FACULTY OF SCIENCE, ENGINEERING AND SOCIAL SCIENCES

Automatic non-biting midge (Chironomidae) identification through the application of object detection and deep learning techniques

By
Jack D Hollister

Canterbury Christ Church University

Thesis submitted for the degree of MSc by Research in Biology.

2020

Table of Contents

i. Figures and Tables 5
ii. Acknowledgments 9
iii. Abstract 10
1. Introduction 12
1.1 Freshwater ecosystems 14
1.2 Chironomids 16
1.3 Visual identification of chironomids 17
1.4 Identification through DNA barcoding 19
1.5 eDNA metabarcoding 20
1.6 Automated identification of organisms 21
1.7 Computer Vision 21
1.8 Neural Networks 23
1.9 Convolution neural networks 24
1.10 Image Classification and Object detection 26
1.11 Evaluation of Object detection models 28
1.12 TensorFlow and TensorBoard 30
1.13 Finetuning an object detection model 31
1.14 Quality of data 33
1.15 Transfer learning 35
2. Aims and Hypothesis 38
2.1 Aims 38
2.2 Hypothesis 39
3. Methodology 40
3.1 Specimen collection and Taxonomic identification 40
3.2 Dataset preparation and Image annotation 41
3.2.1 Dataset construction 41
3.2.2 Creating the bounding boxes 46
3.3 Pre-processing 48
3.3.1 Quality of data 48
3.3.2 Splitting of test images 50
3.3.3 Test image removal 50

3.3.4 Creation of image sets 51
3.3.5 Up-sampling by image augmentation 52
3.3.6 Down-Sampling 54
3.3.7 Nonbiased image organisation 55
3.3.8 Model architecture construction 55
3.3.9 Learning rate hyperparameter 57
3.3.10 IOU Union threshold hyperparameter 58
3.3.11 Configuration files 58
3.4 Training 61
3.4.1 Workstation 61
3.4.2 TensorFlow Installation 61
3.4.3 TensorFlow 'Model_Main' and Premade models 62
3.4.4 Training schedule 62
3.4.5 Validation 63
3.5 Testing 64
3.5.1 Post training model preparation 64
3.5.2 Confusion matrix analysis 65
3.5.3 Confidence threshold analysis 68
3.6 Statistical analysis 68
4. Results 70
4.1 Learning rate mAP analysis 70
4.2 Learning rate confusion matrix analysis 73
4.3 Learning rate confidence score analysis 74
4.4 Statistical analysis for all LR models 76
4.5 IOU mAP analysis 77
4.6 IOU confusion matrix analysis 81
4.7 IOU confidence score analysis 82
4.8 IOU significance evaluation 83
5. Discussion 85
5.1 Object detection framework significance 86
5.2 Image sets significance 87
5.3 Hyperparameter significance 88
5.4 Statistically reviewing the performance of a whole model 89
5.5 mAP values 91

5.6 Post training performance.. 92
5.7 Potential use as a biomonitoring tool.. 94
5.8 Limitations 97
6. Conclusion... 100
7. References... 103

i. Figures and Tables

Figure 1. Three different chironomid head capsules belonging to the tribes a) Chironomini and b) Tanytarsini, and the subfamily c) Orthocladiinae described by Laurindo da Silva *et al*. (2018). Images were modified from Laurindo da Silva *et al*. (2018). The labelled structures are currently used for visual taxonomic identification of different chironomid genera.

Figure 2. A small neural network structure with three hidden layers where all circles represent nodes, and the interconnecting lines represent the connections and weights.

Figure 3. Intersection Over Union (IOU) diagram showing an example of an actual bounding box overlapping a predicted bounding box.

Figure 4. Image showing a close up of a head capsule mounted on a microscope slide.

Figure 5. The Raspberry Pi (RPI) device used, showing the components for image capture (2-pin push button and RPI camera module) attached to a breadboard.

Figure 6. High power microscope (Leica DM500) and makeshift camera setup connected to Raspberry Pi (RPI).

Figure 7. Image showing a microscope slide mounted with four chironomid specimens obtained from the River Stour, Kent (label indicates that samples were taken the 24th of May 2015, site EA4).

Figure 8. Examples of images of chironomid larvae head capsules captured under the microscope (40X total magnification) using a camera - Left to right: *Cricotopus*, *Eukiefferiella*, and *Rheotanytarsus*. Mandibles, antenna and mentums for each specimen has been labelled to allow a visual comparison of relative sizes and shapes of these structures which are typically used for morphological identification and taxonomic classification.

Figure 9. Example of an image being annotated with bounding boxes (green square).

Figure 10. Head capsules of *Cricotopus* showing differences in quality of the images.

Figure 11. Example of a model showing its prediction and confidence score. A chironomid larva head capsule has been identified belonging to the genus *Rheotanytarsus* with 100% confidence.

Figure 12. A 2x2 confusion matrix typically used for simple classification model evaluation.

Figure 13. A 3x3 confusion matrix as used in this study for the classification of three chironomid genera.

Figure 14. mAP values in ascending order for A) SDD models and B) FR models, each under four different Learning Rates (LR) and three replicates.

Figure 15. Average mAP values for the three replicate runs of each model configuration (SDD and FR).

Figure 16. Best performing models for A) SDD models and B) FR models, each under four different Learning Rates (LR) and three replicates. The models were run for 5000 epochs.

Figure 17. mAP values arranged in ascending order for A) the best SDD models and B) the best FR models, each under three different IOUs (0.5, 0.6 and 0.7) and three replicates.

Figure 18. Average mAP values for SDD and FR models, each under three different IOUs (0.5, 0.6 and 0.7) and three replicates.

Figure 19. mAP values for the best performing for A) SDD and B) FR models, each under three different IOUs (0.5, 0.6 and 0.7) and three replicates run for 5000 epochs.

Figure 20. Images of *Eukiefferiella* incorrectly identified by all models. From left to right, image A identified as *Cricotopus*, and image B identified as *Rheotanytarsus*.

Table 1. Chironomid genera and total number of images of identified specimens used for this study.

Table 2. Image stock totals after removal of test images.

Table 3. Total original and augmented images for image set C.

Table 4. total original and augmented images for image set B.

Table 5. Highlights of the three image sets.

Table 6. Image sets with their train and validation amounts.

Table 7. Model configurations and image sets for comparison.

Table 8. Model configuration, image sets and learning rates.

Table 9. Model configuration, image set and IOU variation.

Table 10. Confusion matrix results for the best performing model configurations.

Table 11. Confidence score analysis for best performing model configurations for SDD and FR models.

Table 12. Nested ANOVA of mAP values with three levels (Framework, Image set and Learning rate).

Table 13. Nested ANOVA of mAP values with two levels (Framework and Learning rate).

Table 14. Results of post-hoc examination for Learning Rate among the two different frameworks (SSD and FR).

Table 15. Confusion matrix analysis for best performing models.

Table 16. Confidence score analysis.

Table 17. Nested ANOVA of mAP values with three levels (Framework, Image Set and IOU).

Table 18. Nested ANOVA of mAP values with two levels (Framework and Image Set).

Table 19. Results of post-hoc examination.

ii. Acknowledgments

Firstly, I would like to thank my wife, Hettie. Your constant support over the last year has allowed me to effectively juggle my work life, this project, and our social lives. You have been there to encourage me on when I felt down and defeated. You have always listened to my ramblings surrounding this project and have offered up words of encouragement when I needed it most. I would then like to thank my main supervisor, Rodrigo Vega. Your mentoring has helped me create and shape this project into something that I am proud of. While it is vastly different to what we had originally planned, you helped create this new project when we had no other choice to move into this discipline due to lockdown restrictions. I am sure it has been an interesting journey for yourself having to quickly learn many new concepts in a science background that was not your own just to help me with my project, to which I will always be grateful for. I would like to also thank Hannan Azhar for coming in as my second supervisor. As a specialist within the world of computer science it has been reassuring to have you on this team to offer your guidance, which was needed on many occasions. I would like to give special mentions to Joe Burman for acting as the chairperson within this project and a final thanks for the technicians at the CCCU research labs, Hannah Shilling, Naomi Beddoe and Jamie Morris. While My final project changed during lockdown which moved me away from the labs, I spent the many months with the original project inside these labs and you all offered your guidance when I was learning how to use new instruments and techniques.

iii. Abstract

This research study introduces a possible new method for the identification of chironomid larvae mounted on microscope slides in the form of an automatic computer-based identification tool using deep learning techniques. Deep learning is becoming an important tool for ecologists where there are advantages and limitations for its use as a rapid biomonitoring tool. Chironomids collected from the River Stour in Kent had their head capsules mounted on microscope slides and images of these were then captured using a Raspberry PI. Using these images, a series of object detection models were created to classify several different chironomid genera. These models were then used to show how different deep learning approaches, focusing on pre-training preparation, could improve the performance of image classification. The model comparisons included two object detection frameworks (Faster-RCNN and SDD frameworks), three balanced image sets (with and without augmentation) and variations of two hyperparameter values (Learning Rate and Intersection Over Union). All models were reported using the standard computer science object detection evaluation protocol, the mean average precision metric. Each model configuration was run three times, to allow for statistical significance evaluation. Additionally, a series of novel post training performance metrics were created examining a model's prediction accuracy and its given confidence value in its prediction choice. The highest mean average precision value achieved was 0.751 by Faster-RCNN. The models highlighted significance between the two object detection frameworks, where the Faster-RCNN framework performed better than SDD framework; however, there was non-significance between the image sets and the hyperparameters values. All models produced similar accuracy results regardless of framework used (between 95.5%-97.7%), however, there were large differences between the confidence examinations, wherein Faster-RCNN produced more confident predictions than SSD. In conclusion, this investigation successfully developed object detection models using SSD and Faster-RCNN to classify between three chironomid genera. As a proof of concept,

this study highlighted that automatic and rapid classification models using deep learning techniques can be applied for the correct taxonomic identification of difficult organisms, like chironomid larvae, further advancing the prospect of using this relatively new field of computer science for ecological research.

1. Introduction

Deep Learning systems are now performing on par or far exceeding human potential and are now being used in many different sectors to either assist or replace human operators (Kermany *et al*. 2018; Raiman *et al*. 2019). Deep learning builds on the concepts of artificial intelligence (AI) and machine learning (ML) with its ability to make predictions based on data it has observed but does so in a manner similar to humans by utilising neural networks which are often compared to the workings of the human brain (Abraham 2005). There are many examples on how deep learning is now routinely used within modern society. Organisations around the globe are using surveillance to recognise human faces when searching for specific individuals (Cárdenas *et al*. 2019), and modern mobile phones and smart home devices are now capable of speech recognition (Ma *et al*. 2019).

The use of automated identification applications in the form of deep learning neural networks are growing in popularity within the life sciences as a way for researchers to alleviate workload and provide assistance to replace the lack of expertise knowledge within a workforce. For instance, Kermany *et al*. (2018) described a Convolution Neural Network (CNN), a form of a deep neural network that works with images, to identify two common and treatable blinding retinal diseases within humans (Wong *et al*. 2014; Varma *et al*. 2014). Several versions of a CNN were created that ranged in identification accuracy between 93% - 100% , while human experts within the field where able to achieve an identification accuracy of 95% Kermany *et al*. (2018) also noted that each year, over 30 million images are taken and used to identify these conditions, and having an automated device that is just as good as experts within the field would be advantageous as the time-frame and manpower to manually process this large collection of images would be enormous.

Bondi *et al*. (2018) described an object detection system, which builds on top of techniques used in CNNs, that could automatically detect and identify poachers or high-risk animals in real-time when used with a video feed. By using an Unmanned Arial Vehicle (UAV) at night-time, the object detection system could alert a human operator almost instantaneously of any poaching activities and could cover large areas quickly that would take a single human much longer to search. While these UAVs can fly for extensive amounts of time while sending a live video feedback unaided, the monitoring of these systems traditionally has to be operated by a human user for up to 12 hours at a time. Thus, having an automated system that can both identify both poachers and specific animals, and then alert operators of a potential target while the operator is not at the monitoring station increases the odds that poachers can be stopped, and endangered animals can be protected. Poaching has been on the rise for many years now and the monetary resources involved in the protection of vulnerable species is increasing with it so there is demand in techniques that can alleviate these costs (Naidoo *et al*. 2016).

One important achievement incorporating deep learning in the form of an automated identification tool is the mobile device application called 'PlantSnap' (PlantSnap 2020). This application states that it can identify and distinguish between over 620,000 different plants species and their respective variants from around the globe, around 90% of all described species of plants, and can do so with great ease to the user (PlantSnap 2020). Plant identification using PlantSnap can be achieved by the user by simply uploading an image of a leaf and/or the flower over the application which is then identified over a server with the results sent back to the user. While the technology that has led to the creation of this system is more closely guarded than the other research projects described thus far, the fact that it has been translated into 37 languages and downloaded by over 30 million users globally is an amazing feat of the sharing of scientific knowledge. This and the other systems described have shown the potential of automated identification systems as a way of increasing the work output of experts in various life science fields and have shown that nonexperts are willing to use the knowledge that these systems can provide.

For this investigation, a deep learning identification tool in the form of an object detection model has been proposed to correctly classify several chironomid genera in larvae form that have been previously mounted on microscope slides. Chironomid larvae are increasingly being used as bioindicators in water systems and there is interest in their use to monitor freshwater systems (Rawal *et al*. 2018). The identification of chironomid larvae has traditionally been through the use of visual methods by observing the animal's morphology. This would typically be conducted by an expert taxonomist using identification keys and a microscope; however, these methods can be very time consuming and are prone to misidentifications (Haase *et al*. 1998; Wiederholm 1983). A deep learning-based identification tool for chironomid larvae could potentially allow a wide range of users to reliably produce correct identification in a quick and easy to use manner. This would ultimately become a valuable tool within biomonitoring and when performing environmental assessments of aquatic systems.

