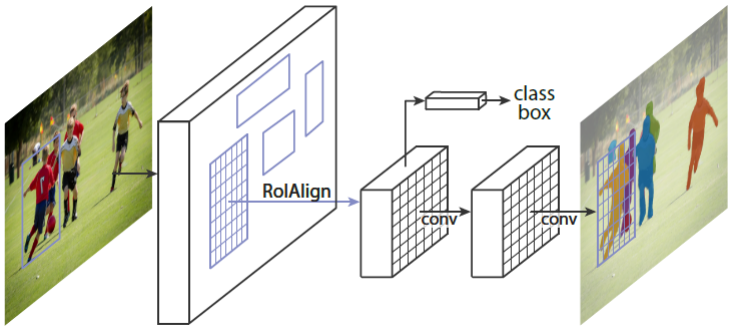


Figure 3.5: Mask R-CNN Architecture

In order to fine tune the model, various hyperparameters will be experimented on to observe the models performance, such as:

- The Learning Rate: This refers to how fast the model actually learns. If it's too fast, the model may not get the optimal results, but if it's too slow the model may take much longer to get the optimal results.

- The Epochs: This refers to the number of times the model processes the training set. With the aim of the model to reduce error by reaching a global mimima, the number of epochs can affect the models chance of achieving a true global minima and not just a local minima.

- The Steps per Epoch: This refers to the number of training steps it takes to complete processing the dataset. It is heavily influenced by the batch size which in this case, is the number of images processed at a time.

$$\text{Steps per Epoch} = \frac{\text{Dataset Count}}{\text{Batch Size}} \tag{3.1}$$

- The Number of Region Proposals: This refers to the number of regions in an image which can contain objects of interest.

- The Backbone: This refers to the separate layer which performs feature extraction and feature mapping for the model. This research uses the earlier discussed ResNet architecture as the backbone.

3.5 Evaluation

This research recognizes the mean average precision (mAP) as a recognized performance indicator in instance segmentation and it will be used to evaluate model performance. For tasks that involve object detection, the mAP requires the Intersection over Union (IoU). In Object detection tasks, the IoU is the ratio of the area of overlap (i.e. area where the prediction and ground truth intersect) and area of union (i.e. area where the prediction and ground truth cover) of the predicted and ground truth values and is used in assessing if the prediction is a True Positive, False Positive or False Negative.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{3.2}$$

The IoU is a threshold used to determine if a prediction is a True Positive, False Positive or False Negative. Li, Qi, Dai, Ji, and Wei (2017) stated the mAP@[0.5] is the traditional evaluation metric in instance segmentation tasks. where 0.5 is the IoU and if greater than 0.5 a prediction is considered a True Positive and a False Positive if less than 0.5.

3.5.1 Hypothesis Testing

In order to test the hypothesis, the mAP for the baseline model and selected fine tuned model will be computed for each of these test sets. A t-test of difference will be conducted between the two models for the mAP and the null hypothesis will be rejected or failed to be rejected based on the p value, with an alpha value set at 0.05.

3.6 Experiment Design, Software & Environment

3.6.1 Software & Environment

For data preparation, the programming language used for this experiment was Python. It was chosen due to its ease of use, versatility and availability of a wide array of libraries. GNU Image Manipulation Program (GIMP) was used for image editing as

it provides a powerful image editing tool for free. All data preparation was performed on a Windows 10 Home Laptop with an 8GB RAM and 320GB SSD.

For model training and testing, the Google Colaboratory known also known as Google Colab or Colab was used. Colab offers a free Jupyter notebook environment with free access to 1 Tesla K80 GPU having 2496 CUDA cores, 12GB(11.439GB Usable) GDDR5 VRAM and 320GB cloud storage.

3.6.2 Experiment Design: Training Datasets

In order to replicate similar surroundings to the chicks in the farm, four different backgrounds will be used with the foreground chick images to allow the model better differentiate between background objects and the chicks. For this experiment, four will be enough due to the fact that the surroundings of the chicks stays constant and has little variability.

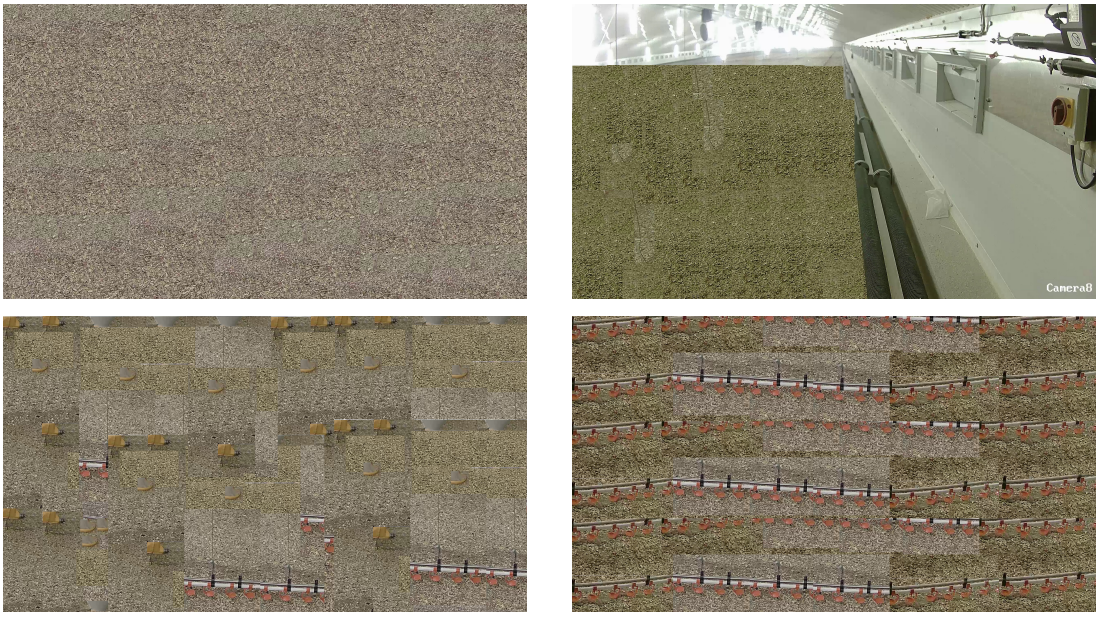


Figure 3.6: Background Images used for Image Composition

Three different versions of the training data will be generated in order to explore the impact of different factors applied on the training data to the model.

Version 1: Training Data no Transformations

The first version of the training dataset simply composes different combinations of the foreground images (cropped out chicks) to the background without any further augmentations applied to the images.

Version 2: Training Data with Transformations

the second version applies 3 different transformations (rotations, scaling and brightness) to the foreground images.

- Each foreground is randomly rotated at an angle between 0 and 356 degrees.
- Each foreground is then randomly scaled by a factor between 0.5 and 1.
- The brightness of each foreground is then randomly enhanced by a factor between 0.8 and 1.2.

```
def transform_chick(self, chick):

    # Open cropped chick Image
    chick_image = Image.open(chick)


    # Rotations
    angle = random.randint(0, 359)
    chick_image = chick_image.rotate(angle_degrees, resample=Image.BICUBIC, expand=True)

    # Scaling
    scale = np.random.uniform(0.5, 1.1) # Pick something between .5 and 1
    new_size = (int(chick_image.size[0] * scale), int(chick_image.size[1] * scale))
    chick_image = chick_image.resize(new_size, resample=Image.BICUBIC)

    # Brightness
    brightness = np.random.uniform(0.8, 1.2)# Pick something between .0 and 1.2
    enhancer = ImageEnhance.Brightness(chick_image)
    chick_image = enhancer.enhance(brightness)

    return chick_image
```

Figure 3.7: Image Transformation Function Snippet

Version 3: Training Data with Transformations & Noise

The third version retains the same transformations as version 2, however, salt-and-pepper noise has been added to each image. One of the reasons for low quality images is that they have been affected by noise (Fu, Zhao, Song, Li, & Wang, 2019). Fu et al. (2019) state salt-and-pepper noise as a common form of noise and Azzeh, Zahran,

and Alqadi (2018) defined salt-and-pepper noise as sparsely occurring white and black pixels on an image (Azzeh et al., 2018).

```
def saltpep_noise(self, image, noise_prob):
    '''
    noise_prob: Probability of the noise
    '''
    img_output = np.zeros(image.shape,np.uint8)
    noise_thresh = 1 - noise_prob
    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            rdn = random.random()
            if rdn < noise_prob:
                img_output[i][j] = 0
            elif rdn > noise_thresh:
                img_output[i][j] = 255
            else:
                img_output[i][j] = image[i][j]
    return img_output
```

Figure 3.8: Salt & Pepper Noise Function Snippet

Figure 3.9: Example of Salt-and-Pepper noise applied on an image (right).

The dimensions of the background images which are all 2560 X 1440. However, a size of 512 X 512 will be randomly extracted from each image when composing for each of the 10k training data.

3.6.3 Experiment Design: Test Datasets

For each version of the training datasets, a test dataset with the same characteristics will be generated. It is however expected that because these test sets were similarly created, the model will have a high performance in relation to the mAP. Hence another test set will be generated from the original collection of CCTV footage not used during the generation of the training dataset.

From various images in the CCTV collection, different areas which feature single chicks or a group of chicks will be cropped and annotated. For this experiment, 5 different test datasets each with 12 different images all holding a group of chicks or a single chick will be used to measure the performance of the model trained on each version of the training datasets.

Figure 3.10: Sample Images that will be annotated and used to test models.

3.6.4 Experiment Design: Models and Evaluation

Other Image Segmentation Techniques Overview

This experiment will explore how different image segmentation algorithms effect the images. Given that the quality and content of the images are pushing the boundaries of the limitations of these traditional image segmentation algorithms such as little difference in contrast between the objects (chicks) and the background and too many edges due to the large number of objects in the images.

Vanilla Mask R-CNN Implementation

The vanilla Mask R-CNN implementation using just the weights trained on the MS-COCO datasets will be used to observe how it is affected by the data. However, even though birds are represented in the dataset, it is expected that due to the difference in quality and content this implementation will have difficulties performing on the prepared test datasets. Nevertheless, the impact of the difference in the datasets is going to be observed.

```
# Create model in Inference mode as no training required
model = modellib.MaskRCNN(mode="inference", config=config,
                          model_dir=MODEL_DIR)

#Load model weights
model.load_weights(COCO_MODEL_PATH, by_name=True)

# COCO Class names
class_names = ['BG', 'person', 'bicycle', 'car', 'motorcycle', 'airplane',
               'bus', 'train', 'truck', 'boat', 'traffic light',
               'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird',
               'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear',
               'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie',
               'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball',
               'kite', 'baseball bat', 'baseball glove', 'skateboard',
               'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup',
               'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
               'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',
               'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed',
               'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote',
               'keyboard', 'cell phone', 'microwave', 'oven', 'toaster',
               'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors',
               'teddy bear', 'hair drier', 'toothbrush']
```

Figure 3.11: Default Mask R-CNN Implementation

Baseline Model Implementation

For each version of the training datasets, the default parameters of the Mask R-CNN model will be used to create the baseline model for each training dataset version. The performance of the baseline model on each of the 5 test datasets will be compared for each version of the model trained on each different training dataset. This will allow

this experiment explore the impact of the transformations and addition of noise on the model performance.

```
# Create model in training mode
model = modellib.MaskRCNN(mode="training", config=config,
                          model_dir=MODEL_DIR)
# Load COCO Weights for Training
model.load_weights(COCO_MODEL_PATH, by_name=True,
                   exclude=["mrcnn_class_logits", "mrcnn_bbox_fc",
                            "mrcnn_bbox", "mrcnn_mask"])
#Train Model with LR = 0.001
model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE,
            epochs=12,
            layers='heads')
```

Figure 3.12: Base Models: Loading Mask R-CNN Implementation Weights

Mask R-CNN allows for various configurations to be made to save memory and improve training time. One of such configurations is the layers that can be trained. based on the Resnet architecture, all layers can be trained or layers from various stages can be trained. For the baseline model for this research, the heads layers which is the output layer will be trained while all other layers will remain frozen. This will allow for the loading of the pre-trained MS-COCO weights and the retraining of just the output layers.

Model Fine tuning

To fine tune the model, rather than train just the output layers of the models, all layers will be trained still using the default hyper parameter settings.

Based on the top performing models from the baseline and first fine tuned versions of the model, a final model will be trained making various changes to the hyper parameters to match the data requirements. In 3.13 and 3.14, the parameters and model training are shown in the snippet .

```
class ChickenSynthConfig(Config):
    NAME = "chicken_synthetic_comb_coco"
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
    # Number of classes (including background)
    NUM_CLASSES = 1 + 1  # background + 1 chick image type
    # Whether to use image augmentation in training mode
    AUGMENT = True
    # Whether to use image scaling and rotations in training mode
    SCALE = True
    # Optimizer, default is 'SGD'
    OPTIMIZER = 'ADAM'
    # How many anchors per image to use for RPN training
    RPN_TRAIN_ANCHORS_PER_IMAGE = 500  #
    # Number of ROIs per image to feed to classifier/mask heads
    TRAIN_ROIS_PER_IMAGE = 500
    # Using small anchors because of chick size
    RPN_ANCHOR_SCALES = (8, 16, 32, 64, 128)
    #Steps Per Epoch
    STEPS_PER_EPOCH = 10000
    #Validation steps
    VALIDATION_STEPS = 1000
    #Backbone
    BACKBONE = 'resnet101'

config = ChickenSynthConfig()
```

Figure 3.13: Fine Tuned Parameters

```
#Train all Layers with LR = 0.001
model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE,
            epochs=4,
            layers='all')
#Fine Tune: Train all Layers witt LR = 0.0001 for next 4 epochs
model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE / 10,
            epochs=8,
            layers='all')
#Fine Tune: Train all Layers witt LR = 0.00001 for last 4 epochs
model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE / 10,
            epochs=12,
            layers='all')
```

```
# Which weights to start with?
init_with = "coco"  # imagenet, coco, or last
```

Figure 3.14: Fine Tuned Model Training

Model Evaluation

To evaluate the performance of each model compared, The mAP of each model from each of the 5 test datasets will be computed and compared to each other. A One Way ANOVA test will be used to determine if the difference between the mAP for each of the three baseline models trained with the three different training datasets is statistically significant (A one-way ANOVA (or one-way ANalysis Of VAriance) is a stastical technique that can be used to compare means of two or more samples using the F distribution). This test will also be conducted to test the difference between the three fine-tuned models.

A paired t-test will be used to determine if the difference between a baseline model and the fine-tuned model for the same dataset are statistically significant (A paired t-test (or dependent sample t-test) is a statistical technique that can be used to determine whether the mean difference between two sets of observations is zero). This will be repeated between the best performing baseline model and the final fine-tuned model which has had its hyper parameters changed.

3.7 Summary

This chapter gave an overview of the design and development of the CIRSY system for use in this experiment to be conducted to test the hypothesis. The dataset was

discussed in detail, as well as the procedure to create the generated dataset, include the addition of noise, rotation and scaling. The development process closely followed the CRISP-DM methodology, which focuses on managing datasets. This new dataset was used for the experiment, and it was discussed as well as the model to be used for the experiment and how the evaluation of the model will be conducted, focusing on the use of the mean average precision (mAP) as a recognized performance indicator.

In the next chapter a detailed discussion of the experiment carried out to conduct the research and the evaluation of produced results will be discussed.

Chapter 4

The CIRSY System: Evaluation and Discussion

This chapter discusses the results of the implementation of the experiments conducted for this research using the CIRSY System, focusing on the evaluation of the proposed hypothesis and a discussion of the impact of the results on the proposed research question. As well as the strengths and weaknesses of the experiments as well as potential improvements that could have been made to the experiment.

4.1 Model Designs and Result

4.1.1 Model 1: Pre-Trained Coco Weights Only

Using only the default Mask R-CNN algorithm with the weights pre-trained on the MS COCO datasets as the first model, The model was tested on the 5 test datasets and the 3 versions of the generated test dataset. Visual inference was also performed on the original dataset. This model was created to examine the performance of the Mask R-CNN algorithm on a dataset such as the one used in this experiment.

Model 1 Results

The model performed poorly, most likely due to the difference between the quality of images containing birds from the MS COCO dataset and the chicks in the dataset used for this experiment. Table 4.1 below gives the result for Model 1.

	Mean Average Precision (mAP)
Test Dataset 1	0.000
Test Dataset 2	0.000
Test Dataset 3	0.042
Test Dataset 4	0.000
Test Dataset 5	0.007
Test Dataset No Transformations	0.012
Test Dataset With Transformations	0.013
Test Dataset With Noise	0.011

Table 4.1: Model 1 Results

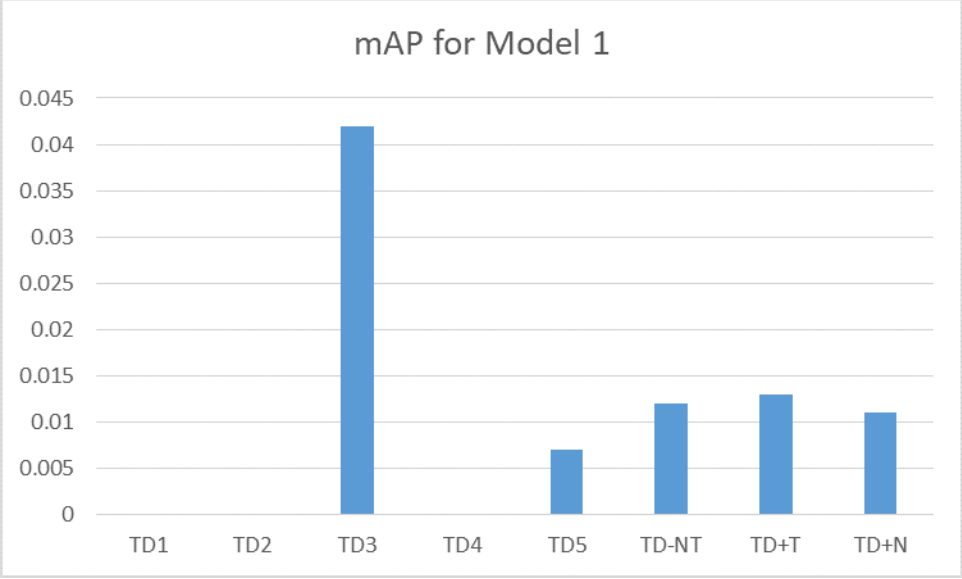
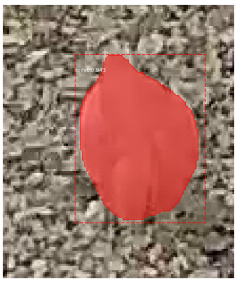


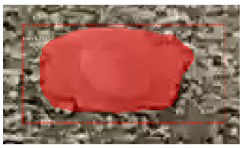
Figure 4.1: Model 1 Results

Model 1: Inference

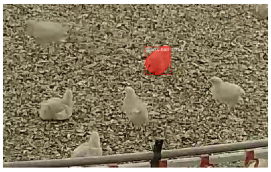
With a visual inference, the performance of the model on the dataset can be visualized. This gives a visual understanding of the models performance and can provide an overview into the low model performance.



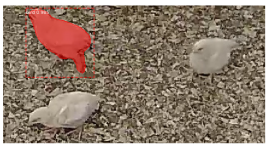
(a) Predicted Toilet



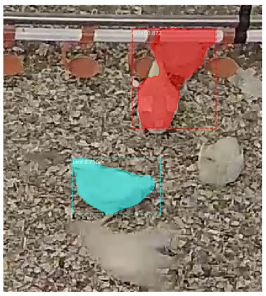
(b) Predicted Train



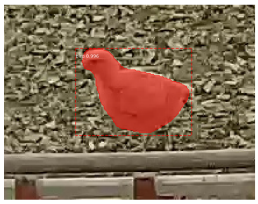
(c) Predicted Sports Ball



(d) Predicted Bird



(e) Predicted Bird



(f) Predicted Bird

Figure 4.2: Some wrong and correct inferences made by Model 1

Figure 4.3: Inference on Full CCTV footage Image

Figures 4.2 and 4.3 above show the visual inference of Model 1 on the dataset.