1.1 Freshwater ecosystems

Freshwater ecosystems can be found on all continents within the world but are more frequent in North America, Europe and Asia (Siberia). Only 3% of the world's water is fresh water, with a large portion of this held within the polar icecaps (Gerbeaux *et al*. 2018). The majority of living organisms relying on fresh water as a form of sustenance, and the ecosystems surrounding these waters provide habitats to a large range of species, highlighting the importance of maintaining these systems (Hughes 2019). There are a range of fresh waters both natural and manmade such as, but not limited to, rivers, streams, lakes, marshes, chalk streams and reservoirs. Despite their importance to providing sustenance to almost all life, and to supporting the surrounding habitats, freshwater ecosystems are in danger of degradation due to anthropogenic interference with the main contributing factors being pollution, climate change and habitat transformation (Cao *et al*. 2009). This degradation is having a knock-on effect to the organisms that depend on these ecosystems. For instance, the global decrease in macroinvertebrate populations and the decrease in macroinvertebrate species diversity is being linked directly to this anthropogenic interference (Živić *et al*. 2014; Costa *et al*. 2020) which in turn is having a wider knock-on effect to the ecosystem in which these organisms inhabit (Cao *et al*. 2018). To prevent this, ecologists and conservationists can use biomonitoring techniques to assess these ecosystems in their current and ongoing condition. For the biomonitoring of all aquatic ecosystems, the community structure of benthic macroinvertebrates can be used and can include the presence or lack of certain species (Sharma and Chowdhary 2011; Uherek and Pinto Gouveia 2014; Zeybek 2016; Costa *et al*. 2020). Cao *et al*. (2018) proposed that a lack of expected benthic macroinvertebrate communities and the presence of certain ubiquitous species, particularly those considered pollutant tolerant (*i.e.* sludge-worms, *Tubifex tubifex*), could be used as indicators to show how the degradation of river water is affected by municipal waste. This type of approach is routinely used by researchers and governing bodies across the globe to assess the quality of water systems and the surrounding ecosystems. Biggs *et al*. (2000), commissioned by the United Kingdom (UK) Environmental Agency, justified the use of benthic macroinvertebrates, along with macrophytes and their presence within different water systems across the UK, as bioindicators and proposed how the use of these can be used to assess the water condition of ponds, lakes and rivers as well as the condition of the banks of these water systems. This type of biomonitoring assessments has also been applied in other parts of the world, *i.e.* South America (Ríos-Touma 2014), Africa (Beyene *et al*. 2009) and Asia (Morse *et al*. 2007). While there is a selection of species that can contribute to these assessments (*i.e.* stone fly nymphs, oligochaetes, caddisfly larvae), chironomid larvae (Insecta: Diptera: Chironomidae) are considered ideal candidates for such assessments (García and Añón Suárez 2007; Rawal *et al*. 2018).

1.2 Chironomids

Chironomids, also known as ‘non-biting midges’ or ‘bloodworms’ (when in their larval stage), are one of the most abundant and species-rich benthic macroinvertebrates in freshwater ecosystems (Pinder 1986; Ferrington 2008; Nicacio and Juen 2015). Chironomids are suggested to make up 50% of the total benthic macroinvertebrate population within their respective habitats (Nadjla *et al*. 2013). They are found in almost all freshwater ecosystems including lakes, ponds, swamps, streams and rivers, and can also be found within isolated habitats such as tree stumps, and man-made water ways like flood-prevention drainages (Frouz *et al*. 2003; Pinder 1995). There are an estimated 600 species found within the United Kingdom and an estimated 20,000 species worldwide (Ferrington 2008). Some species of chironomids can live in a variety of aquatic systems tolerating a range of environmental conditions including pH, salinity, temperature and sediments, while others require very specific conditions (Lencioni *et al*. 2012), and some can even be found in aquatic systems considered polluted and inhabitable for the majority of other species (Luoto 2011). This has led to the exploration of chironomids as bioindicators for the general condition of aquatic ecosystems (Lencioni *et al*. 2012), however, they can also be used for more specific and streamlined assessments. For instance, Orendt (1999) created a technical water monitoring method that provided an acidity assessment for water systems, where the pH tolerance of 25 species of chironomids were identified by their presence within several bodies of water with known pH. In this study, a more diverse range of species was found in waters with a low pH while there were few species that could tolerate a higher pH, and by correlating the presence of species with their tolerance the acidity condition of a body of water could be ascertained. The study of chironomids also benefits from their use as a paleoclimatology and paleoenvironmental assessment tool (Nazarova *et al*. 2008; Nazarova *et al*. 2018; Lang *et al*. 2018). Here, chironomid fossil records, particularly those of their head capsules found in water sediment and how deep these are found, are linked to historic temperatures. The remains are taxonomically identified and linked with known paleoclimatic conditions at that point in time to generate a score-based biotic index.

The use of chironomid larvae for biomonitoring and paleoclimatically assessments depends on the correct identification to specific taxonomic levels. However, one of the main issues with the identification of chironomid larvae is their minute size. Chironomid larvae are typically millimetres in length which makes it difficult to accurately identify them below the taxonomical classification of family (Chironomidae) without expert taxonomic knowledge or the means of molecular-based procedures. At present, the two main recognised methods for correct identification and taxonomic classification are through visual methods using classical taxonomic keys and using molecular methods such as DNA barcoding.

1.3 Visual identification of chironomids

Identification of chironomids has traditionally been carried out using specimen morphology. This is typically done using the knowledge of an expert taxonomist or entomologist (*e.g.* Vega *et al.* 2021). For the analysis of chironomids by visual inspection, the use of taxonomic keys and a microscope is needed. The head capsules of chironomid larvae are primarily used for this; however, the final segments of the abdomen can also be included as these do show differences between species. Head capsule and final segments of abdomen need to be carefully removed from euthanised specimens and then mounted on microscope slides using a method outlined by Beckett and Lewis (1982). With the slides mounted, these can then be examined under a high-powered microscope where specific features can be used to determine to which taxonomy group they belong to. For example, Laurindo da Silva *et al*. (2018) presented and described the head capsules of three different members of the family

Chironomidae, the tribes Chironomini and Tanytarsini, and the subfamily Orthocladiinae (Fig. 1).

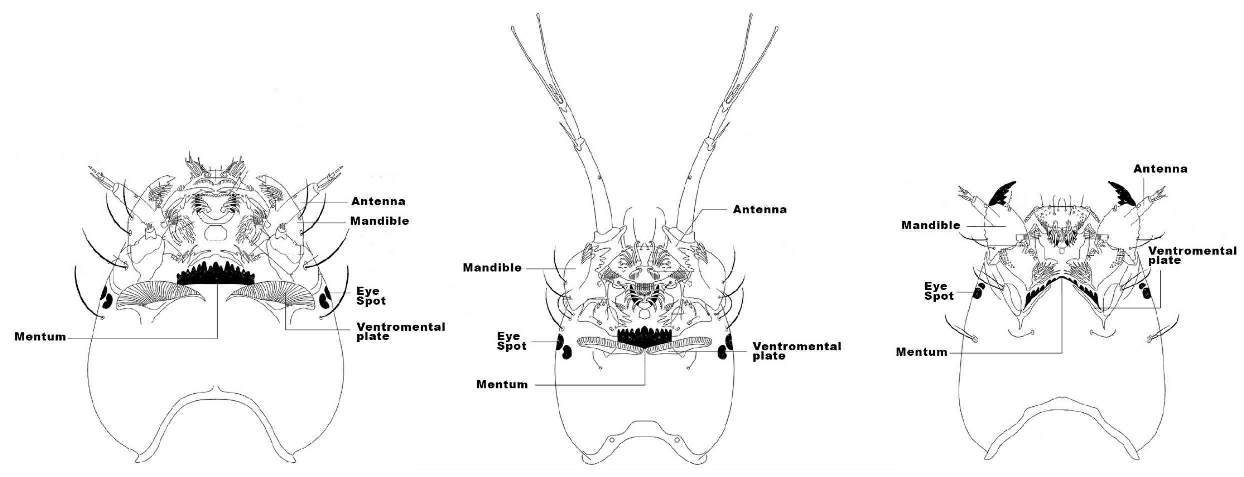


Figure 1. Three different chironomid head capsules belonging to the tribes a) Chironomini and b) Tanytarsini, and the subfamily c) Orthocladiinae described by Laurindo da Silva *et al*. (2018). Images were modified from Laurindo da Silva *et al*. (2018). The labelled structures are currently used for visual taxonomic identification of different chironomid genera.

While they share many features such as mentums, ventromental plates, eye spots, mandibles, and antennae, they are all different in shape, length, and position on each head capsule. This variation of morphological features between each species and the large number of estimated species (20,000 worldwide) highlights why it is extremely difficult to accurately identify chironomids to beyond the taxonomy level of family, even by experts within the field (Wiederholm 1983; Haase *et al.* 1998).

1.4 Identification through DNA barcoding

At present, one of the leading techniques for identifying and classifying species is by molecular means, specifically by sequencing particular DNA sequences *(i.e.* molecular markers) and comparing them against a known set of previously taxonomically identified vouchers specimens with DNA data (*i.e.* a DNA library). This comparison is done using computational methods in biology (bioinformatics) across different taxonomic levels, a process known as DNA barcoding (Valentini *et al.* 2009). The currently considered ideal gene for DNA barcoding animals, and specifically insects, is the mitochondrially encoded Cytochrome C Oxidase Subunit 1 (MT-CO1). Mitochondria have its own unique DNA (mtDNA) separate from the nucleoid DNA (nDNA) (Meier *et al*. 2006), it is maternally inherited, and it is passed down the lineage unaltered through processes like crossing over and recombination, which occur in nDNA. While DNA barcoding has been used successfully in many instances, it has its own limitations (Reynaud *et al*. 2015). In particular, there are costs associated with DNA extraction, sequencing procedures and laboratory equipment needed to achieve the molecular identification of organism, plus the time in which DNA barcoding can be completed. Moreover, DNA barcoding requires the use of a DNA library. This is a set of individuals within the taxonomic group of interest which have been previously identified by traditional morphological means and DNA barcoded, and which serves as a reference to which new specimens can be matched against (Benson *et al*. 2015).

DNA barcoding could bring the total cost of a single individual sample to over £7 (Stein *et al*. 2014). Although the costs can be minimized by the use of more cost-effective protocols like multiplexing and more economical reagents for DNA extraction, PCR, PCR purification and DNA sequencing, the total cost of a large-scale ecological biodiversity survey would incur much larger costs (Shendure *et al*. 2017). Moreover, the DNA barcoding process is time consuming and often not automated, requiring the researcher to spend time in the laboratory carrying out each step of the protocol. Time-saving protocols can be used but required expensive laboratory equipment like automated DNA extraction systems (*e.g.* QIACube Connect, QIAGEN) (Valentini *et al.* 2009). Also, the final DNA product needs to be sequenced, a process that is usually outsourced, meaning that the identification of a single sample could take several days. The time and effort incurred would be increased dramatically when processing a large batch of specimens, such as those being utilised by this investigation.

1.5 eDNA metabarcoding

Environmental DNA (eDNA) metabarcoding is a technique that can detect multiple organisms from one single environmental source, for example bodies of water, sediments, soils and air, using genetic markers (Ruppert *et al*. 2019). Bodies of water, sediments, soils or air will contain various sources of DNA including, but not limited to, skin, eggs, blood, faeces, urine and shedding. From these samples, genetic markers from a range of organisms can be amplified, sequenced and identified against a DNA library. The organisms identified can then be used to provide ecological assessments which can be done in a rapid time frame compared to singular organism identification via traditional DNA barcoding or morphological identification (Deiner *et al*. 2015). Because taxonomic data can be easily obtained using eDNA, such as from a sample of pond water, habitats do not need to be disturbed as when using kick-sampling methods for benthic organisms (Goldberg *et al*. 2016). However, eDNA methods cannot provide information one would expect by investigating a single individual, such as body conditions or sex ratios (Deiner *et al*. 2017). Again, much like DNA barcoding, eDNA methods rely on a variety of scientific instruments and chemical reagents that come at a cost, and sufficient bioinformatic knowledge is required to perform all operations (Shendure *et al*. 2017).

1.6 Automated identification of organisms

The automatic detection, identification and classification of organisms has been in development since around the 1970's where the growth of various microorganisms would be photographed at specific intervals when placed upon different substrates. The images of their growth patterns could be used to identify which specific microorganism was present using custom built computer systems and programmes (Aldridge *et al*. 1977). Stager and Davies (1992) reviewed the origins of these early methods and suggested that these produced mixed and often disputed results in terms of performance and evaluation protocols. Since then, a range of automated techniques have been devised to alleviate the workload of investigations within the life sciences.

Non-computer based electronic devices such as camera traps are able to automatically capture images when an object passes through a motion sensor and are routinely being used within biodiversity assessments; however, camera trapping requires a human operator to manually examine each image individually which can range into millions of images, a daunting workload for the user (Fegraus *et al*. 2011). Traditional computers and electronic devices are unable to process these sources of information to automatically detect, identify and classify in their current format without the aid of specialised equipment, programmes, and computer-based techniques, and thus the subject known as computer vision was created.

1.7 Computer Vision

Computer vision (CV) is the broad subject of creating computer programs that can extract information from digital images and video. There are a multitude of techniques that can achieve CV which include image classification, where an image is classified as a single desired subject (*e.g.* a dog vs a cat), or object segmentation, where an image is separated into specific desired segments (*e.g.* the sky and ground) (Khan *et al*. 2020). CV involves the conversion of digital image pixels values to binary values using a set of programs which can then be read by computers where patterns within the binary values are used to create recognisable features that can eventually be combined to recognise specific subjects within images, whether they are singular, part of groups, their position within images and whether they are 3D or 2D (Ikeuchi 2014; Lin *et al*. 2015). To achieve this, a computer program, more commonly referred to as a model, needs to be created. This model needs to go through a self-teaching process known as training. Data in the form of images are fed into the model and the internal coding structure of the model, referred to as the architecture, instructs the model how to process the data held within the image pixels (Szeliski, 2010) (*e.g.* if the model was intended to recognise a dog within an image, a model would be fed images of a dog, go through a training process where the architecture is designed to allow the model to self-teach the recognition of specific features within the images that make up a dog, and then finally make a prediction based on these learned features).

For instance, Yu *et al*. (2013) created a computer-based system that was able to automatically detect, identify and classify over 18 species of animal held within over 7,000 images captured from camera traps with an accuracy of >90%, and pointed out that this method removed a large number of images that would be considered false positives as camera traps can be incorrectly triggered when nonanimal objects pass through the motion sensors. Hoque *et al*. (2011) used a computer-based system to automatically identify individual great crested newts (*Triturus cristatus*) from markings on their back by using a collection of 30 images and pointed out that being able to identify individuals using visual zoometrics would alleviate the need to utilise more traditional invasive methods such as amputation or tagging and automatization of the process could potentially increase the accuracy and speed of identification. These methods used computer-based programmes that are able to extract information from sources such as images or live video streams.