Not only is the model misidentifying a lot of the objects in the image, it also failed to detect majority of the objects in the images.

4.1.2 Model 2: Non Transformed Dataset

Using only the default Mask R-CNN model implementation hyper-parameters, but training on the synthetic generated dataset made without any form of image transformation. The model was trained on 10,000 training images each containing 3 chicks all labelled as birds. A validation set of 1,000 images was used as well. Only the head layer was trained for 12 epochs.

Model 2 Results

Table 4.2 below gives the result for Model 2. The results show a greater improvement from Model 1. Such an may be an argument for the importance of the training data being representative of the test data in computer vision tasks.

	Mean Average Precision (mAP)
Test Dataset 1	0.426
Test Dataset 2	0.626
Test Dataset 3	0.354
Test Dataset 4	0.658
Test Dataset 5	0.688
Test Dataset No Transformations	0.962
Test Dataset With Transformations	0.745
Test Dataset With Noise	0.188

Table 4.2: Model 2 Results

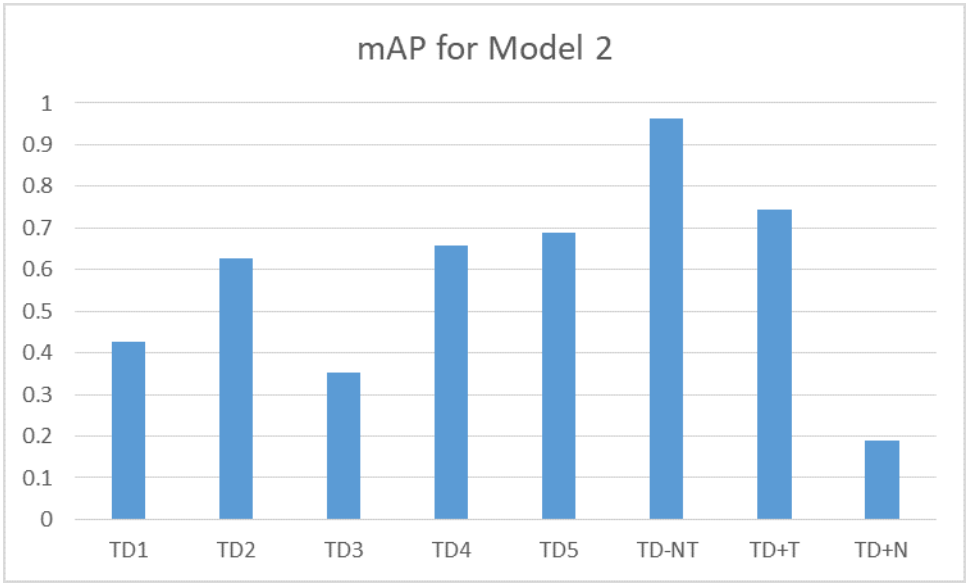


Figure 4.4: Model 2 Results

With the exception of Test Dataset 1 and 3, the model achieved good results on all test datasets. For the synthetic generated test datasets, the model performed very wall on the test dataset with no transformations. This must be due to the dataset being much more easier than the others. However, the model had its worst performance on the test dataset with noise. Although the dataset was also synthetically generated, the addition of noise caused great difficulty for the model and severely hampered its performance.

Reviewing the training and validation loss plot shows no sign of overfitting in the model.

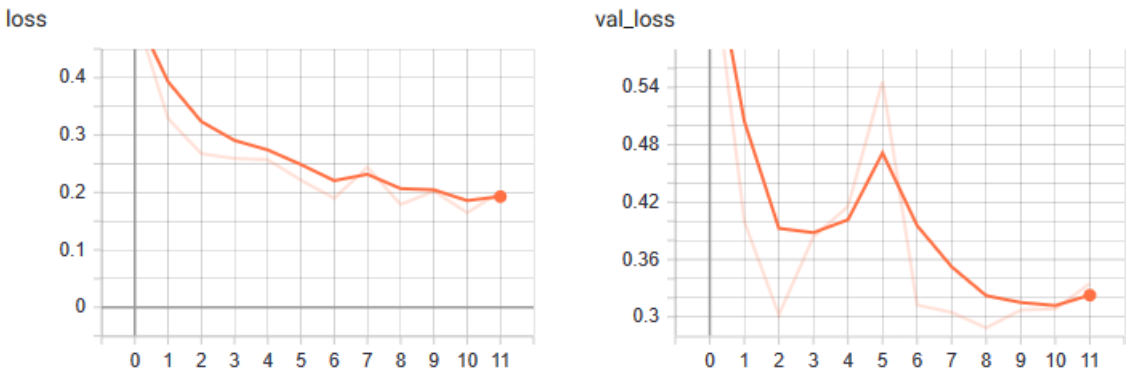


Figure 4.5: Model 2 Loss plots

4.1.3 Model 3: Transformed Dataset

Similar to Model 2, using the default hyper-parameters bu still training on a training dataset with 10,000 images and 1,000 validation images. However, the chicks in these images were transformed with various rotations, scaling and brightness settings applied to each chick before composing them on to the image. This augmentation process will allow for more variety from the composed 22 foreground images used to generate the training dataset.

Model 3 Results

Table 4.3 below gives the result for Model 3. The results show a greater improvement from both Models 1 and 2.

	Mean Average Precision (mAP)
Test Dataset 1	0.682
Test Dataset 2	0.650
Test Dataset 3	0.543
Test Dataset 4	0.758
Test Dataset 5	0.757
Test Dataset No Transformations	0.952
Test Dataset With Transformations	0.877
Test Dataset With Noise	0.025

Table 4.3: Model 3 Results

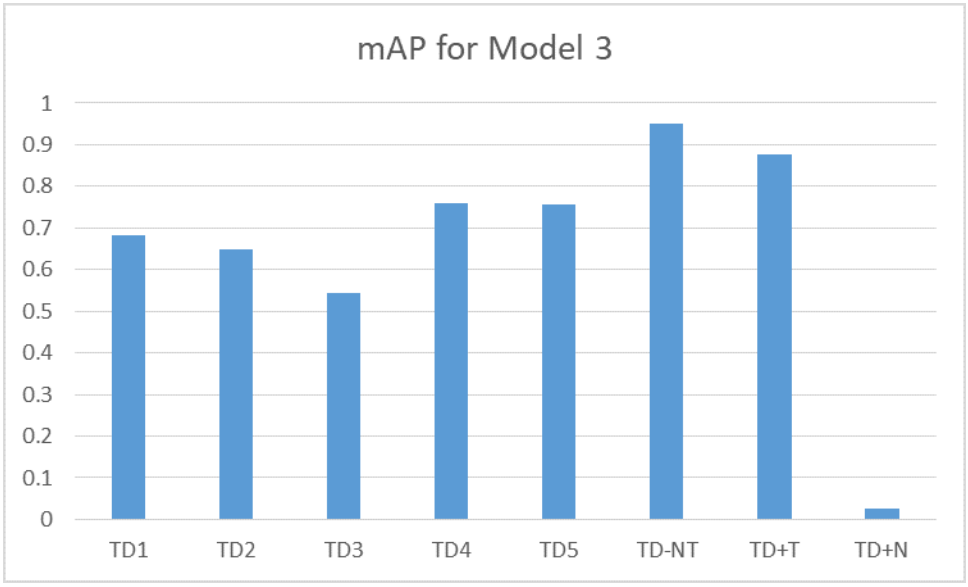


Figure 4.6: Model 3 Results

With the exception of Test Dataset 3, the model achieved good results on all test datasets. For the synthetic generated test datasets, the model performed poorly on the test dataset with noise, most likely due to he difficulty of detecting the objects in the noise as it was not trained with it.

Reviewing the training and validation loss plot shows no sign of overfitting in the model.

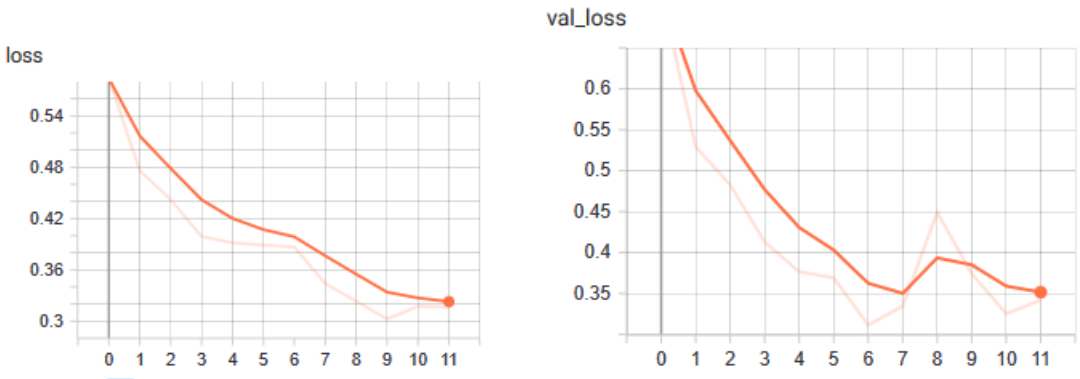


Figure 4.7: Model 3 Loss plots

4.1.4 Model 4: Noise Dataset

Like Model 2 and 3, using the default hyper-parameters but trained on the third version of the training dataset which has transformations applied to the foreground objects (the chicks) and salt and pepper noise applied to the overall image. The motivation for applying noise to the image is that by applying this layer of difficulty, the model will better identify the regions covered by the chicks in the test images.

Model 4 Results

Table 4.4 below gives the result for Model 3. The results show a greater improvement from both Models 1 and 2.

	Mean Average Precision (mAP)
Test Dataset 1	0.787
Test Dataset 2	0.879
Test Dataset 3	0.738
Test Dataset 4	0.873
Test Dataset 5	0.903
Test Dataset No Transformations	0.958
Test Dataset With Transformations	0.886
Test Dataset With Noise	0.830

Table 4.4: Model 4 Results

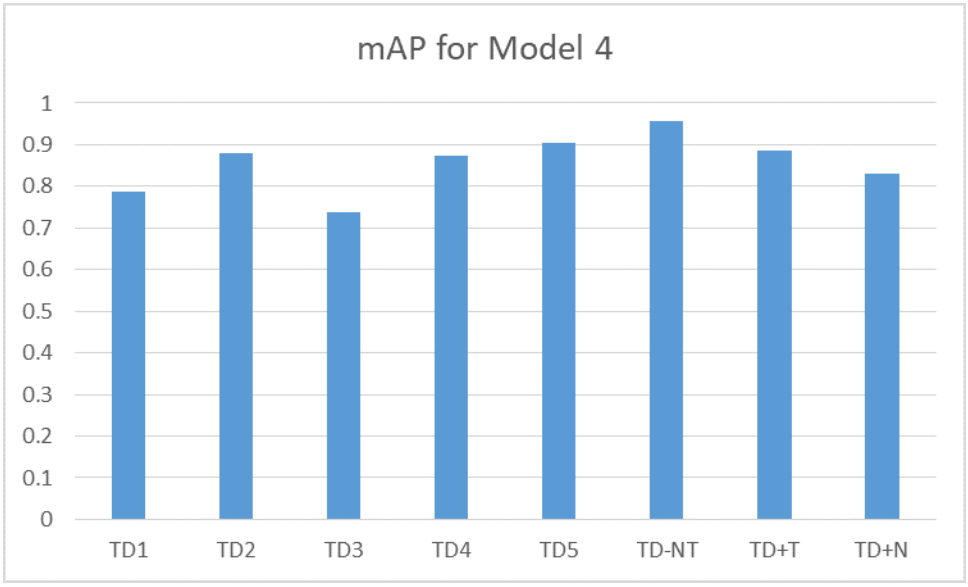


Figure 4.8: Model 4 Results

This model performs much better than all previous models (1,2 and 3) on all datasets. It may suggest that the increased difficulty of the training dataset due to the addition of salt and pepper noise allowed the model to improve it's general performance on the test datasets. As opposed to the previous models, the model performed much better on the noise synthetic dataset than the previous models.

Although the model did not achieve a similar loss value to the first two models, the loss plot below shows no clear signs of overfitting.

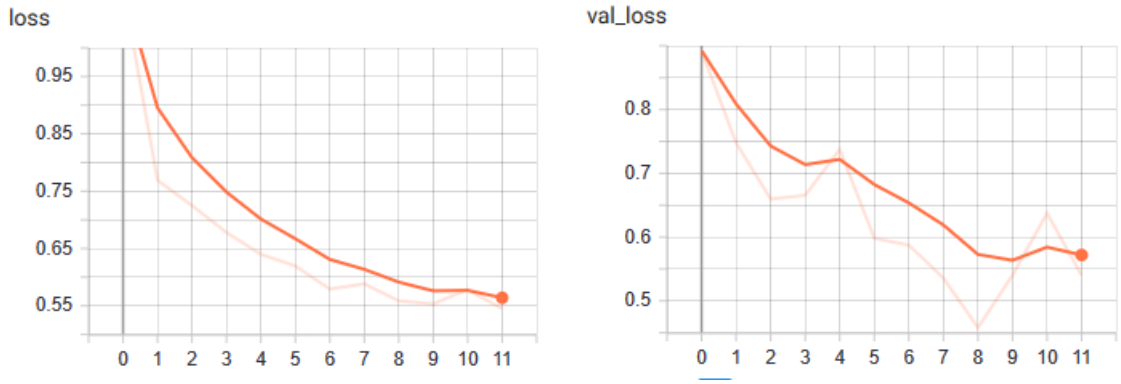


Figure 4.9: Model 4 Loss plots

4.1.5 Model 5: Fine Tuned Model 2

Model 5 builds on Model 2. In an attempt to fine-tune the model, keeping the same hyper-parameters, the model instead trains all layers of the network. The first 4 epochs are trained with the default learning rate (dlr) of 0.001, the learning rate for the next 4 epochs is then reduced by 10 (i.e. dlr/10) and the finally by another 10 (dlr/100) for the final 4 epochs.

Model 5 Results

Table 4.5 below gives the result for Model 5.

	Mean Average Precision (mAP)
Test Dataset 1	0.444
Test Dataset 2	0.406
Test Dataset 3	0.218
Test Dataset 4	0.600
Test Dataset 5	0.688
Test Dataset No Transformations	0.844
Test Dataset With Transformations	0.315
Test Dataset With Noise	0.067

Table 4.5: Model 5 Results

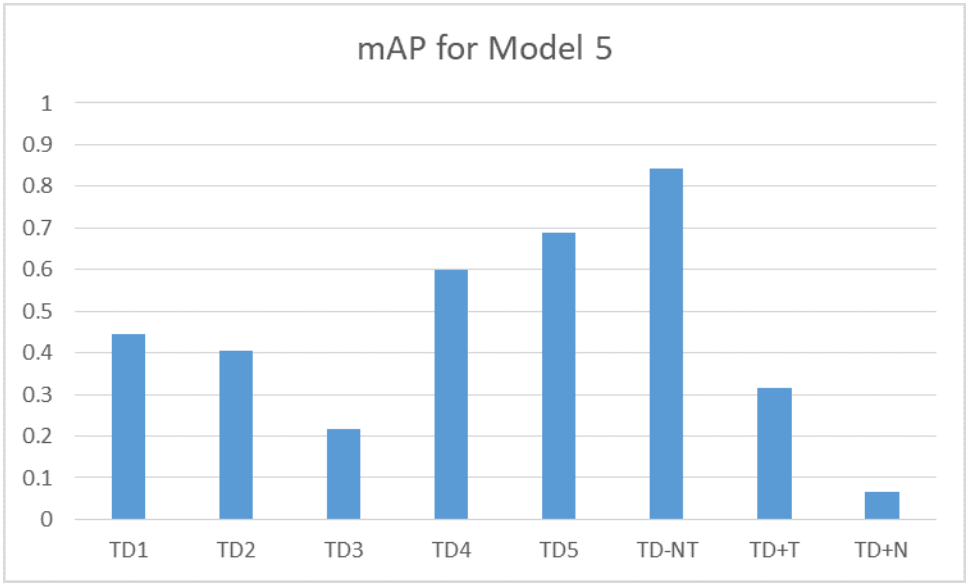


Figure 4.10: Model 5 Results

With the exception of Test Dataset 4 and 5, the model did not stand out. On the synthetic datasets, it performed extremely poor on the test dataset with noise.

The model achieved lower loss values in the training and validation set, although the ease of the dataset used in training and validation due to lack of noise and transformations may be the reason for this low loss value, as despite the low values, this model looks to be among the worst performing models so far.

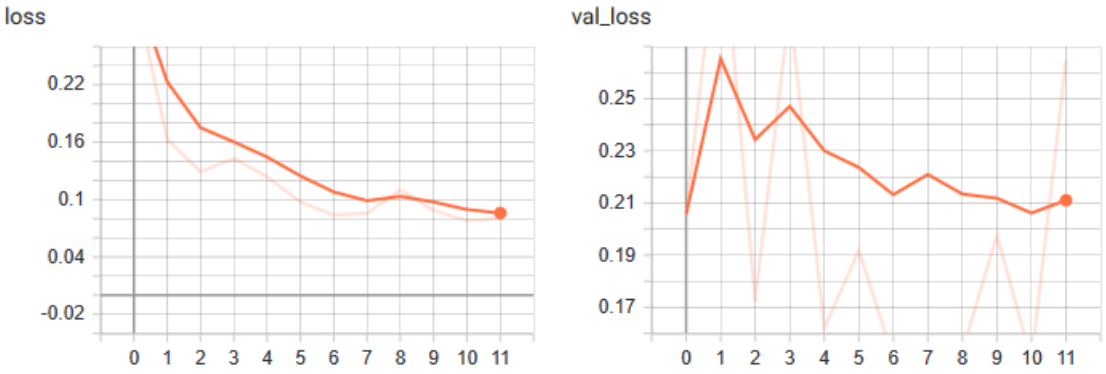


Figure 4.11: Model 5 Loss plots

4.1.6 Model 6: Fine Tuned Model 3

Similar to how Model 5 builds on Model 2, Model 6 builds on Model 3 as it involved training all layers of the network rather than just the heads. The model was trained for 12 epochs, with the first 4 epochs are trained with the default learning rate (dlr) of 0.001, the learning rate for the next 4 epochs is then reduced by 10 (i.e. dlr/10) and the finally by another 10 (dlr/100) for the final 4 epochs.

Model 6 Results

Table 4.6 below gives the result for Model 6.

	Mean Average Precision (mAP)
Test Dataset 1	0.790
Test Dataset 2	0.729
Test Dataset 3	0.710
Test Dataset 4	0.880
Test Dataset 5	0.917
Test Dataset No Transformations	0.985
Test Dataset With Transformations	0.936
Test Dataset With Noise	0.156

Table 4.6: Model 6 Results

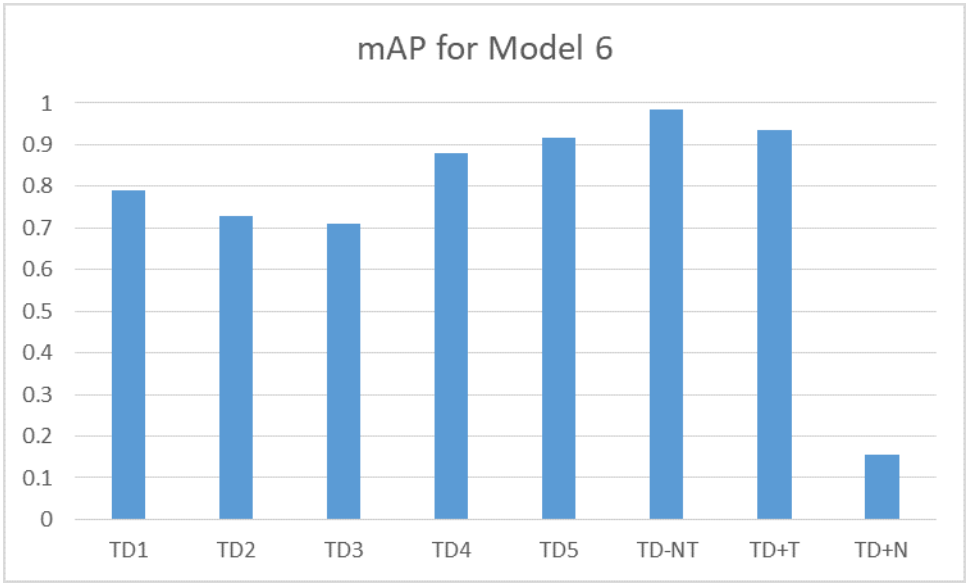


Figure 4.12: Model 6 Results

Compared to Model 5, this model achieved much better results on all dataset, with the exception of the synthetic test dataset with noise. However, at this stage such low performance has been expected from models trained without the noise dataset since no denoise technique was applied to any of the images before or after training and testing.