While some CV models can produce identification and classification results that are comparable to human identification (Kermany *et al*. 2018), there are still limitations that affect the subject as a whole. For instance, to create a CV model, extensive coding knowledge and computer literacy is required (Bebis *et al*. 2003). There are a variety of computer coding languages that the CV models can be written in (*e.g.* Python or C++) (TensorFlow 2020c), and there are a variety of operating systems that the computer hosting the CV model can be run on (*e.g.* Linux or Windows). Therefore, computer coding languages and operating systems need to be understood to successfully operate a model. The CV models are also resource-intensive when being trained, where they require the computation power of many high-end computer components, in particular the Graphical Processing Unit (GPU), which all come at a high monetary cost (Coates *et al*. 2009). However, once a CV model is created, the computation power required to run a finished model is drastically lowered. Most high-end mobile devices at the time of writing this investigation are able to host and run a slimline version of finished model (Howard *et al*. 2017, 2019; Sandler *et al*. 2018). Finished models can even be hosted by a powerful device and accessed by multiple, less competent devices via a server. As mentioned, the application PlantSnap utilises this method to allow its 30 million users identify a range of plant species with almost any mobile device with a camera feed (PlantSnap 2020).

1.8 Neural Networks

Neural networks (NN) consist of nodes that are interconnected and are built in layers, typically referred to as input, hidden and output layers (Fig. 2) (Abraham 2005). Information in numerical form is inserted into the input layers and will lead to an output layer in numerical form that corresponds to a prediction while passing through hidden layers. The hidden layers will transform the original input value by multiplying these by a value know as weights. While these weights start of as random numbers, they will be updated during each pass through after the output has been identified as a correct or incorrect prediction. These weights will continue to be updated until the value is optimised to produce a correct output value. NN can range in sizes for both the number of individual nodes within each payer and the number of hidden layers.

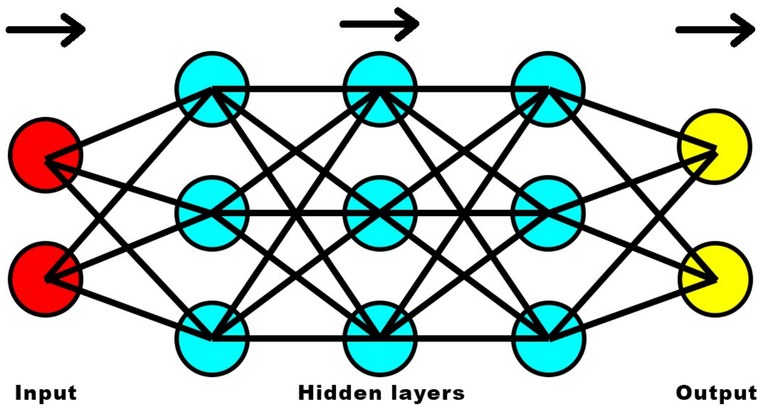


Figure 2. A small neural network structure with three hidden layers where all circles represent nodes, and the interconnecting lines represent the connections and weights.

1.9 Convolution neural networks

When processing images, traditional NN would take an entire images pixels value and attempted to pass this through the network at a single instance (Amer and Maul 2019). For example, a colour image with 3x colour channels (Red, Green, Blue) of 300x300 (Height x Width) pixel resolution would equate to 3x300x300 or 270,000 individual inputs, where each pixel within each colour channel can have a pixel intensity value of 0-255. This large number of inputs would then require their own weights simultaneously updated during the training process, which would become unmanageable especially for large images (Yamashita *et al*. 2018; Amer and Maul 2019; Batmaz *et al*. 2019). Traditional NN also have issues with the translocation of subjects within an image. For example, if a NN has learned that the position of a subject is at the top right of an image, and then it is presented with an image where the subject is not within this area, the weights would not be associated with the subject in this area and would subsequently misidentify (Al-Saffar *et al*. 2017; Yamashita *et al*. 2018).

Thus, a different technique called a Convolution Neural Networks (CNN) has been created that can classify subjects despite its location within an image. Although a CNN still contains a neural network, they include other layers that prevent the issues that would arise from the use of traditional NN (Al-Saffar *et al*. 2017; Khan *et al*. 2020; Yamashita *et al*. 2018). The first layers which are convolution layers pass numerous boxes known as filters over the images from left to right, top to bottom. These filters, which are size defined by the user, are generally much smaller than the image (*i.e.* 3x3 pixel resolution) and are used to search for patterns within the values that are being passed over. The CNN designates its own patterns to recognise based on the data fed in and assigns a specific pattern to each of the filters. These patterns will relate to features within the object that the CNN is being trained to detect starting of as simple features such as lines, corners and bends which are referred to as lower order features. These features are then put together in pooling layers to create feature maps that store all features. From this, more complex features such as specific sections of an object are then constructed from the lower order features (*e.g.* if a model was being trained to detect human faces, then the complex features could be items such as the eyes, ears or mouth). All these features would be pooled together again to create feature maps that make up the final predictions of a whole object. This sliding filter allows for a model to learn features without focusing on where these features are on an image (Yamashita *et al*. 2018).

There are a multitude of different CNN architectures that use slightly different methods to achieve the desire results, for example Visual Geometry Group (VGG), Residual Neural Network (ResNet) and Inception (Khan *et al*. 2020). Gyanendra *et al.* (2018) and Favorskaya and Pakhirka (2019) report CNN systems which utilised the VGG architecture. These were used to classify images from camera trap of animals within a woodland area. They report that their system can positively classify with an accuracy of 91.4% and 94.1% respectively.

1.10 Image Classification and Object detection

Objects within images can be classified into specific predetermined categories using a technique called image classification (Rawat *et al*. 2017). Here, a single object can be identified within an image by predicting the class that the image would be assigned to, referred to as a label, from a set selection of categories. This typically only works with one object per image. There are several different techniques that can be used to achieve this, such as K-nearest neighbour, support vector machine, multi-layer perception and CNN. CNN has become the optimum and the most used method for achieving image classification since it was first showcased in the 2012 ImageNet large scale visual recognition challenge (ILSVRC), where it subsequently took 1st place within the competition (Krizhevsky *et al*. 2012). The life sciences have begun to utilise image classification. For instance, the CNN described by Kermany *et al*. (2018) wherein images were used to distinguish between two blinding retinal diseases, and the CNN described by Yu *et al*. (2013) wherein images were taken from camera traps and used to identify 18 individual animals.

Building on from image classification is a technique called Object detection. Object detection is the name given to computer vision techniques that can localise the object within an image, can classify objects within an image, , can classify multiple objects within a single image or in real time over a live video feed (Huang *et al*. 2017; Nam and Hung 2018; Janahiraman and Subuhan 2019; Zhao *et al*. 2019; Hall *et al*. 2020). There are numerous object detection frameworks, but two that appear more popular among researchers being the Single Shot Detection (SDD) framework and the Faster-Region-based CNN (Faster-RCNN) framework (Arcos-Garcia *et al*. 2018; Nam and Hung 2018; Janahiraman and Subuhan 2019; Bose and Kumar 2020). The object detection framework is built on top of a CNN which has allowed for a multitude of possible model combinations (*e.g.* SSD_inception, Faster-RCNN_VGG, SSD_ResNet) (Zhao *et al*. 2019; Zhang *et al*. 2019).

SSD is an object detection framework first described by Lui *et al*. (2016). SDD works in a single step where the CNN feed-forwards its learned features to the SSD framework, and then places a grid over an image where each grid space will have a large amount of default possible locations, referred to as bounding boxes or anchors. In SDD, each grid uses the feature maps from the CNN and assigns the best anchor to predict objects and their location within the image. First described by Ren *et al*. (2016), Faster-RCNN uses a two-step approach for object detection. Built on top of a CNN where the features are learned, are then passed to two separate functions. One is a regional proposal network that uses a sliding window approach, but the window has its own set of anchors. These anchors will use the feature map to detect any subjects, however, it only indicates that there is an object within the location and does not define a class to this location. The second function is the one that defines the class. The Faster-RCNN frameworks are commonly reported as being more accurate than the SSD (Arcos-Garcia *et al*. 2018; Janahiraman and Subuhan 2019; Bose and Kumar 2020). Despite its lower performance potential, SSD is regarded as performing at greater speeds than the Faster-RCNN due to its single action, and it is suggested that the inferior speed of Faster-RCNN prevents it from being used where near instant identification is required, such as the detection needed for autonomous navigation (Lui *et al*. 2016). The SSD framework also benefits from the architecture's abilities to become streamlined when paired with the CNN MobileNet architecture, allowing the creation and operation of trained models that require extensively less computations, which the Faster-RCNN frameworks is currently unable to achieve (Howard *et al*. 2017; Sandler *et al*. 2017; Howard *et al*. 2019).

There are other notable object detection frameworks. 'You only look once' (YOLO), is another framework that utilises a CNN and performs both classification and localisation in the same instance rather than separating the two tasks, again performing at high speeds while trading this for accuracy potential (Redmon *et al.* 2015). Histogram of orientation gradients, more commonly referred to as HOG, is a framework that does not rely on the use of a CNN. HOG framework detects edges with their orientation and gradient and does so in localised sections, and a histogram of each section is created to observe for frequency of continuous data (Rybski *et al.* 2010).

1.11 Evaluation of Object detection models

To evaluate the performance of an object detection model several metrics have been designed over the years (Mariano *et al*. 2002). These include metrics such as Average fragmentation, which will detect and highlight if multiple detections occur for the same object and localised object count recall, which will detect and highlight how much of a detection is covered by the prediction box. However, current research almost exclusively uses a metric known as the mean average precision score (mAP). The mAP metric is used to evaluate different object detection models, and it was created so that different framework and CNN combinations, operating in slightly different ways to one another, could all be evaluated under the same metric system with a value produced between 0 – 1 (Girshick *et al*. 2014; Arcos-Garcia *et al*. 2018; Shi *et al*. 2019). However, this value is worked out slightly differently depending on what evaluation protocol system is used, and there have been several evolving iterations that have been created, updated, and adopted by the scientific community (Hall *et al*. 2020). For instance, the Pascal-VOC protocol first described in 2007 and ran until 2012 (Everingham *et al*. 2009) has since been replaced by the MSCOCO (Microsoft Common Objects in Context) protocol which is now the most commonly used evaluation protocol (Lin *et al*. 2015; Hall *et al*. 2020). Janahiraman and Subuhan (2019) compared both the SSD and Faster-RCNN frameworks and reported MSCOCO mAP values of 0.220 and 0.280, respectively. Ren *et al*. (2016) reported a MSCOCO mAP of 0.219 when first reporting the Faster-RCNN framework. Huang *et al*. (2016) reported a MSCOCO mAP of 0.413 for as custom object detection framework. However, some researchers still report the Pascal-VOC and some report both the Pascal-VOC and MSCOCO protocols (Arcos-Garcia *et al*. 2018). The MSCOCO metric is reported to be more stringent than the Pascal-VOC metric (Huang *et al*. 2016). For instance, Lui *et al.* (2016) report a Pascal-VOC value of 0.822 while their MSCOCO value is 0.288.

The MSCOCO mAP metric is calculated by creating recall-precision curves at 10 different Intersection Over Union (IOU) thresholds. The IOU threshold is the minimum area allowed between the overlap of an object detection's prediction of where an object is to where it is within an image with a value between 0 – 1 (Bose and Kumar 2020). Both the prediction and actual image location are defined to location by a surrounding box which is known as the bounding box. Equation 1 shows the IOU formula where the *Union* is defined as the accumulation of areas of both the actual and predicted bounding boxes that are not overlapped, while the *Overlap* is the area where both bounding boxes overlap (Xia *et al*. 2018; Chudzik *et al*. 2020) (Fig 3.).

$$IOU = \frac{Area\ of\ OverLap}{Area\ of\ Union} \tag{1}$$

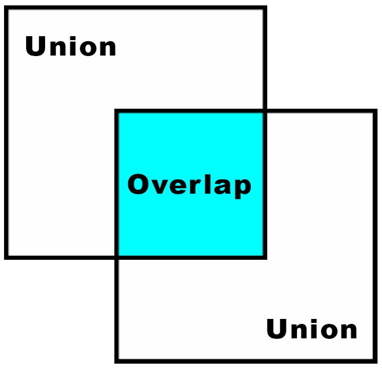


Figure 3. Intersection Over Union (IOU) diagram showing an example of an actual bounding box overlapping a predicted bounding box.

These minimum thresholds start of at 0.5 and increase at 0.05 intervals until they reach 0.95. The values obtained from each area under the precision-recall curve at each IOU value are then averaged out to create an Average Precision Value (AP). This AP is worked out for each individual subject that is being trained for detection. Each of these individual AP values are then averaged again to produce the final mAP score. It is typical to report the mAP values only and rank these with the model producing the highest value being regarded as the best performing model out of any other within a sampled group (Girshick *et al*. 2014; Liu *et al*. 2016; Ren *et al*. 2016; Ren *et al*. 2017; Arcos-Garcia *et al*. 2018; Shi *et al*. 2019; Bose and Kumar 2020).

1.12 TensorFlow and TensorBoard

Training an object detection model requires the combination of many different computation components made up of both hardware and software (Coates *et al*. 2009), and to allow all of these to work in unison and achieve the final resulting model, the use of a deep learning library is generally utilised (Goldsborough 2016; Pang *et al*. 2019). These libraries contain a range of toolkits and programs that are designed for the easy devolvement and deployment of machine learning and deep leaning models. There are several deep learning models in circulation, both paid-for and open-sourced, which include PyTorch created by Facebook, CNTK created by Microsoft, Apache MXNet created by Amazon, and TensorFlow created by Google. While all can perform object detection, the use of TensorFlow is a popular option with researchers and hobbyists due to its ease of use and community that provide guidance for its use (Al-Azzo *et al*. 2018; Janahiraman and Subuhan 2019; Chulu *et al*. 2019).

The TensorFlow deep learning library is an open-sourced deep learning library that is designed to allow machine learning models to be deployed with ease to the user by working behind the scenes to connect a range of programs and hardware, such as the object detection frameworks, while harvesting the computational power of a workstation (Goldsborough 2016; Pang *et al*. 2019). Within the library is the TensorFlow Python API which is designed to work with a variety of deep learning models, including the object detection models, which can then be trained and deployed using the Python programming language. TensorFlow allows the monitoring of model training through an included tool called TensorBoard. This produces a set of graphs that can illustrate how a model is evolving at defined intervals. TensorBoard is designed to work in unison with several evaluation protocols, including the MSCOCO protocols and its version of the mAP metric.

1.13 Finetuning an object detection model

To achieve the best possible mAP score, a model can be optimised and finetuned using a variety of techniques before the start of the training process (Xia *et al*. 2018; Probst *et al*. 2019). Within machine learning which encompasses deep learning, a hyperparameter is an internal configuration value or specific function that is set from within the coded architecture itself (Bergstra and Bengio 2012; Probst *et al*. 2019). While some object detection models will share some basic hyperparameters such as the batch size (the number of images that is examined that are examined at any one instance) or epochs (how many times the total number of images are examined), some models will have individual hyperparameters that are individual to that specific architecture (*e.g.* the Faster-RCNN framework will contain hypermeters for its two separate functions). Some hyperparameters can be calculated by understanding the type of data and how much of it will be involved during the training phases (*i.e.* the batch size depends more on the capacity of the workstation and could be worked out prior to training). However, other hyperparameters cannot be calculated beforehand, therefore the only way to find optimum values is through trial-and-error methods or by reviewing the work of other researchers. Bergstra and Bengio (2012) highlighted two trial-and-error methods: one is a systematic method called grid searching where a large portion of hyperparameter values is systematically tested at set intervals, moving through a large portion, and sometimes all possible choices; the other is a random selection of values. They found that randomly selecting values was the optimum method and proposed that this method produced models with equal optimisation to that of grid searching and did so within much greater timeframes than using grid searching.

Chudzik *et al*. (2020) suggested that one of the most important hyperparameters is the Learning Rate (LR) and proposed how there is no universal value for this. The LR is a value that the weights within the neural network are updated by, and these weights will be unique to each model meaning that the only way to optimise this is through a trial-and-error process. Xia *et al*. (2018) trialled four LRs; 0.01, 0.001, 0.005 and 0.0005, and found that the learning rate of 0.001 produced the best results in terms of mAP values obtained when attempting to optimise the performance of a CNN model with a 1.4% increase in the performance of the worst performing LR of 0.01. Chulu *et al*. (2019) trialled several LRs and found 0.01 and 0.001 to be the optimal values for two different epoch rates in terms of mAP values obtained when attempting to optimise the performance of InceptionV3. Xia *et al*. (2018) and Chudzik *et al*. (2020) both suggested that having a LR too high causes the weights to overshoot its optimum value, whereas having a LR too small will not allow the weights to keep up with the desired development.

The IOU threshold is another hyperparameter that is utilised to optimise model performance. The IOU threshold is calculated in the same way that was shown in equation 1, however, this IOU threshold is used during the training stages, rather than within evaluation protocols. The optimum IOU can be dependent on the size of the object that is desired to be detected meaning that not all of a defined bounding box encompassing a subject need to be within a predicted bounding box to be correctly classified. Yan *et al*. (2019) suggested that large objects work well with large IOU thresholds, but smaller objects tend to do worse as there is more chance that there will be false positives contained around the objects. Having an IOU that is too small could allow too few features to be picked up, which would mean not enough feature information would be available to make a suitable correct prediction. Having a threshold that is too high can result in a decrease of computer performance by making it difficult for the actual and predictive bounding boxes to align under the threshold. Redmon and

Farhadi (2018) described how a high IOU threshold significantly decreased the performance of YOLO v3, another object detection framework. Redmon and Farhadi (2018), and Xia *et al*. (2018) and Bose and Kumar (2020) proposed that an IOU value of 0.5 produced the best performing models in terms of their respective mAP values.

1.14 Quality of data

The optimised performance of machine learning models depends on the quality of the data fed to it for training (Obermeyer and Emanuel 2016). To train an object detection model, a selection of data in the form of images that depict the required subject is required. Each individual subject is referred to as a class. For example, if an object detection model was designed to classify between dogs, cats and humans, that would mean that it would contain three classes and each class would be trained using images that portray each of the three objects. At present there is no definitive set number of images required to train an object detection model, however, it is generally recommended to use both a large set of images per class as well as many images in total overall. This is suggested to be because a model will not have enough data to correctly identify specific features within images and will identify random collections of pixels that are found shared among the images (Srivastava *et al.* 2014; Wong *et al.* 2016; Hussain *et al.* 2017; Shijie *et al.* 2017; Valan *et al.* 2019).

There are also suggestions that an imbalance between the total images within each specific class could result in a drop in a model's potential, as this would produce skewed results where one or more classes and their features are trained more robustly than other classes and their features (Yue 2017; Batista *et al*. 2016). To counter this, there are two similar techniques that can be utilised known as up-sampling and down-sampling (Batista *et al*. 2016; Pickek *et al*. 2016). To down-sample, the number of images within the smallest class, known as the minority class, would be taken to down sample all other classes until each had the same number of images as the minority class, ultimately balancing the classes. While this does not require the addition of new images, it is possible to use randomisation techniques to avoid biases when downsizing the groups. Up-sampling is where the class with the highest number of images is consider the benchmark, known as the majority class, meaning that the minority classes would need to be increased to the same value. This approach could also be extended if a number of images exceeding the majority class was desired and thus all classes would have to have additional images. However, to achieve up-sampling methods of acquiring additional data need to be considered, for example by harvesting new images from sources such as the web which is not always possible. Typically, an up-sampling approach would be paired with data augmentation. This is the creation of synthetic images by creating copies of the original images and then altering their appearance.

Within the literature there is an inconsistency between the different number of images used within their respective models. Some studies have used extremely large numbers of images to train the models. Krizhevsky *et al.* (2012) used 1.2 million total images equally spread over 1000 classes which equates to approximately 1,200 for each class when developing ‘Alexnet’, an award-winning CNN architecture. The mobile device application PlantSnap used over 100 million total images to train its 610,000+ classes although make no reference to how these are divided between classes (PlantSnap 2020). Others have used much smaller numbers of images with balanced image sets. Xia *et al*. (2018) used 4800 total images, spread equally across 24 classes, equating to approximately 200 images per class. However, others have used both small and imbalanced image sets. Chulu *et al.* (2019) used 780 and 400 respectively for a two-class model. Patel and Bhatt (2019) used 319, 131 and 143 images for a three-class model.

The size of each individual image can affect a model’s optimum outcome. There is no current set image resolution for optimum performance that is required when training an object detection model, however, it is recommended that images be of a low-resolution quality (Sabottke and Spieler 2020). An image with a high resolution would require considerably more calculations within the internal algorithms that are involved in the training. This would in turn require extensive finetuning of hyperparameters and would require the computing power of an advanced workstation to handle the resulting workload. However, previous studies show that images with higher resolution would ultimately produce more accurate results if the resources needed to utilise these large images were available; for instance, Lui *et al.* (2016) reported that image resolutions between 300x300 and 512x512 pixels worked optimally when creating and reporting the original SSD object detection framework.

What is present in the images also can affect a model's optimum outcome. An image of interest would contain both a subject of interest and a background. A model's ability to classify a subject can be directly tied in to whether the model can also detect what kind of background the subject of interest is set in. Xiao *et al.* (2020) showed within their research that a model's performance can be skewed by putting desired subjects on backgrounds unfamiliar to what the model was taught on. They suggest that this can be both a disadvantage and advantageous. Some subjects could potentially skew a model training or chance to classify correctly if they are within unfamiliar backgrounds. However, if a subject is going to be found within a background exclusively through both the training phase and post-training detection phase, this would ultimately result in a higher chance for a positive detection.

1.15 Transfer learning

There is a final technique known as transfer learning that can be used to help optimise the performance potential of models pre-training and it can be used when working with models that have image sets with a small number of images. As mentioned, while there is no final number of images required to train a model, to train it specifically from the beginning an extremely large number of images would be required to allow a neural network to learn enough features to correctly classify a subject (Kermany *et al*. 2018). This can be troublesome as the hours required to collect, organise and label these images will be extremely high. For instance, the team that created the MSCOCO mAP metric also created a large dataset consisting of 330,00 images where 1.5 million objects were labelled (Lin *et al*. 2015). They created this set so that researchers could test new model architectures while using this image set and evaluation metric in unison as a benchmark. During its creation, Lin *et al.* (2015) mentioned that it took over 70,000 combined hours to prepare and organise the images, and that it took the combined efforts of over 1000 people to achieve this. This timeframe and team effort would be unachievable for the majority of investigations. However, rather than being faced with this daunting task of collecting and organising images, models that have been trained on these large image sets can be used as steppingstones or checkpoint for models that have much smaller image sets.

The checkpoint model trained on large image sets neural networks consists of layers. These layers move from the beginning where it started off learning low order features, to the last layers where neural networks learn higher order features, and finally combine these to create whole objects. The last few layers can effectively be ‘undone’ by removing any information that has been created to form the final objects and leaving behind stored lower order features. These undone networks can then have a smaller dataset inserted in place. With a large selection of features already established and a large selection of tuned weights set within the network, features can be adapted to the new images held within a smaller dataset. These image sets can be specific to objects; for example, when the desired object to train are vertebrate species, an image set called ‘pets’ would be recommended as it contains a large selection of images made up of vertebrate species such as dogs and cats. Therefore, its lower ordered features and assigned weights that would be held within the checkpoint model would be tailored towards vertebrate species and could be adapted within a new model easily. The MSCOCO image set is a general image set, hence the name ‘common objects’, where the common objects are considered to be made up of general shapes, meaning that its stored features and weights could easily be adapted to a range of specific subjects. This use of checkpoints is widely considered as an effective method to help train a model with small datasets and is routinely used within many investigations (Lim *et al*. 2011; Yang *et al*. 2016; Thenmozhi and Srinivasulu Reddy 2019).

2. Aims and Hypothesis

2.1 Aims

The general aim of this study was to create a proof of concept of an automatic identification tool of chironomid larvae heads mounted on microscope slides through the application of object detection classification and deep learning techniques. The purpose of this project was to develop a user friendly and computationally simple or 'lite' approach to rapidly classify images that could, in the future, be used by ecologists. This computational process involved the creation of an optimal object detection program by testing two different neural network architectures and varying object detection parameters that could automatically detect and distinguish three previously identified chironomid genera, *Cricotopus*, *Eukiefferiella* and *Rheotanytarsus*. For this study, a large set images of taxonomically identified head capsules of chironomid larvae mounted on microscope slides, originated from the River Stour (Kent), were used for the training, validation and testing of the object detection programs. This study then evaluated the models in their complete format, with top models being put through a separate testing phase to highlight the actual classification potential for identifying freshwater organisms for ecological studies, including biomonitoring.

The specific aims of this study were:

- To develop object detection models (using SSD and Faster-RCNN frameworks) with the intention of automatically identifying and classifying images of head capsules of chironomid larvae mounted on microscope slides belonging to three taxonomic genera, *Cricotopus*, *Eukiefferiella* and *Rheotanytarsus*.

- To study the effect of image augmentation within an object detection model for the classification of these three genera of chironomid larvae.

- To test how different hyperparameters (Learning Rate, Intersection Over Union) affect overall performance of object detection models of the three genera of chironomid larvae.

2.2 Hypothesis

It is hypothesised that:

- There will be a significant difference in the correct classification of the three genera of chironomid larvae between the two object detection frameworks, SDD and Faster-RCNN

- There will be a significant difference in the correct classification of the three genera of chironomid larvae between the three image sets (A, B and C) generated using different augmentation methods.

- There will be a significant difference in the correct classification of the three genera of chironomid larvae when varying the learning rate and Intersection over union hyperparameters.

3. Methodology

3.1 Specimen collection and Taxonomic identification

For the taxonomic identification of chironomid larvae, animals were collected via kick sampling at various points on the river stour, within the river benthos and sorted out visually from other freshwater invertebrates. Chironomid larvae were then mounted onto microscope slides, separating the head capsule from the rest of the body, and the mounted specimens were finally identified based on morphological features using taxonomic keys to the taxonomic level of genus (Vega *et al.* 2021). The original collection comprised of 1352 individually prepared chironomid head capsules mounted on microscope slides (Fig. 4).

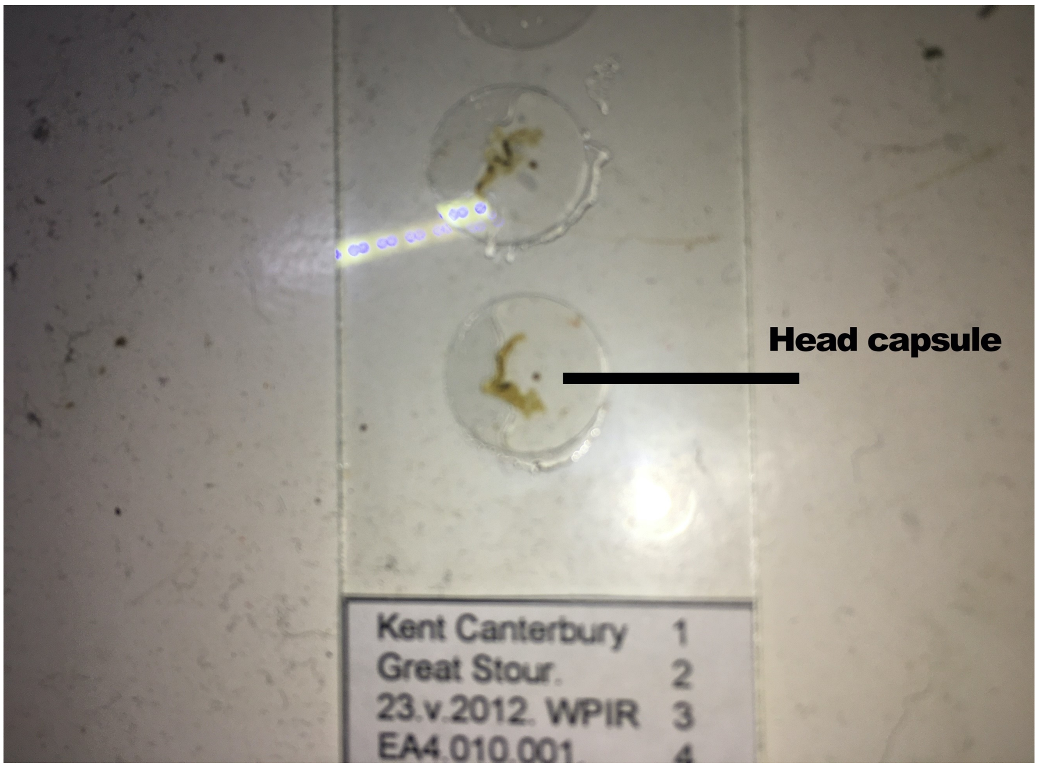


Figure 4. Image showing a close-up of a head capsule of a chironomid larva mounted on a microscope slide.

Some of the chironomid specimens had been identified to the taxonomic level of species; however, all were identified to genus level. Therefore, for this study, this was the taxonomic level used. Unfortunately, several specimens appeared to be labelled incorrectly, and a small number of samples had degraded or had been damaged during the microscope slide mounting steps. Therefore, because of this, a total of 863 specimens, which covered three chironomid genera (*Cricotopus*, *Eukiefferiella* and *Rheotanytarsus*) were used for the object detection experiment (Table 1).

Table 1. Chironomid genera and total number of images of identified specimens used for this study.