The loss plots of the model show no signs of the model overfitting. The strong performance of the model on the test datasets can also infer underfitting may also not have been an issue.

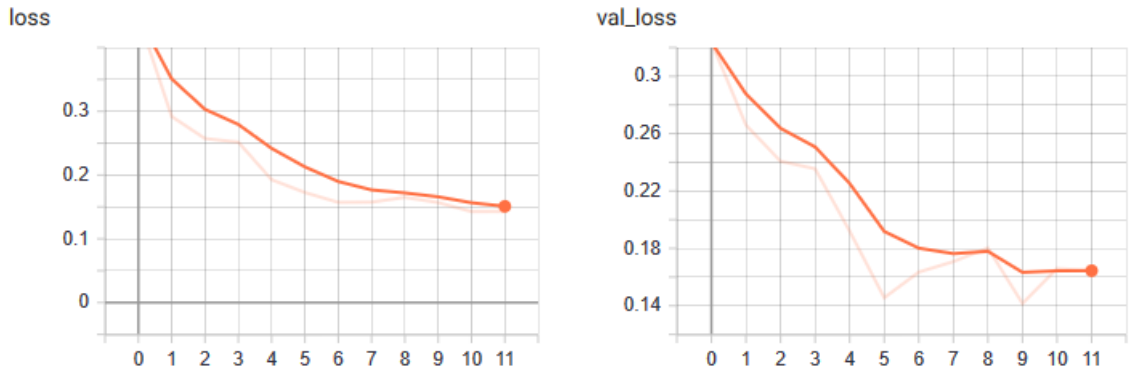


Figure 4.13: Model 6 Loss plots

4.1.7 Model 7: Fine Tuned Model 4

Model 7 builds on Model 4 by keeping the default Mask R-CNN implementation hyper-parameters, but training all layers of the network rather than just the heads. The model was trained for 12 epochs, with the first 4 epochs are trained with the default learning rate (dlr) of 0.001, the learning rate for the next 4 epochs is then reduced by 10 (i.e. dlr/10) and the finally by another 10 (dlr/100) for the final 4 epochs.

Model 7 Results

Table 4.7 below gives the result for Model 7.

	Mean Average Precision (mAP)
Test Dataset 1	0.799
Test Dataset 2	0.842
Test Dataset 3	0.738
Test Dataset 4	0.813
Test Dataset 5	0.931
Test Dataset No Transformations	0.984
Test Dataset With Transformations	0.889
Test Dataset With Noise	0.738

Table 4.7: Model 7 Results

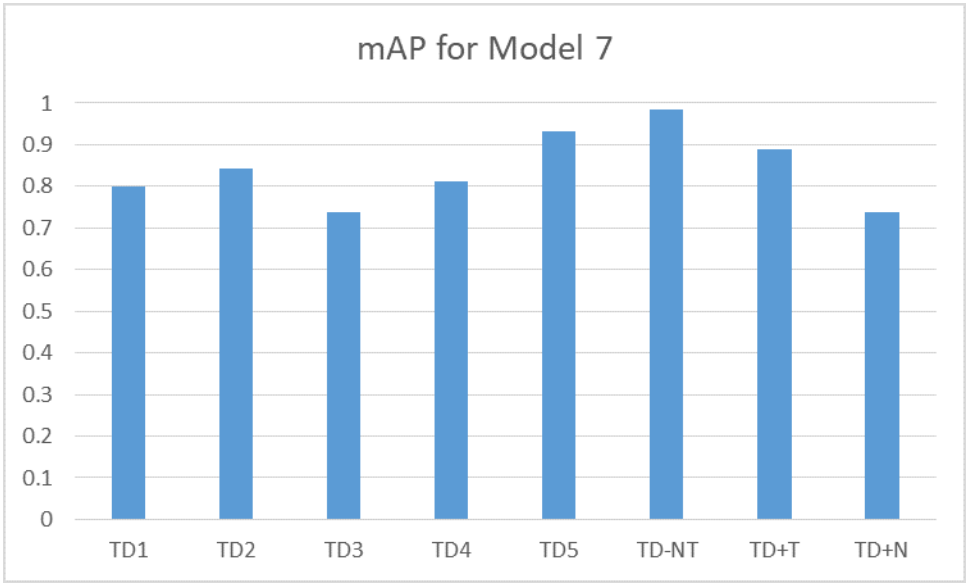


Figure 4.14: Model 7 Results

Similar to Model 6, this model achieved much better results on all dataset and unsurprisingly performed well on the synthetic dataset with noise.

The experiments so far have shown the importance of the dataset used in training as the greater the difficulty of the dataset used in training the better the model performance when evaluated on the test datasets.

The loss plots of the model were again inspected to check for overfitting of the model. The loss plots can be seen below in Figure 4.15

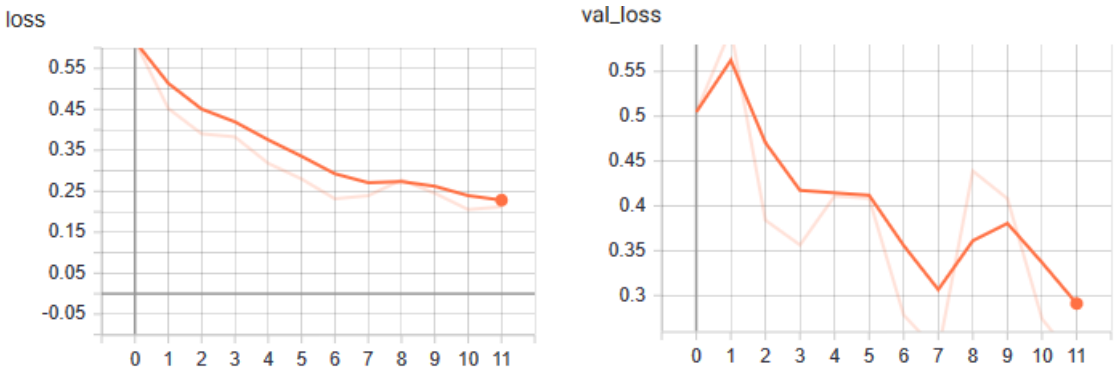


Figure 4.15: Model 7 Loss plots

4.1.8 Model 8: Fine Tuned Mask R-CNN Implementation

For the final model for the experiment, aside from training all layers of the network for 15 epochs and decreasing the learning rate from 0.001 by dividing it by 10 for the first 5 epochs and dividing it by 100 for the last 5 epochs, various hyper-parameters were also changed in order to better align the model with the goal of detecting a large number of small objects in an image. The hyper-parameters changed include:

- OPTIMIZER: Changing the optimizer from the default Stochastic gradient descent (SGD) to Adam.
- RPN_ANCHOR_SCALES: Reducing the RPN scale sizes to (8, 16, 32, 64, 12). This was done due to the chicks being small in respect to the image.
- RPN_TRAIN_ANCHORS_PER_IMAGE: The RPN anchors per image was increased to 500 due to the fact that the chicks could be in any region of the image.
- TRAIN_ROIS_PER_IMAGE: The ROIs to feed into the classifier was increased to 500 to allow more regions of interest where chicks may be located.
- STEPS_PER_EPOCH: Unlike the previous experiments where the steps per epoch was always 1000, the number of steps per epoch for this model was increased to 10,000 to match the number of training images. 10,000 was selected since GPU_COUNT and IMAGES_PER_GPU were set to 1 for all experiments in this research.
- VALIDATION_STEPS: The validation steps for this model was increased from 50 to 1,000 in order to match the number of validation images.
- AUGMENT: The augment parameter was set to true to use the implementations augmentation function during training.
- SCALE: The augment parameter was set to true to use the implementations scaling and rotation function during training.

The model was also trained on the synthetic training dataset with noise, which has so far been shown to generate better results when evaluating all prior models.

Model 8 Results

The model performed very well on all test datasets, with its weakest performance coming in test dataset 3 which has so far been seen as the most difficult test dataset.

	Mean Average Precision (mAP)
Test Dataset 1	0.830
Test Dataset 2	0.847
Test Dataset 3	0.696
Test Dataset 4	0.893
Test Dataset 5	0.861
Test Dataset No Transformations	0.951
Test Dataset With Transformations	0.907
Test Dataset With Noise	0.789

Table 4.8: Model 8 Results

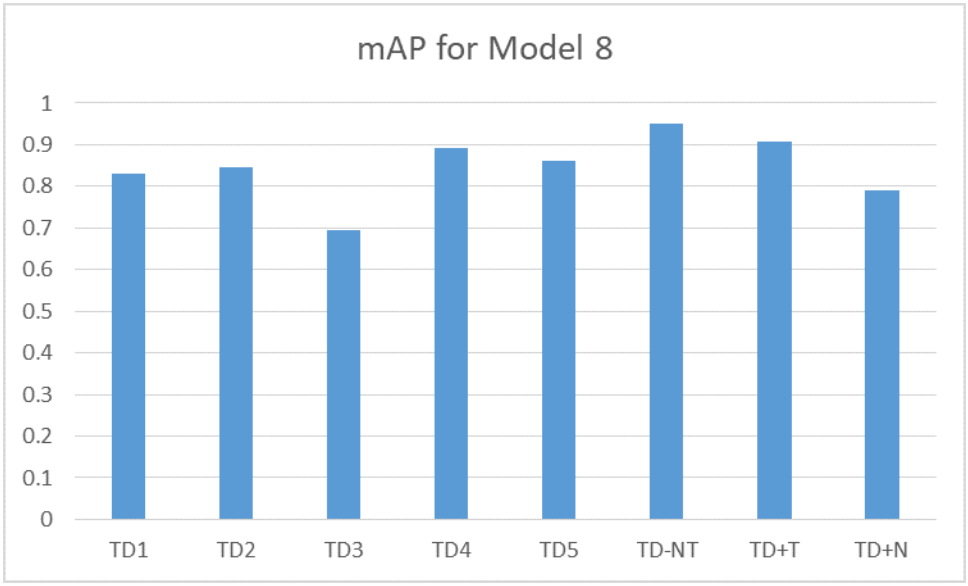


Figure 4.16: Model 8 Results

The loss plots for model 8 can be seen in Figure 4.17 below. Like with the previous trained models, this model shows no signs of overfitting or even underfitting. The model achieved the greatest decrease in loss in the first 5 epochs. It went on to slowly elbow over the next 5 epochs and no significant change in the loss value could be seen over the last 5 epochs.