Genus	Number of images
Cricotopus	487
Eukiefferiella	115
Rheotanytarsus	261

3.2 Dataset preparation and Image annotation

3.2.1 Dataset construction

A database consisting of a set of images was constructed for use within this investigation where these images would comprise of chironomid head capsules. Although the head capsules are individually mounted on microscope slides, their minute size means that to currently view them, the use of a high-powered microscope was needed (40X total magnification). However, the microscopes available for this study were not equipped with image taking capabilities. Therefore, it was decided to develop a camera system that could capture images through the eye piece of a regular high-powered microscope. To achieve this, a Raspberry Pi (RPI), model 3b+ along with several other components was used (Raspberry Pi 2020). An RPI is a low powered and cheap credit card sized single board computer that is often used by researchers and hobbyists to construct a whole range of electronic and custom devices due to the large amount of modular components that can be added to it. These include, for example, wireless environmental monitoring systems (Ferdoush and Li 2014) and email-controlled home automation systems (Jain *et al*. 2014). Pagnutti *et al*. (2017) reviewed the capabilities of an RPI and its ability to capture and process images for use within the scientific and engineer community and concluded that it could produce high quality grade images. An RPI was set up to run the Raspbian Buster operating system. An RPI official camera v2.1 was used to take digital images of the chironomid larvae head capsule slides. This is designed to plug straight into the RPI for ease of use. A button was added to a breadboard and attached to the RPI at GPIO point 17 so that images could be captured in quick succession (Fig. 5).

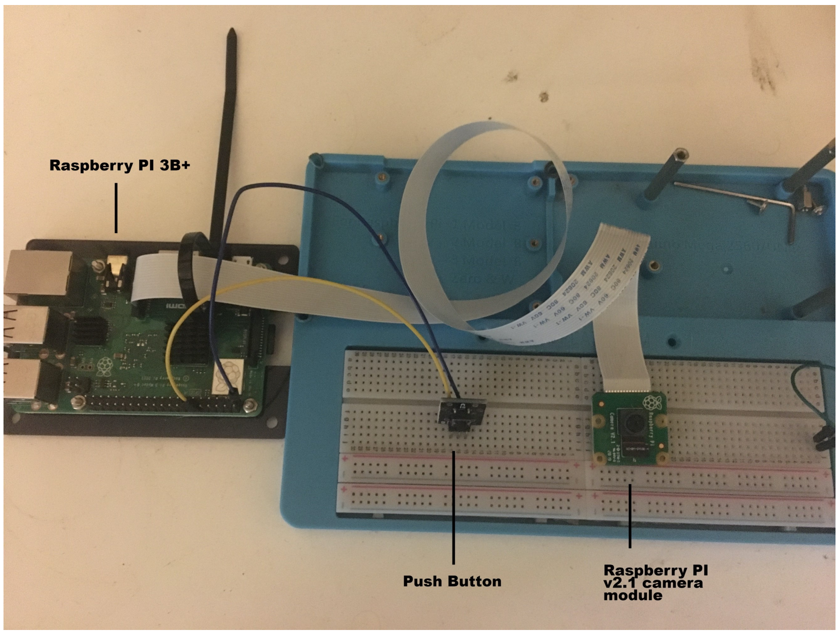


Figure 5. The Raspberry Pi (RPI) device used, showing the components for image capture (2-pin push button and RPI camera module) attached to a breadboard.

A script titled RPI_image_capture.py was written to produce a window box that displayed a video showing the feed from the camera module. When the button was pushed an image was captured and saved into an instructed folder. The camera feed remained visible until instructed to cancel. The camera module was mounted on top of an eye-piece of a high-powered microscope (Leica DM500). As there was no official camera mount available, a makeshift mount was created consisting of a piece of plastic sheeting with two thick rubber bands. The plastic sheeting was wrapped around one of the two available eye-pieces of the microscope and the camera module. One rubber band was placed at the bottom of the plastic sheeting to secure it around the eye piece. The camera module was then inserted into the tube so that the lens faced towards the eye-piece and it was secured into the plastic sheeting with the second rubber band. The plastic sheeting around the eye-piece secured by the first band allowed the camera to be moved closer and further from the eye-piece. This allowed for initial focusing and positioning of the video feed through a trial-and-error process, much like it would be expected if a person was using a microscope. Once a near optimum image was observed, final finetuning of the focusing was done using the course and fine adjustment knobs (focusing dials) of the microscope (Fig. 6).

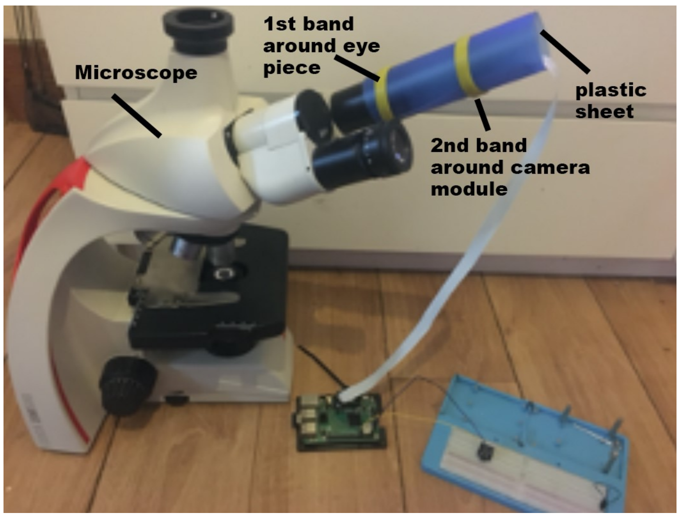


Figure 6. High power microscope (Leica DM500) and makeshift camera setup connected to Raspberry Pi (RPI).

The 4X objective lens was used for all images (40X total magnification). The microscope has an internal light source, so no additional light sources were used. Microscope slides containing the mounted specimens of chironomid larvae were placed onto the microscope stage, secured in place by the stage clips, and images of the slides were taken. Each microscope slide had either three or four chironomid larvae mounted, each one containing the head capsule and abdominal segments, and each specimen was placed under a circular cover slip (Fig. 7). The label on slide followed standard labelling system for presenting specimens mounted on microscope slides (location, site and date system). The slide is shown sitting on top of a labelled microscope slide box with general details. The head capsules were recorded with a reference number and all its taxonomic classification information.

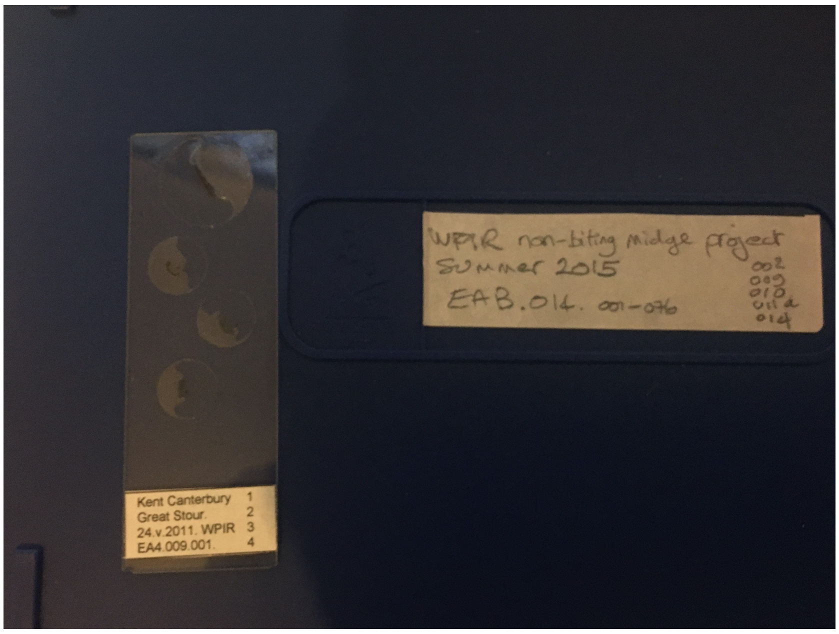


Figure 7. Image showing a microscope slide mounted with four chironomid specimens obtained from the River Stour, Kent (label indicates that samples were taken the 24th of May 2015, site EA4).

All identified specimens were photographed (Fig. 8) and the images were labelled with their corresponding Excel reference number so that they could be organised and referenced with ease. All images used were then organised into their respective taxonomic group, also known as their class, at the genus level.

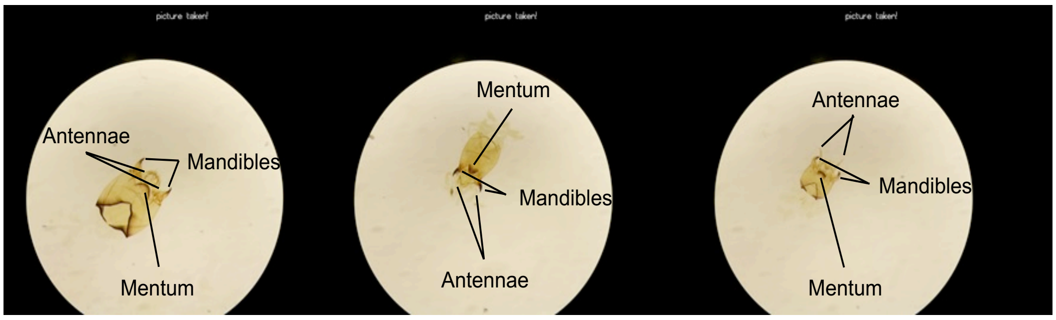


Figure 8. Examples of images of chironomid larvae head capsules captured under the microscope (40X total magnification) using a camera - Left to right: *Cricotopus*, *Eukiefferiella*, and *Rheotanytarsus*. Mandibles, antenna and mentums for each specimen has been labelled to allow a visual comparison of relative sizes and shapes of these structures which are typically used for morphological identification and taxonomic classification.

Cricotopus has wide head capsules with thin, curved mentums, relatively large mandibles and no obvious antenna. *Euiefferiella* has thin head capsules with dark mandibles and very dark, curved mentums. *Rheotanytarsus* has a wide head capsule, mandibles on the side of their heads, a flat mentum and very prominent antennae. The relative shapes and sizes of the mandibles also vary among the three genera.

3.2.2 Creating the bounding boxes

When training an object detection model, each image needs to be labelled or annotated with a bounding box creating a set of coordinates that can be used to train the object detection models. There are several methods for achieving this, including free opensource methods, paid for programmes and paid for service such as outsourcing to third party teams. For this investigation, a free opensource python-based programme called LabelImg (Tzutalin, 2015) was used. LabelImg has the ability to annotate several subjects within a single image. Figure 10 shows a screen grab of LabelImg in operation.

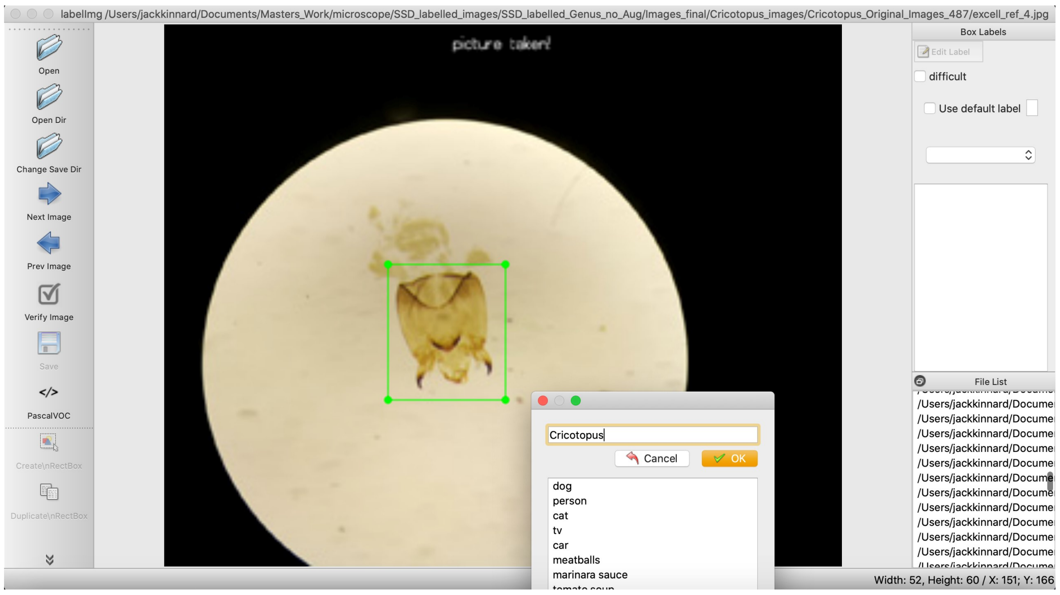

labelImg /Users/jackkinnard/Documents/Masters_Work/microscope/SSD_labelled_images/SSD_labelled_Genus_no_Aug/Images_final/Cricotopus_images/Cricotopus_Original_Images_487/excell_ref_4.jpg
Open
Open Dir
Change Save Dir
Next Image
Prev Image
Verify Image
Save
PascalVOC
Create\nRectBox
Duplicate\nRectBox
picture taken!
Cricotopus
Cancel
OK
dog
person
cat
tv
car
meatballs
marinara sauce
Box Labels
Edit Label
difficult
Use default label
File List
/Users/jackkinnard/Docume
Width: 52, Height: 60 / X: 151; Y: 166

Figure 10. Example of an image being annotated with bounding boxes (green square).

Here an image of a *Chironomid* head capsule is being labelled with a bounding box with label *Cricotopus*. This label can be anything desired by the user, in this case the three classes that represent the three *Chironomid* genera used: *Cricotopus*, *Rheotanytarsus* and *Eukiefferiella*. Therefore, all images were labelled with its corresponding bounding box label.

3.3 Pre-processing

3.3.1 Quality of data

While all images contain one chironomid head capsule of either of the three chironomid genera, the quality of the specimen preparation varied from complete subjects to those that had broken apart while conducting mounting procedures or due to general degradation. Some images contained rear segments, and some would contain entrails where the head capsule and rear segments were detached from each other. Some images also showed wear and degradation to the slide itself as air and dirt make their way within the slide. There was also a difference in the shade of the background within each slide. Figure 9 shows several different images from the genus *Cricotopus* image collection where differences in quality can be observed, such as a difference in background colour, the varying in the completeness of the structure of the head capsule itself and the quality of the slide.

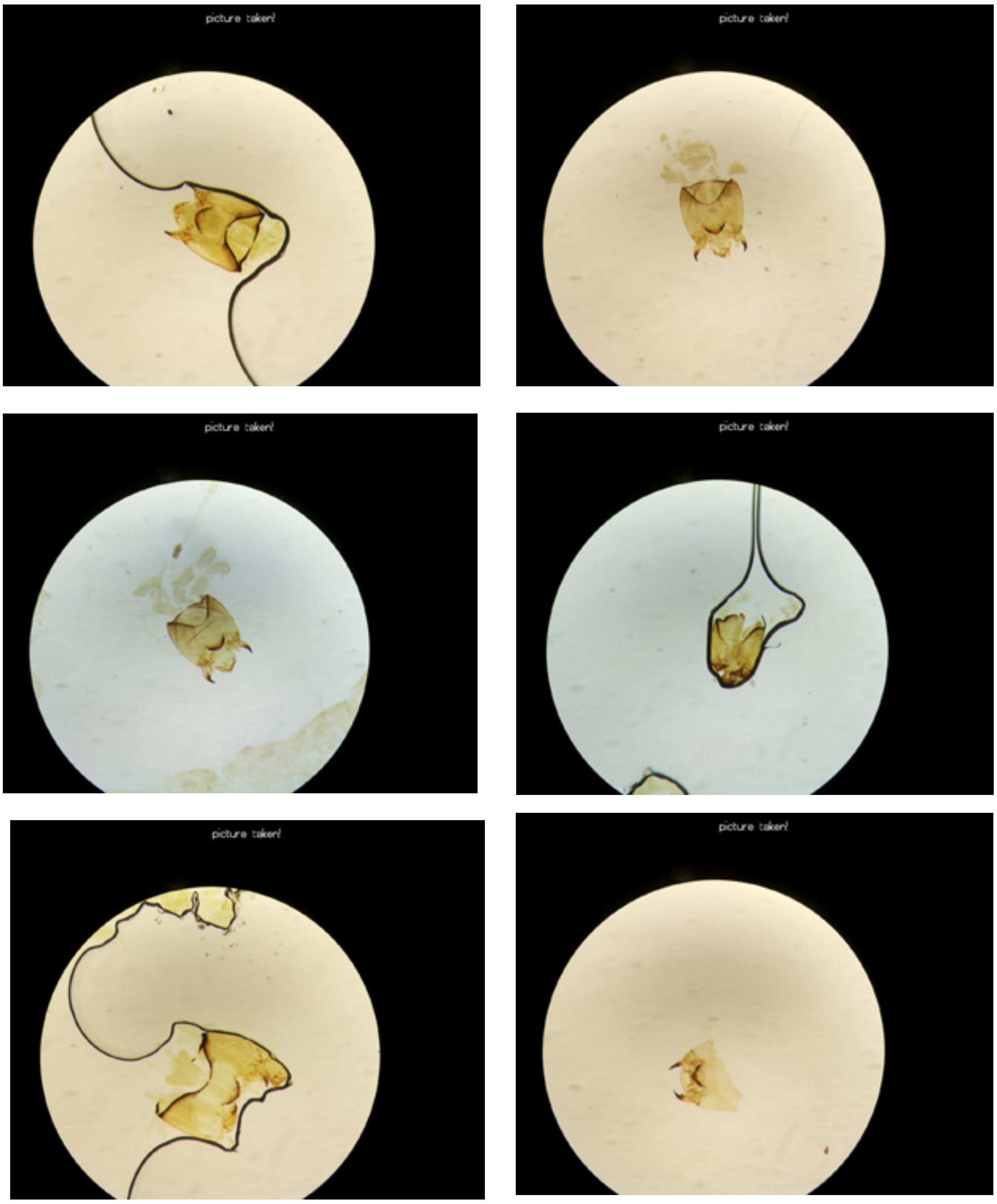


Figure 9. Head capsules of *Cricotopus* showing differences in quality of the images.

3.3.2 Splitting of test images

When training, validating, and testing the models used within this investigation, the images sets used within this investigation needed to split. To do this, the holdout method, was utilised wherein a set percentage of the total image set stock is set aside (Yadav and Shukla. 2016). These images are taken randomly from the stock of images and there is no set amount for this process which can range from various splits. For example, the following researchers utilise the holdout method with the following splits for their training and validation: 70%/30% (Russakovsky *et al*. 2015), 80%/20% (Yang *et al.* 2016) and 90%/10% (Chulu *et al.* 2019). When bringing on a final testing phase, this can then create another split, *i.e.* a split of 80%/10%/10% for training, validation and testing respectively. However, this investigation will be comparing many different models under different conditions, including different sized images sets. While the holdout method would be utilised for the training and validation split, set at 90%10% respectively, a set amount of test images was taken from the original stock of images so that all models in their finished format could be evaluated equally.

3.3.3 Test image removal

Thirty images were removed from each of the three classes original stock of images so that each class still had a fairly large number of images to train with across all image sets. A script titled rm_images.py was created to randomly remove 30 images from each class to avoid chances of biased based selection (Salkind 2010). With the test images removed this now brought the total images within each class to the following (Table 2):

Table 2. Image stock totals after removal of test images.

Genus	Number of images
Cricotopus	457
Eukiefferiella	85
Rheotanytarsus	231

3.3.4 Creation of image sets

Three image sets were created for comparison and evaluation of models. As proposed by Yue (2017) and Batista *et al*. (2016) it is recommended to keep images within each class balanced, so this advice was followed for all image sets. The first image set titled 'image set A' comprised a small number of images per class. As the class 'Eukiefferiella' had the lowest number of images (N = 85), this number was picked to be the number used for all classes, and the other two classes within this image set were down-sampled to this amount, giving a total of 255 images across the three chironomid classes. The second image set, titled 'image set B' comprised 500 images per class, which equated to 1500 images in total. The third image set, titled 'image set C' comprised 1000 images per class, which equated to 3000 images in total. However, this now required additional images to be added to each class for image sets B and C. While there are online databases that contain images of chironomid larvae head capsules, not all the required genera used were available within these databases. Therefore, this would now require image sets B and C to have additional images created through the augmentation purposes. Perez *et al*. (2018) proposed that a model's performance can be increased if the training, validation, and testing images are augmented. Because of this, it was decided to use augmentation for training and validation images for image sets B and C. however, as image set A used nonaugmented images for both training and validation images, it was decided to keep the test images nonaugmented so that could be consistent for all image sets.

3.3.5 Up-sampling by image augmentation

With image augmentation, a single image can have duplications but with random transformations made to effectively produce images that are considered new to the model that uses it to train with. There are a variety of augmentations that can be implemented. This includes, but is not limited to, translocations, horizontal and vertical flips, rotations, drop in image quality, zooming, colour changes, the removal of random pixels, and the removal of sections. There is currently no definitive list of which augmentations provide the best results and thus different investigations use different augmentations. For instance, Shijie *et al.* (2017) trialled several techniques but found flipping, rotating, and cropping as the most effective. Hussain *et al.* (2017) trialled several techniques and found flips, rotations and shears as the most affective forms of augmentation; they also notably showed a decrease in performance with augmentations such as the removal or adding of random pixels.

While these augmentations can be implemented by the user at an image-by-image basis, this could be considered time consuming. Thus, an automated application was used specifically for this purpose. While there are a several applications, both free to use and paid for, it was decided to use the free to use augmentation Python programme Augmentor (Bloice *et al.* 2017). This has a variety of different augmentation possibilities, which includes rotations, horizontal flips, vertical flips, zooms, warps and the removal of random sections of an image. Upon the completion of augmentations, a new image kept at the same resolution as the original is produced (*i.e.* an image with the resolution of 300x300 pixels produces a new augmented image with a 300x300 pixels resolution). Each augmentation can have a probability score between 0 – 1 (0% - 100%) which determines the frequency that each augmentation can occur, with each augmentation having its own respective metrics. The augmentations are designed to be done at random which creates a non-bias in their creation.

For instance, by observing the following code:

```
p.zoom(probability=0.5, min_factor=0.5, max_factor=1.5)
```

In this code the augmentation applied is a zoom factor (p.zoom), that has a probability of 0.5 (probability of 50%) and can be produced between 0.5 and 1.5, which equates to a zoom factor randomly set between the values of 0.5x and 1.5x. The augmentations used for this investigation were rotation to the left up to 180 degrees, rotation to the right up to 180 degrees, zooming in, zooming out, a horizontal flip and a vertical flip. Each augmentation had a random probability of 0.5 (50%) applied so that each augmentation had equal chances of being applied as well as a combination of all the augmentations. The script titled Augmentations.py was used to create all augmented images. Table 3 shows the number of original images within each class and the number of images that were augmented to bring this to 1000 images.

Table 3. Total original and augmented images for image set C.

Genus	Number of Images	Number of augmented images	Total number of images
Cricotopus	457	543	1000
Eukiefferiella	85	915	1000
Rheotanytarsus	231	769	1000

Table 4 shows the number of original images within each class and the number of the augmented images that were needed to bring this to a total of 500 images per class.

Table 4. total original and augmented images for image set B.

Genus	Number of images	Number of augmented images	Total number of images
Cricotopus	457	43	500
Eukiefferiella	85	415	500
Rheotanytarsus	231	269	500

Rather than creating new augmentations, the required number of images were randomly selected from the pools of augmented images for each respective class using the rm_images.py script.

3.3.6 Down-Sampling

To down-sample the majority classes, the rm_image.py script was used to randomly select 85 images from the original stock within both the *Cricotopus* and *Rheotanytarsus* files. This now created three image sets for investigation (Table 5).

Table 5. Highlights of the three image sets.

Image set	Number of images per class	Total number of images	Image type
A	85	255	All original
B	500	1500	Combination of original & augmented
C	1000	3000	Combination of original & augmented

3.3.7 Nonbiased image organisation

Using the holdback method, a split of 90%/10% for training and validation was used. The rm_image.py script was used with each image set to randomly remove the required images from each class from each image set:

Image set A: 10% of 85 = 9

Image set B: 10% of 500 = 50

Image set C: 10% of 100 = 1000

All three classes train and validation sets were combined from each respective image set which created new image splits within each image set (Table 6).

Table 6. Image sets with their train and validation amounts.

Image set	No. of training images	No. of validation images
A	228	27
B	1350	150
C	2700	300

3.3.8 Model architecture construction

Object detection models can be constructed using a variety of different components which includes an object detection framework, a CNN and an evaluation method. While these can be configured manually, the team at TensorFlow have also produced a collection of preconfigured models that contain both a pre-constructed model architecture which is linked via a configuration file and a pretrained model using this model configuration which has been trained from scratch using a large image data set (*I.e.* the MSCOCO set or the Pets set) which can be utilised for a transfer learning checkpoint (TensorFlow 2020a).

For use in this investigation, two model configurations were selected: The SSD_inception_v2_coco and the Faster_RCNN_Inception_v2_coco. Each model configuration is made up of an object detection framework, CNN and evaluation protocol. For instance, the SSD_inception_v2_coco consists of the object detection framework SSD, the CNN Inception v2, and the evaluation protocol MSCOCO. The SSD and the Faster-RCNN frameworks were selected as focal points for comparison within this investigation because each framework is recognised as powerful and accurate additions to object detection models, yet each is shown to work very differently to one another. However, to act as a standardisation, the same CNN and evaluation protocols were picked to work alongside both architectures. Therefore, the Inception v2 was picked because TensorFlow has created an SSD model and a Faster-RCNN model where both include this particular CNN. The MSCOCO metric protocols were selected due to their current use as one of the standard metrics within current object detection model evaluation within the majority of literature. The preconstructed networks can be downloaded from the TensorFlow model repository which is found on their GitHub page (TensorFlow 2020ab). As mentioned, these files also come with a pretrained model that has been trained using the MSCOCO image set which has created a general collection of features and weights within the internal network with the intention of being adapted when used with a new image set. These models were used as transfer learning checkpoints when new models were eventually trained.

Using the three image sets and the two different model configurations, combining these created six models for comparison (Table 7). From this point on, all models containing the model configurations of SSD_inceptionV2_coco and Faster-RCNN_inceptionV2_coco would be referred to as SSD and FR, respectively.

Table 7. Model configurations and image sets for comparison.

Model configuration	Image set
SSD	A
SSD	B
SSD	C
FR	A
FR	B
FR	C

3.3.9 Learning rate hyperparameter

Following the work of Xia *et al*. (2018), four LR hyperparameter values (0.1, 0.001, 0.005, and 0.0005) were chosen for model performance comparison (Table 8). Each model configuration would comprise of an object detection framework, an Image set and a LR.

Table 8. Model configuration, image sets and learning rates.

Model configuration	Image set	LR
SSD	A	0.01
SSD	A	0.001
SSD	A	0.005
SSD	A	0.0005
SSD	B	0.01
SSD	B	0.001
SSD	B	0.005
SSD	B	0.0005
SSD	C	0.01
SSD	C	0.001
SSD	C	0.005
SSD	C	0.0005
FR	A	0.01
FR	A	0.001
FR	A	0.005
FR	A	0.0005
FR	B	0.01
FR	B	0.001
FR	B	0.005
FR	B	0.0005
FR	C	0.01
FR	C	0.001
FR	C	0.005
FR	C	0.0005

3.3.10 IOU Union threshold hyperparameter

Three IOU threshold hyperparameter were chosen while keeping the optimum performing LR that worked best for the model configuration and image-set combination. The IOU values investigated were 0.5, 0.6 and 0.7. Both the SSD_InceptionV2_coco and Faster-RCNN_InceptionV2_coco model configurations had an IOU threshold initially set at a value of 0.6 when it was used to examine the LR threshold.

Table 9. Framework, image set and IOU variation.

Model configuration	Image set	IOU
SSD	A	0.5
SSD	A	0.6
SSD	A	0.7
SSD	B	0.5
SSD	B	0.6
SSD	B	0.7
SSD	C	0.5
SSD	C	0.6
SSD	C	0.7
FR	A	0.5
FR	A	0.6
FR	A	0.7
FR	B	0.5
FR	B	0.6
FR	B	0.7
FR	C	0.5
FR	C	0.6
FR	C	0.7

3.3.11 Configuration files

The configuration file that was downloaded with the pretrained models links all the separate components that make up the model configurations as well as dictating the hyperparameter settings and linking the transfer learning checkpoint. A configuration file comes with both the SSD_inceptionV2_coco and the Faster-RCNN_inceptionV2_coco. These files have already had hyperparameters set when they were training the respective models that they were included in. The majority of the values set for each respective hyperparameter is adequate to train models with custom image sets, however, there is a possibility a trained model's performance could be improved by finetuning the hyperparameters. Therefore, these files would be used for this investigation but would have their hyperparameters changed in several sections. Because each of the object detection frameworks are structured differentially in terms of their coding architecture and how they perform object detection, the configuration files and their respective hyperparameters are laid out differently. For instance, for the ssd_inception_v2_coco.config configuration file:

The number of classes needs to be set to 3 as there are three classes that equate to the three chironomid genera:

```
Line 9: num_classes:3
```

The size of the images that are used during the training stages can be determined within the configuration file where both the SSD and Faster-RCNN frameworks can automatically resize the images before they are utilised. The original image sizes of the chironomid specimens were of resolution 1280x1024 pixels. Several initial trials were conducted using a range of resolutions. Firstly, the images were left at their original resolution of 1280x1024 pixels, but this brought up memory allocation faults during trial training practises from the limitations of the GPU used. The file was then changed so that the images were resized to 512x512 pixels but again this brought up memory allocation issues. The resolution of 300x300 pixels was then trialled, and this was found to show no issue. Therefore, the configuration file was set to resized to the aspect ratio of 300x300 pixels.

```
Line 51: height: 300
Line 52: width:300
```

The batch size was set to 10. Several trials were set with batch size and a value of batch size 10 worked for both the SSD and the Faster-RCNN model configurations so this number was used with both to act as a form of standardisation:

```
Line 136: batch_size:10
```

The number of images within the validation image set (*i.e.* for the models using image set C, where the validation images within this set is 300):

```
Line 176: num_examples: 300
```

Several different epoch values were trialled, and it was found that the mAP value plateaued off around 3-5k epochs on the majority of models. The number of epochs was therefore set to 5000:

```
Line 157: num_steps: 5000
```

This also benefits from a small amount of time to train each iteration which was advantageous as there were many iterations of models throughout the investigation. The files also contain lines that need to be linked to the label map, transfer learning check point, the training images, training image labels, the validation images and the validation image labels which are found at lines 172, 151, 170 and 184, respectively.