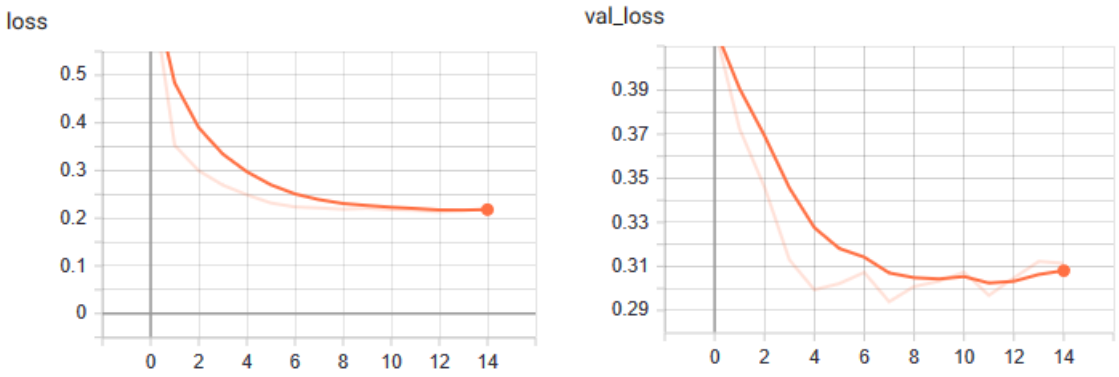


Figure 4.17: Model 8 Loss plots

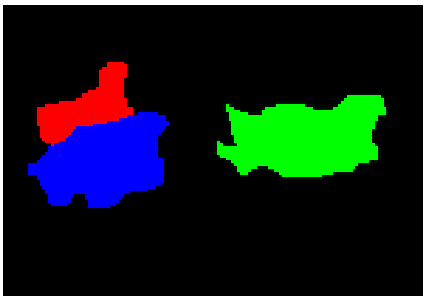
4.2 Interesting Model Inferences

This section reviews some challenging images evaluated by the models and discusses the models performance.

4.2.1 Test Dataset Inferences

Figure 4.18 below gives two challenging images from the test dataset and their masks to clearly show the location of the chicks in the image.

(a) Image 1



(b) Image 1 Mask

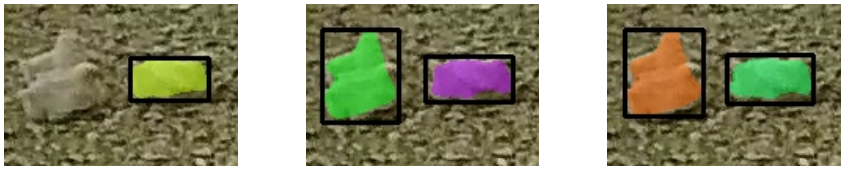
(c) Image 2

(d) Image 2 Mask

Figure 4.18: Some Challenging images from the Test Dataset

Figure 4.19 below shows the performance of all models on Image 1. It shows that no model was able to correctly predict the presence of 3 different chicks in the image despite the increase in complexity of the model. At most, the models could predict two images predicting the two chicks on the left as 1. This error is understandable due to the closeness of the chicks and similarity of the shape formed with majority of that used to train the model.

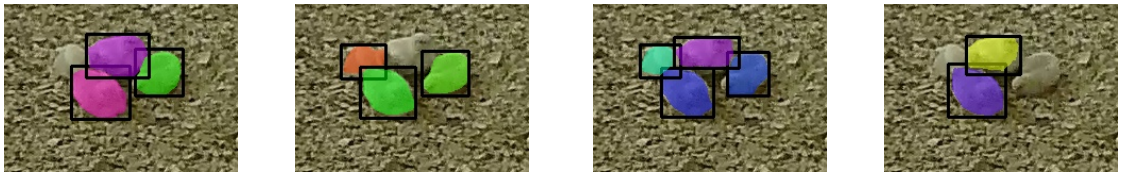
(a) Model 2 Prediction(b) Model 3 Prediction (c) Model 4 Prediction(d) Model 5 Prediction



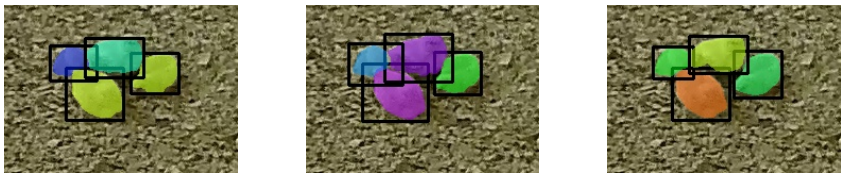
(e) Model 6 Prediction (f) Model 7 Prediction (g) Model 8 Prediction

Figure 4.19: Model Predictions for Image 1

Interestingly, Figure 4.20 below shows an image with almost similar closeness between the chicks, however, most likely due to the clearer appearance of edges, as the model increase in complexity, they are able to solve this problem. It should be noted that due to the variability provided from adding transformations and noise in the synthetic training datasets, this research considers Models 3 and 4 to be more complex or stronger than Model 5.



(a) Model 2 Prediction(b) Model 3 Prediction (c) Model 4 Prediction(d) Model 5 Prediction



(e) Model 6 Prediction (f) Model 7 Prediction (g) Model 8 Prediction

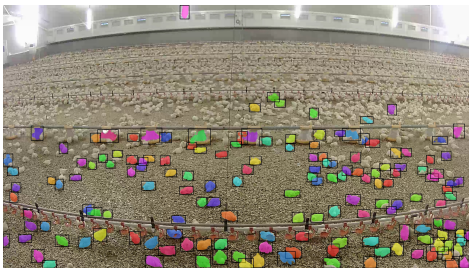
Figure 4.20: Model Predictions for Image 2

4.2.2 CCTV Footage Inference

Figure 4.21 below shows the model predictions on a Sample full sized CCTV image. It should be noted that without changing the IMAGE_MAX_DIM parameter from the default Mask R-CNN implementation value to 2560, no model could perform any inference on these full sized CCTV images in this research.

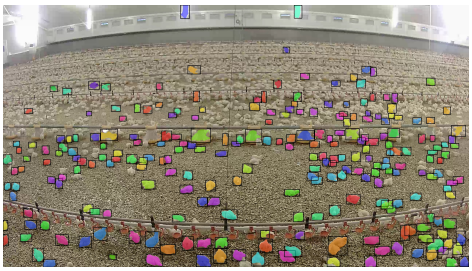
Although it is difficult to measure each models performance on these images as they were not annotated, by visually examining the model performance, it shows a good level of performance from all models.

The inference images show that the models all performed similarly well in predicting instances of chicks which appeared towards the front of the image as opposed to the rear of the image which is very difficult to see. This research did not use the "iscrowd" feature for COCO like datasets and thus is not particularly interested in model detection of crowd instances for this research.

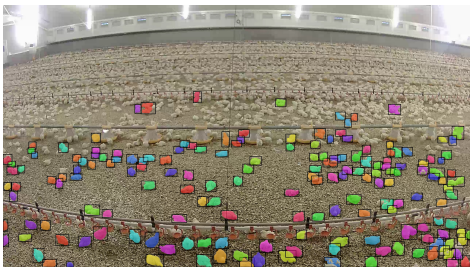


(a) Model 2 Prediction

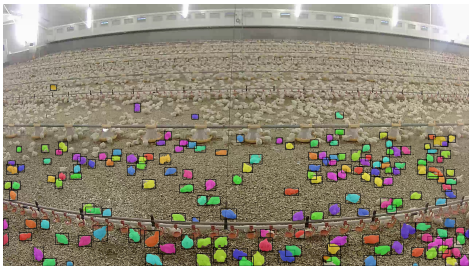
(b) Model 3 Prediction



(c) Model 4 Prediction

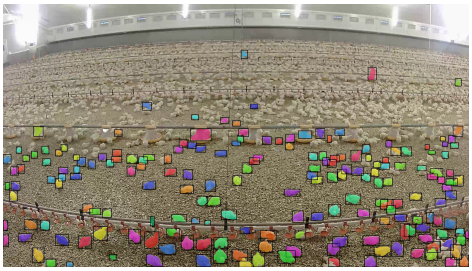


(d) Model 5 Prediction

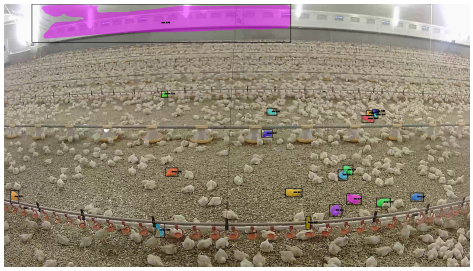


(e) Model 6 Prediction

(f) Model 7 Prediction



(g) Model 8 Prediction



(h) Model 1 Prediction

Figure 4.21: Model Predictions on Sample CCTV Image

Reviewing Figure 4.21 above, with the exception of Model 1 understandably, all models performed reasonably well on the CCTV footage. However, models 4, 7 and 8 performed better in predicting instances of chicks further back in the images than other models.

4.3 Hypothesis Testing

To answer the research question and explore the hypothesis of this experiment, difference tests will be conducted on the results of the models.

The null hypothesis states fine tuning the models will not provide a significant improvement in the mAP of the model when evaluated on the test datasets. Hence to conduct this test, a Right Tailed Test will be conducted to test if the mAP of the models significantly improved after fine tuning the model.

The significance value (p value) for these statistical test is less than or equal to .05.

4.3.1 Right Tailed Test Models 2 & 5

A right tailed test was conducted between the mAP results of Models 2 and 5 to compare if the mAP of the model improved after fine tuning. There was no significant increase in the mean Average Precision of Model 5 (M = 0.448, SD = 0.255) compared to Model 2 (M = 0.581, SD = 0.245), t(7) = -2.641, $p > .05$

4.3.2 Right Tailed Test Models 3 & 6

A right tailed test was conducted between the mAP results of Models 3 and 6 to compare if the mAP of the model improved after fine tuning. There was a significant increase in the mean Average Precision of Model 6 (M = 0.763, SD = 0.265) compared to Model 3 (M = 0.666, SD = 0.258), t(7) = 5.362, $p < .05$

4.3.3 Right Tailed Test Models 4 & 7

A right tailed test was conducted between the mAP results of Models 4 and 7 to compare if the mAP of the model improved after fine tuning. There was no significant increase in the mean Average Precision of Model 7 (M = 0.842, SD = 0.088) compared to Model 4 (M = 0.857, SD = 0.069), t(7) = -0.975, $p > .05$

4.3.4 Right Tailed Test Models 4 & 8

A right tailed test was conducted between the mAP results of Models 4 and 8 to compare if the mAP of the model improved after fine tuning. There was no significant increase in the mean Average Precision of Model 8 (M = 0.847, SD = 0.079) compared to Model 4 (M = 0.857, SD = 0.069), t(7) = -0.828, p > .05

4.3.5 Hypothesis Evaluation

With the exception of the Right Tailed Test between Model 3 and its fine tuned model Model 6, we fail to reject the other Null hypothesis for the other 3 experiments. These results suggest that perhaps fine tuning training and hyperparameters of the Mask R-CNN implementation when using the pre-trained MS COCO weights for instance segmentation tasks may not significantly improve the model performance provided the data used to train the model is representative of the problem the model is required to solve.

4.4 Summary

This chapter reviewed the results of the experiments done to test the hypothesis. The results showed using the implementation of the Mask R-CNN model trained on the MS COCO dataset was unable to properly identify a decent number of chicks in the test images or orignal CCTV footage and misclassified majority of the few instances it did detect.

By training the model on a synthtetic dataset similar to the desired task required to be solved by the model, the performance greatly increased. The results obtained from the experiments conducted in this research suggests that provided the dataset used in training the model is relevant and representative of the desired problem to solve, fine-tuning the model did not provide an increase in performance that was deemed statistically significant.

As discussed in the literature review, the results from these experiments show

the benefits of transfer learning. As by using the weights trained on the MS COCO dataset, the model could quickly train and perform very well without the need to train from scratch. Given that the MS-COCO dataset had the bird class as well may have been an added factor as to the overall strong performance of the models.

The next chapter reviews the overall reseach conducted and assesses the impact of this study as well as the limitations that may have been experienced and a future direction for further devlopment of this reseach.

Chapter 5

Conclusions and Future Work

5.1 Research Overview

This research explored the impact of the use of the CRISY system, based on the Mask R-CNN algorithm to instance segmentation tasks, particularly detecting a large number of instances of small objects in low resolution footage where it was difficult to differentiate between an object and the background.

For this study, CCTV footage of chicks in a poultry farm was used as it met the criteria of the study. It provided an environment which had a large number of instances of objects to be detected and these objects where difficult to separate from the background with the human eye.

5.2 Problem Definition

Although studies have shown the effectiveness of Mask R-CNN in image instance segmentation tasks, they focus on well lit images where the objects of interest are clearly differentiated from the background and other instances within the image.

This research looked to study how effective the Mask R-CNN algorithm will be on detecting a large number of instances in low lit images where the objects are not easily separable from the background.

Such images have been known to pose a problem for traditional image segmentation

techniques as they experience difficulties in processing images with a large number of objects, due to too many edges and even experience difficulties in separating objects from the background when no clear dissimilarity exists between them.

5.3 Design & Experimentation

This research was conducted to test the hypothesis: There will be no statistically significant improvement in the mean average precision of the Mask R-CNN algorithm applied to a low resolution images after the algorithm has been fine tuned.

To test the hypothesis, synthetic datasets were generated to train and test the models. The use of synthtetic datasets were to remove the difficulty of manually annotating images which contained a large number of chicks in order to train and test the model. The datasets were made by composing cropped chicks from the collection of CCTV images with various sections of the cropped out background of those images.

Three versions of the synthetic dataset was generated. The first had no transformations applied to the foreground images, the second had transformations in the form of scaling, rotating and brightness adjusting on the foreground images and the third had salt and pepper noise applied to the composed images which had received similar transformations with the second version.

5.4 Evaluation & Results

The results showed improvements in the model performance based on the synthetic datasets used. As the models which used the first version with no transformations achieved lower results than the other models which used the other two versions of the dataset.

The best performing models where those trained on the synthetic dataset which had salt and pepper noise applied to the images. This suggests that the added level of difficult in training the model provided by the addition of the noise allowed it to better detect the chciks in the image compared to other models. In respect to inferring

on the original CCTV footage to study how the model performed, with the exception of the model which used only the weights trained on the MS COCO dataset (Model 1) all models were able to satisfactorily infer the objects (chicks) when applied on the original CCTV footage.

In respect to the hypothesis, a right tailed test showed no significant improvement in a models performance after various fine tuning operations for each model trained on the same dataset and thus this experiment failed to reject the null hypothesis.

5.5 Contributions and impact

This research has proven without a doubt the benefit of training computer vision models to perform instance segmentation and object detection tasks on synthetic generated datasets that are representative of the desired problem to be solved.

By training models on such datasets, time and cost can be saved from manually annotating a large number of images in order to build training datasets for such tasks.

Given the training images composed of a maximum of 3 instances of chicks per image, but were still able to detect 100+ instances of chicks in the original CCTV footage image, it is now clear the number of instances on the training image is not a factor that will affect the number of instances the model can detect in a test image.

This research has shown that provided the model has been sufficiently trained with the appropriate dataset, the Mask R-CNN algorithm can effectively detect and segment instances of small objects in poorly lit images where it can be difficult to differentiate the object of interest from the background.

The design of this experiment reinforce the relevance and effectiveness of transfer learning for deep learning tasks as the Models did not require extensive training time to achieve good results most likely due to the use of the weights applied from the MS COCO dataset task.

5.6 Future Recommendations

Even though the models could satisfactorily detect the instances of chicks towards the front of the image, the chicks at the back where unable to be detected by the models. A future advancement could be to further tune and train the models to attempt to detect those instances.

The models were all trained with two classes, bird which signified the chicks desired to be detected and the background. Clearly this provided a level of simplicity for the model in tackling the test data. As seen when evaluating the default implementation of the Mask R-CNN model using only the trained weights, the model missclassified majority of the instances of chicks as other objects. Below in figure 5.1 the model performance on a sample CCTV image, the rear end chickens are undetected as shown clearly in figure 5.2

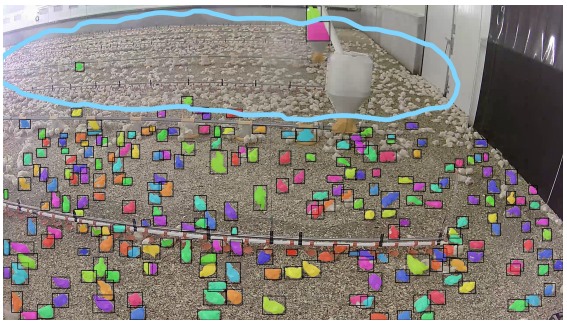


Figure 5.1: Model CCTV Image Detection

Figure 5.2: Undetected Back/Rear Images

A direction to take this research would be to add other classes based on the area of interest. For example as this dataset is representative of a poultry farm classes such as other animals, farm equipment, people and food or crops could be added to not only increase the complexity of the model but also improve its application in real world scenarios. In figures 5.3 to 5.5 below, the models occasionally predicted some equipment such as lights, feeders of gate as chickens.

Figure 5.3: Lights

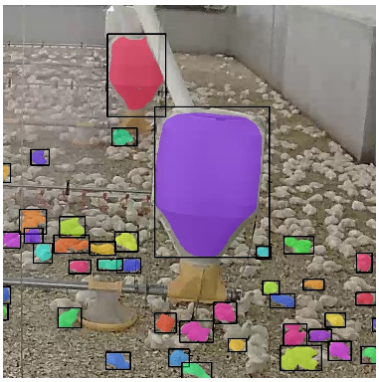


Figure 5.4: Feeder

Figure 5.5: Gate

A further extension of the research could be to explore the implementation using a range of neural network models, e.g. Recurrent Neural Networks, Long/Short Term Memory Networks, and Deep Belief Networks.

- Recurrent Neural Networks (RNNs) unlike traditional Neural Nets have the outputs from previous steps used as inputs for the present step. This ability to allow information persists in RNNs may be seen as an advantage over other neural network architectures. With Ren and Zemel (2017) highlighting the benefits of using RNNs to take advantage of the scenery changes in an image for instance level segmentation.

- Long/Short Term Memory Networks are a form of RNNs which tries to remember all information seen by the model and only forget the information considered to be irrelevant in order to tackle the vanishing gradient problem.

- Another approach to instance segmentation may be through unsupervised learning. Neural networks with unsupervised learning architectures such a Autoencoders (Baldi, 2012) and Deep Belief Networks (Hinton, Osindero, & Teh, 2006) can allow for improved learning of non linear features in the images.

Additionally other approaches to evaluation could be explored, including: RMSE (root-mean-square error), BMR (block-to-mask ratio), EOBD (effect-of-block-distortion), Signal-to-noise ratio, and Contrast-to-noise ratio.