3.4 Training

3.4.1 Workstation

All training was commenced from a workstation with the main contributing components being the Central Processing Unit (CPU) – Intel Core i7 9700K 3.6GHz Octa Core and Graphical Processing Unit (GPU) – Nvidia GeForce GTX 1060 3Gb, both of which are instrumental in leveraging as much performance potential when training an object detection model (Coates *et al*. 2009). The operating system was Linux Ubuntu Bionic beaver 18.04 operating system, with Python programming v3.6. All organised images were transferred from the RPI to this workstation.

3.4.2 TensorFlow Installation

Anaconda is a free open-sourced python package installer and was created with the intention of grouping together relevant and synergistic python packages aimed at data sciences and machine learning which can be quickly installed. It was installed following the protocols from their website (Anaconda 2020). When installed, the Ubuntu terminal was used to create an Anaconda virtual environment. This enables all installations and their respective dependencies to be installed and allows them to remain unchanged and unaffected by the continuous updating of any workstation for the duration of their use. This automatically installed Python v3.6 which is used for the duration of this investigation to run the majority of scripts and programmes from within the virtual environment.

The Anaconda distribution package of the TensorFlow object detection API version TensorFlow-GPU v1.15 was installed. This install includes a collection of programmes and dependencies that have been compiled by Anaconda that are required for the specific version of TensorFlow. This importantly includes Nvidia Cuda and Cudnn both of which are required to utilise the GPU when TensorFlow is training the models.

3.4.3 TensorFlow ‘Model_Main’ and Premade models

TensorFlow provide a master file system that can be downloaded from their GitHub page (TensorFlow 2020a). This contains all the necessary coding files that are required to train a variety of machine learning models which includes, but not limited to, object detection models. This was downloaded so that the training could be successfully run. Training is run using the ‘model_main.py” that is provided by tensorflow within their provided master file system

3.4.4 Training schedule

Each model configuration and image set trailed one of the 4 LR settings, where each was run three times. The resulting MSCOCO evaluation system produced an mAP value which were collected. These were used to calculate which LR hyperparameter variation worked best for each model configuration and image set by taking the average of the three runs. The three runs of the best performing LR models were then put through post training testing phases to compare their performances when used in real-time using the test images. Additionally, a significance evaluation in the form of a nested ANOVA was used to examine how the two model configurations compared, and how their nested image sets and hyperparameter variations affected the mAP value across all the trained models. This was then repeated for the IOU hyperparameter variations while using the optimum LR for each model configuration and image set combination.

3.4.5 Validation

The 'model_main' master file system contains internal coding which includes a validation process (Tensorflow, 2020c). This utilises the validation images from each of the image sets that were split using the hold-out method. This validation can be visual observed from within TensorBoard. This validation is represented as a loss value graph and can be viewed against the loss value of the training data graph which will indicate if a model is overfitting wherein the learned features become to generalised, rather than learning the desired subjects. While the validation has no bearing on the development of the model through the updating of weights during training processes, the validation can be used to verify if a model is developing as intended (Tensorflow, 2020c). The models training can be considered improving if the loss of both training and validation graphs is constantly decreasing at each epoch and if each is on par with each other in terms of the loss value. If either begins to increase or deviate from each other, then this is grounds to speculate that the model has failed to learn its desired subject or has generalised its features.

3.5 Testing

3.5.1 Post training model preparation

Firstly, the trained models had to be converted to a ‘finished model’ format. Once completed these could now be used to classify the 90 test images that were previously set aside. To achieve this, a script was created to classify specific objects (the chironomid larva head) from the batch of test images available. Figure 11 shows an example of one of the test images being classified by a model using the script where it predicted that the chironomid subject belongs to the genus *Rheotanytarsus*, where it has attributed it a confidence score of 100%.

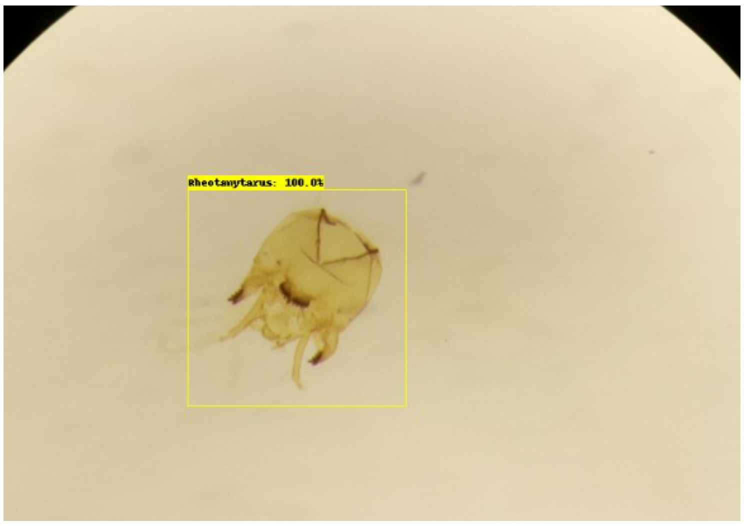


Figure 11. Example of a model showing its prediction and confidence score. A chironomid larva head capsule has been classified as belonging to the genus *Rheotanytarsus* with 100% confidence.

This script could classify a batch of images within a specified folder and given the name identify_bacth_images.py. Once executed the script produces a new window on the workstation that contains the image with the model’s predicted bounding box, as well as a confidence score (%) of how confident the model calculates its prediction. When the ‘Q’ button is pushed, the window will switch to the next image within the folder containing the test images. A single prediction for each of the 90 images was then recorded. If there was more than one prediction, only the prediction with the highest confidence score would be recorded and all lower predictions would be rejected. The threshold that dictates the minimum confidence score to accept as a positive classification was decreased to allow all images to be detected regardless of the confidence value. The results were then put into a 3x3 confusion matrix which is a specific table within machine learning that can be used to assess the performance of a trained model (Xu *et al*. 2019). Using this confusion matrix, a selection of metrics can be obtained to evaluate the working performance of the final model.

3.5.2 Confusion matrix analysis

Figure 12 shows a 2x2 confusion matrix in a format that is typical of simple classification model evaluation. Here, the model’s prediction is pitted against what is actually present. Both can either represent a positive or a negative result, and the combination will produce one of four possibilities: True Positive, False Positive, False Negative and True Negative. A data set is passed through a model where each result is tallied in one of the four outcomes. The amounts within each of the four outcomes can then be used to calculate a series of evaluation metrics. Equations 2 – 5 are used to calculate the following metrics: Recall, Specificity, Precision, and Accuracy.

		Predicted	
		Positive	negative
Actual	Positive	True Positive (TP)	False Positive (FN)
	Negative	False Negative (FN)	True Negative (TN)

Figure 12. A 2x2 confusion matrix typically used for simple classification model evaluation.

Where:

TP is a correct classification of a subject as a subject

FP is the number of non-subjects classification as subjects

FN is the number of subjects not classification as subjects

TN is the non-classification of a non-subject

Within classification, Accuracy is the measurement of the ratio between total correct predictions to the overall total predictions (Eq. 2) (Landgrede *et al*. 2006; Chudzik *et al*. 2020), defined as:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{2}$$

However, this current method would only work on a 2-class classifier. When a classifier takes on >2 classes, the confusion matrix and its respective equations need to be altered to accommodate the extra classes. While a 2-class classifier creates absolute values for the whole model, a model with >2 classes will require the working of individual metrics for each respective class and then an average of all classes taken. The models used in this study consisted of three classes (the three chironomid genera) and thus the confusion matrices were altered accordingly to accommodate this. Figure 13 shows the 3x3 confusion matrix with the three classes: *Cricotopus*, *Rheotanytarsus* and *Eukiefferiella*.

		Predicted		
		Cricotopus	*Rheotanytarsus*	*Eukiefferiella*
Actual	*Cricotopus*	**Tcc**	Fcr	Fce
	Rheotanytarsus	Frc	**Trr**	Fre
	Eukiefferiella	Fec	Fer	**Tee**

Figure 13. A 3x3 confusion matrix as used in this study for the classification of three chironomid genera.

Where:

Tcc is the correct identification of a *Cricotopus* with a prediction of *Cricotopus*

Fcr is the incorrect identification of a *Cricotopus* with a prediction of *Rheotanytarsus*

Fce is the incorrect identification of a *Cricotopus* with a prediction of *Eukiefferiella*

Frc is the incorrect identification of a *Rheotanytarsus* with a prediction of *Cricotopus*

Trr is the correct identification of a *Rheotanytarsus* with a prediction of *Rheotanytarsus*

Fre is the incorrect identification of a *Rheotanytarsus* with a prediction of *Eukiefferiella*

Fec is the incorrect identification of a *Eukiefferiella* with a prediction of *Cricotopus*

Fer is the incorrect identification of a *Eukiefferiella* with a prediction of *Rheotanytarsus*

Tee is the correct identification of a *Eukiefferiella* with a prediction of *Eukiefferiella*

And the accuracy equation for the Cricotopus class now becomes:

$$\textit{Cricotopus Accuracy} = \frac{Tcc + (Trr+Tee)}{Tcc\ + (Fcr + Fcn) + (Frc+Fec) + (Trr+Tee)}$$

(3)

3.5.3 Confidence threshold analysis

TensorFlow attributes a confidence score to each object predicted when a model is used post creation. This value is between 1-100% (where 0% would be no detection). The 90 test images were passed through each model using the identify_batch_images.py script. Again, threshold was decreased so that all detections were made, thus allowing all confidence scores to be recorded. If there was more than one prediction, only the prediction with the highest confidence score would be recorded and all lower predictions would be rejected. All images then had their confidence values tabulated. From this, each model had three metrics calculated: the mean confidence value which was derived from averaging of all confidence scores obtained, the minimum confidence value which was derived from the lowest value given from within the 90 test images, and the maximum confidence value which was derived from highest confidence score given from within the 90 test images.

3.6 Statistical analysis

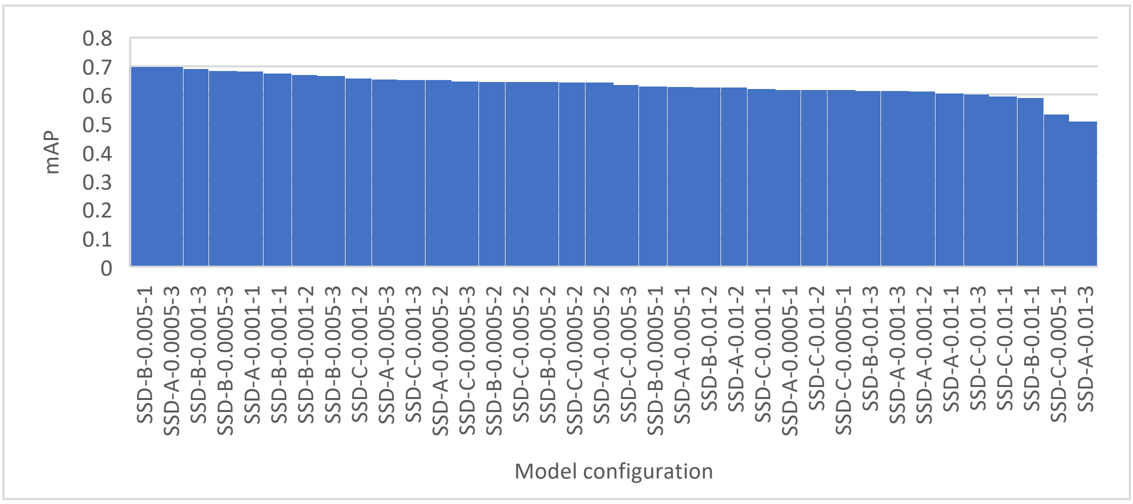
To test if there was a significant difference among the models, a nested Analysis of Variance (ANOVA) was performed on the mAP scores among on LRs nested within Image Sets nested within Models (SDD and FR). The data was first examined for normality using methods outlined by Holmes *et al*. (2016). A post hoc examination was used when significant differences were found among the nested groups. This nested ANOVA test was chosen as it would allow an examination of the relationship between the two object detection frameworks, and how the image sets and learning rates all work together to produce the overall mAP values. This statistical analysis was performed using Minitab v19 (Minitab, LLC).

4. Results

4.1 Learning rate mAP analysis

The training of the six different models' configurations had their LR varied (0.01, 0.001, 0.005, 0.0005) and each was run three times. All these models final mAP is reported in Figure 14**.** The highest mAP value obtained for the SSD models was 0.698 produced by the configuration SSD-B-0.005. The lowest mAP value obtained for the SSD models was 0.507 produced by the configuration SSD-A-0.01. The highest mAP value obtained for the FR models was 0.747 produced by the configuration FR-A-0.001. The lowest mAP value obtained for the FR models was 0.624 produced by the configuration FR-A-0.0005.