Additionaly, by using sub-pixel accuracy, the model can improve the detection of regions covered by the objects of interest and interpolating the objects to the nearest pixel. This should in theory improve the accuracy of the detection of regions covered by the objects of interest.

References

Abdel-Maksoud, E., Elmogy, M., & Al-Awadi, R. (2015). Brain tumor segmentation based on a hybrid clustering technique. *Egyptian Informatics Journal*, *16*(1), 71–81.

Abdulla, W. (2017). *Mask r-cnn for object detection and instance segmentation on keras and tensorflow.* `https://github.com/matterport/Mask_RCNN`. Github.

Alexe, B., Deselaers, T., & Ferrari, V. (2010). What is an object? In *2010 ieee computer society conference on computer vision and pattern recognition* (pp. 73–80).

Azzeh, J., Zahran, B., & Alqadi, Z. (2018). Salt and pepper noise: Effects and removal. *JOIV: International Journal on Informatics Visualization*, *2*(4), 252–256.

Baldi, P. (2012, 02 Jul). Autoencoders, unsupervised learning, and deep architectures. In I. Guyon, G. Dror, V. Lemaire, G. Taylor, & D. Silver (Eds.), *Proceedings of icml workshop on unsupervised and transfer learning* (Vol. 27, pp. 37–49). Bellevue, Washington, USA: PMLR. Retrieved from `http://proceedings.mlr.press/v27/baldi12a.html`

Bazeille, S., Quidu, I., & Jaulin, L. (2006). Automatic underwater image pre-processing. In *in proceedings of the caracterisation du milieu marin (cmm'06.*

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. In *Proceedings of icml workshop on unsupervised and transfer learning* (pp. 17–36).

Bhargavi, K., & Jyothi, S. (2014). A survey on threshold based segmentation technique in image processing. *International Journal of Innovative Research and Development*, *3*(12), 234–239.

Bosch, A., Zisserman, A., & Munoz, X. (2007). Image classification using random forests and ferns. In *2007 ieee 11th international conference on computer vision* (pp. 1–8).

Chen, T., Lu, S., & Fan, J. (2017). S-cnn: Subcategory-aware convolutional networks for object detection. *IEEE transactions on pattern analysis and machine intelligence*, *40*(10), 2522–2528.

Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *2005 ieee computer society conference on computer vision and pattern recognition (cvpr'05)* (Vol. 1, p. 886-893 vol. 1). doi: 10.1109/CVPR.2005.177

Dhanachandra, N., Manglem, K., & Chanu, Y. J. (2015). Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, *54*, 764–771.

Du, C. (2004). Dw sun. *Recent developments in the applications of image processing techniques for food quality evaluation. Trends in Food Science & Technology*, *15*(5), 230–249.

Fu, B., Zhao, X., Song, C., Li, X., & Wang, X. (2019). A salt and pepper noise image denoising method based on the generative classification. *Multimedia Tools and Applications*, *78*(9), 12043–12053.

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., & Garcia-Rodriguez, J. (2017). A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*.

Girshick, R. (2015). Fast r-cnn. In *2015 ieee international conference on computer vision (iccv)* (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 ieee conference on computer vision and pattern recognition* (pp. 580–587).

Grundland, M., & Dodgson, N. A. (2007). Decolorize: Fast, contrast enhancing, color to grayscale conversion. *Pattern Recognition*, *40*(11), 2891–2896.

Guennouni, S., Ahaitouf, A., & Mansouri, A. (2015). A comparative study of multiple object detection using haar-like feature selection and local binary patterns in several platforms. *Modelling and Simulation in Engineering*, *2015*, 17.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the ieee international conference on computer vision* (pp. 2961–2969).

He, K., & Sun, J. (2014). Convolutional neural networks at constrained time cost. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5353-5360.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

Inference, G.-B. D. L. (2015). A performance and power analysis. *Whitepaper, November*.

Jones, M., & Viola, P. (2003). Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, *3*(14), 2.

Kaganami, H. G., & Beiji, Z. (2009). Region-based segmentation versus edge detection. In *2009 fifth international conference on intelligent information hiding and multimedia signal processing* (pp. 1217–1221).

Kanan, C., & Cottrell, G. W. (2012, 01). Color-to-grayscale: Does the method matter in image recognition? *PLOS ONE*, *7*(1), 1-7. Retrieved from `https://doi.org/10.1371/journal.pone.0029740` doi: 10.1371/journal.pone.0029740

Kaur, D., & Kaur, Y. (2014). Various image segmentation techniques: a review. *International Journal of Computer Science and Mobile Computing*, *3*(5), 809–814.

Kaur, M., & Goyal, P. (2015). A review on region based segmentation. *International Journal of Science and Research (IJSR)*, *4*(4), 3194–3197.

Kiran, J. S., Kumar, N. V., Prabha, N. S., & Kavya, M. (2015). A literature survey on digital image processing techniques in character recognition of indian languages. *International Journal of Computer Science and Information Technologies*, *6*(3), 2065–2069.

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

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

Li, Y., Qi, H., Dai, J., Ji, X., & Wei, Y. (2017). Fully convolutional instance-aware semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2359–2367).

Lin, E.-U., McLaughlin, M., Alshehri, A. A., Ezekiel, S., & Farag, W. (2014). Medical image segmentation using multi-scale and super-resolution method. In *2014 ieee applied imagery pattern recognition workshop (aipr)* (pp. 1–5).

Lin, H. (2008). Method of image segmentation on high-resolution image and classification for land covers. In *2008 fourth international conference on natural computation* (Vol. 5, pp. 563–566).

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3431–3440).

Marr, D., & Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, *207*(1167), 187–217.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Ng, H., Ong, S., Foong, K., Goh, P., & Nowinski, W. (2006). Medical image segmentation using k-means clustering and improved watershed algorithm. In *2006 ieee southwest symposium on image analysis and interpretation* (pp. 61–65).

Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25). Determination press San Francisco, CA, USA:.

Nievergelt, J. (1969). R69-13 perceptrons: An introduction to computational geometry. *IEEE Transactions on Computers*, *100*(6), 572–572.

Pal, N. R., & Pal, S. K. (1993). A review on image segmentation techniques. *Pattern recognition*, *26*(9), 1277–1294.

Pan, S. J., & Yang, Q. (2010, Oct). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345-1359. doi: 10.1109/TKDE.2009.191

Ray, S., & Turi, R. H. (1999). Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques* (pp. 137–143).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 779–788).

Ren, M., & Zemel, R. S. (2017, July). End-to-end instance segmentation with recurrent attention. In *2017 ieee conference on computer vision and pattern recognition (cvpr)* (p. 293-301). doi: 10.1109/CVPR.2017.39

Ren, S., He, K., Girshick, R., & 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).

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*(3), 1.

Russo, F. (2002). An image enhancement technique combining sharpening and noise reduction. *IEEE Transactions on Instrumentation and Measurement*, *51*(4), 824–828.

Saha, S., Basu, S., & Nasipuri, M. (2014). A comprehensive survey on different techniques and applications of digital image processing. , 75–87.

Saini, S., & Arora, K. (2014). A study analysis on the different image segmentation techniques. *International Journal of Information & Computation Technology*, *4*(14), 1445–1452.

Saravanan, C. (2010). Color image to grayscale image conversion. In *2010 second international conference on computer engineering and applications* (Vol. 2, pp. 196–199).

Sarmadi, S., & Shamsa, Z. (2016). A new approach in single image super resolution. In *2016 6th international conference on computer and knowledge engineering (iccke)* (pp. 78–81).

Senthilkumaran, N., & Rajesh, R. (2009). Edge detection techniques for image segmentation-a survey of soft computing approaches. *International journal of recent trends in engineering*, *1*(2), 250.

Senthilkumaran, N., & Vaithegi, S. (2016). Image segmentation by using thresholding techniques for medical images. *Computer Science & Engineering: An International Journal*, *6*(1), 1–13.

Sharifi, M., Fathy, M., & Mahmoudi, M. T. (2002). A classified and comparative study of edge detection algorithms. In *Proceedings. international conference on information technology: Coding and computing* (pp. 117–120).

Shi, Z., & Govindaraju, V. (2004). Historical document image enhancement using background light intensity normalization. In *Proceedings of the 17th international conference on pattern recognition, 2004. icpr 2004.* (Vol. 1, pp. 473–476).

Shi, Z., Setlur, S., & Govindaraju, V. (2004). Digital enhancement of palm leaf manuscript images using normalization techniques. In *5th international conference on knowledge based computer systems* (pp. 19–22).

Solem, J. E. (2012). *Programming computer vision with python: Tools and algorithms for analyzing images.* " O'Reilly Media, Inc.".

Sridevi, M., & Mala, C. (2012). A survey on monochrome image segmentation methods. *Procedia Technology*, *6*, 548–555.

Sumithra, K., Buvana, S., & Somasundaram, R. (2015). A survey on various types of image processing technique. *International Journal of Engineering Research & Technology (IJERT) Vol*, *4*(3), 399–403.

Tatiraju, S., & Mehta, A. (2008). Image segmentation using k-means clustering, em and normalized cuts. *Department of EECS*, *1*, 1–7.

Vernon, D. (1991). Machine vision-automated visual inspection and robot vision. *NASA STI/Recon Technical Report A*, *92*.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, *1*(511-518), 3.

Wirth, R., & Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining* (pp. 29–39).

Zafeiriou, S., Zhang, C., & Zhang, Z. (2015, 04). A survey on face detection in the wild: past, present and future. *Computer Vision and Image Understanding*, *138*. doi: 10.1016/j.cviu.2015.03.015

Zhang, D., & He, J. (2014). Super-resolution reconstruction of low-resolution vehicle plates: A comparative study and a new algorithm. In *2014 7th international congress on image and signal processing* (pp. 359–364).

Zhao, Z.-Q., Zheng, P., tao Xu, S., & Wu, X. (2018). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 1-21.

Appendix A

Sample Dataset

Below are sample images from the collection of 186 CCTV Images obtained from Dr. Robert Ross. Various chicks were cropped from these images and sections of the background were extracted in order to compose together the synthetic datasets used to train and test the model used in this research.


Camera8


Camera8


Camera8