A)



B)

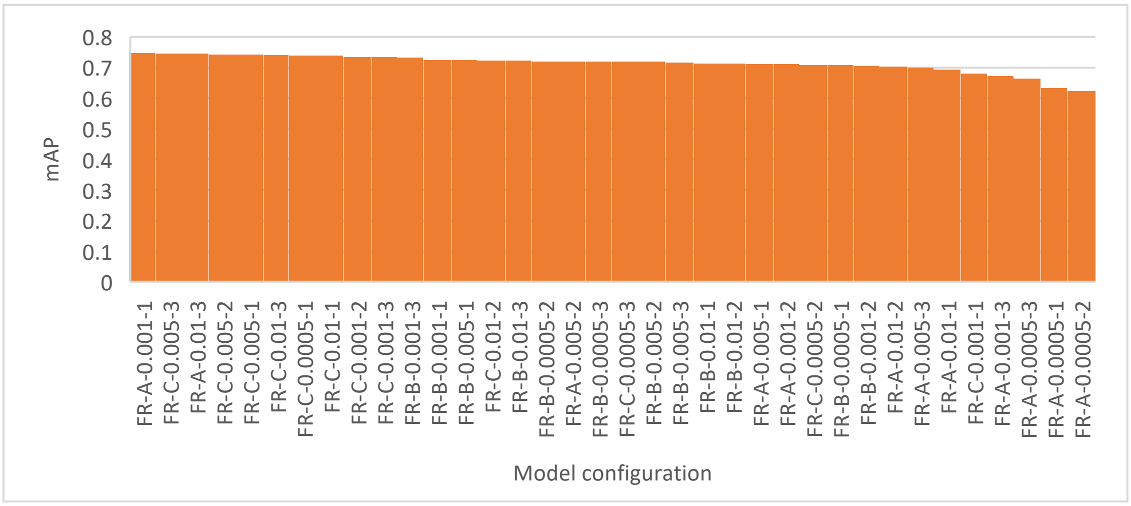


Figure 14. mAP values in ascending order for A) SDD models and B) FR models, each under four different Learning Rates (LR) and three replicates.

The three runs for each LR were then averaged; averaged mAP values for SDD ranged between 0.579 and 0.678, and for FR ranged between 0.639 and 0.744; in all cases FR showed larger mAP values than SDD (Fig. 15). The averaged mAP values indicated which LR was most effective for each model and image-set combination, and thus these runs were brought forward for evaluation and the values were kept when investigating the second hyperparameter IOU.

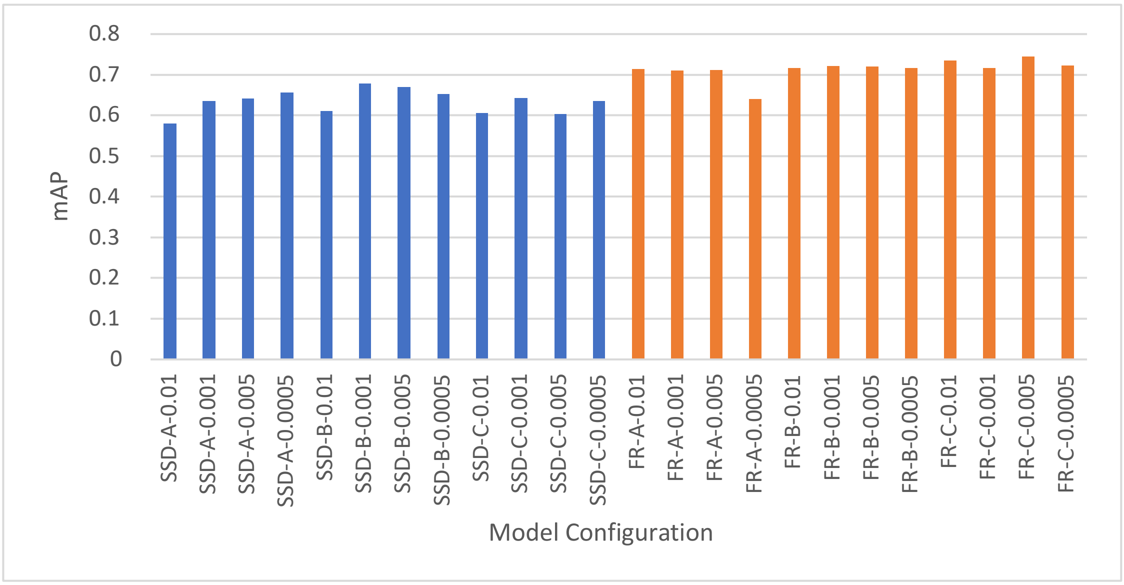
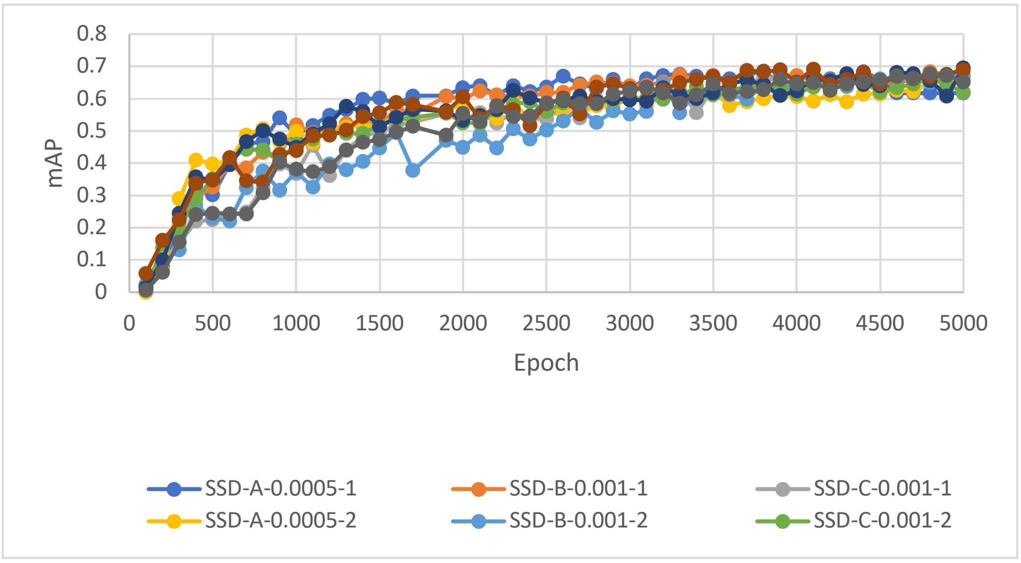


Figure 15. Average mAP values for the three replicate runs of each model configuration (SDD and FR).

Figure 16 shows the mAP values for the best performing SSD and Faster-RCNN focused model configurations with the alternating LR and how they developed over the course of training for the 5000 epochs. The SSD models all started producing mAP values near 0 and increased over time. However, the FR models all produced starting mAP values over 0.4, increased rapidly and then levelled off and maintained a mAP value of approximately 0.7.

A)



B)

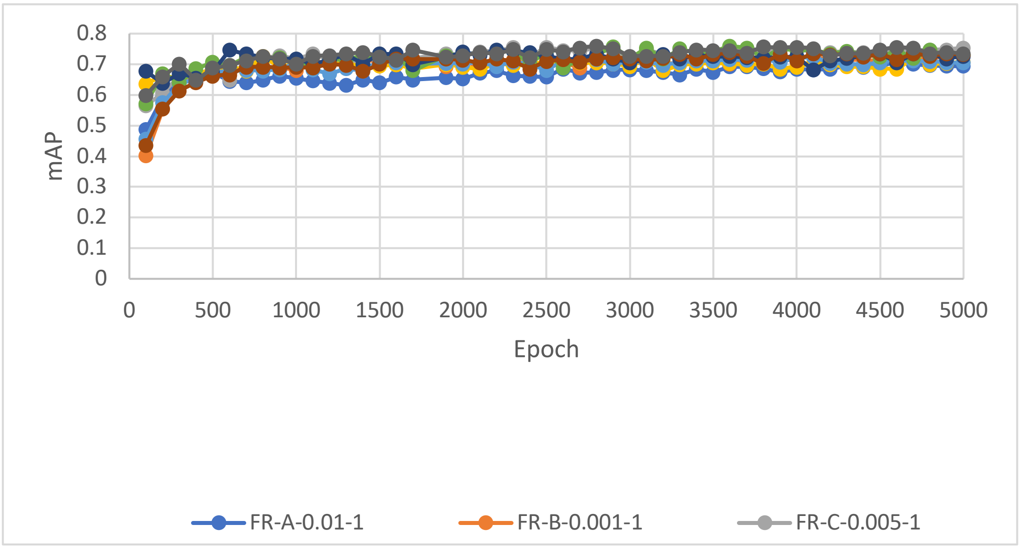


Figure 16. Best performing models for A) SDD models and B) FR models, each under four different Learning Rates (LR) and three replicates. The models were run for 5000 epochs.

4.2 Learning rate confusion matrix analysis

The confusion matrix evaluation protocol outlined in Figure 13 were then performed on all of the best performing models that were brought forward with their respective runs.

Table 10. Confusion matrix results for the best performing model configurations.

Model Config	Run	Accuracy
SSD-A-0.0005	1	0.967
	2	0.978
	3	0.978
SSD-B-0.001	1	0.956
	2	0.978
	3	0.978
SSD-C-0.001	1	0.978
	2	0.978
	3	0.956
FR-A-0.01	1	0.956
	2	0.967
	3	0.978
FR-B-0.001	1	0.978
	2	0.978
	3	0.978
FR-C-0.005	1	0.978
	2	0.978
	3	0.956

Table 10 displays the accuracy score for all the best performing model configurations across their three runs. There was little difference in the values between all the models and their runs. For instance, the worst accuracy score of 0.955 was achieved by the configuration's SSD-2-0.001 run 1, SSD-3-0.001 run 3, FR-1-0.01 run 1 and FR-3-0.005 run 3. The best performing accuracy score of 0.977 was achieved by all the model configurations on at least one run and was achieved by all three runs on the model configuration FR-B-0.001.

4.3 Learning rate confidence score analysis

The confidence score evaluation protocols were then performed on all of the best performing models that were brought forward with their respective runs. Table 11 displays the average confidence score, minimum confidence score and maximum confidence score. The highest average confidence score of 99.9% was achieved by configuration FR-B-0.001 run 2, however, all the FR models achieved an average confidence score of above 99%. The lowest average confidence score of 80.92% was achieved by model configuration SSD-C-0.001 run 3. All SSD models achieved an average confidence score within the range of 80-90% except for SSD-A-0.0005 run 2 which achieved a score of 91.01%. The minimum confidence score achieved across all configurations was 03% which was achieved by configuration SSD-A0.0005 run 2. The maximum confidence score of 100% was achieved by all configurations. The lowest minimum confidence score for the FR model of 42% was achieved by configuration FR-A-0.01 run 3. The highest minimum score for the FR models of 95% was achieved by configuration FR-B-0.001 run 3. The highest minimum score for the SSD models was 27% and was achieved by the configuration SSD-B-0.001 run 3.

Table 11. Confidence score analysis for best performing model configurations for SDD and FR models.

Model Config	Run	Av Conf (%)	Min Conf (%)	Max Conf (%)
SSD-A-0.0005	1	83.430	07	100
	2	91.010	03	100
	3	87.640	14	100
SSD-B-0.001	1	86.840	19	100
	2	82.950	13	100
	3	89.910	27	100
SSD-C-0.001	1	86.100	13	100
	2	86.780	05	100
	3	80.920	15	100
FR-A-0.01	1	99.040	71	100
	2	99.560	78	100
	3	99.300	42	100
FR-B- 0.001	1	99.430	81	100
	2	99.900	93	100
	3	99.800	95	100
FR-C-0.005	1	99.740	93	100
	2	99.180	69	100
	3	99.530	84	100

4.4 Statistical analysis for all LR models

A nested ANOVA test was used to evaluate for significance of the mAP scores achieved by all model configurations by comparing the two frameworks and how each was influenced by the image set and LR (Table 12). There was a significant difference between the two frameworks and among the different learning rates; however, there was no significant difference among the different learning rates. This produced an R^2 value of 81.87%.

Table 12. Nested ANOVA of mAP values with three levels (Framework, Image set and Learning rate).

Source	DF	SS	MS	F	P
Framework	1	0.114	0.114	31.806	0.005
Image Set	4	0.014	0.004	1.840	0.165
Learning Rate	18	0.035	0.002	2.583	0.005
Error	48	0.036	0.001		
Total	71	0.200			

After removing Image Set from the nested design, the nested ANOVA showed significant differences in mAP values between the two models, and among the different learning rates within the models (Table 13). This produced an R^2 value of 68.20%.

Table 13. Nested ANOVA of mAP values with two levels (Framework and Learning rate).

Source	DF	SS	MS	F	P
Framework	1	0.114	0.114	30.920	0.001
Learning Rate	6	0.022	0.004	3.720	0.003
Error	64	0.064	0.001		
Total	71	0.200			

The post-hoc test showed that there was a significant difference between the FR architecture and the learning rate of 0.0005 and that there is significance between the SSD architecture and the learning rates 0.01 with p-values <0.05 (Table 14).

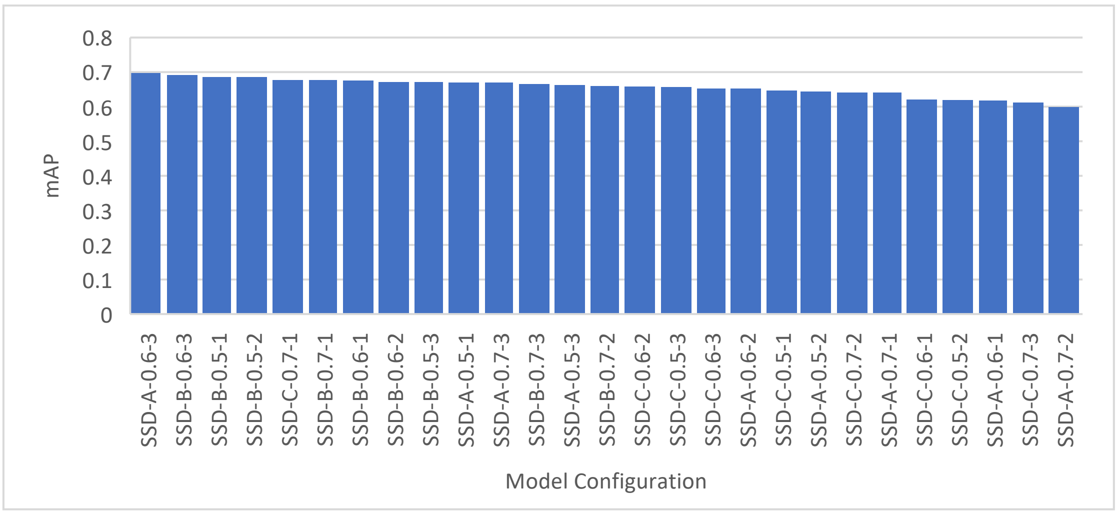
Table 14. Results of post-hoc examination for Learning Rate among the two different frameworks (SSD and FR).

Term LR(Framework)	P-Value
0.0005(FR)	0.025
0.0010(FR)	0.826
0.0050(FR)	0.225
0.0100(FR)	0.397
0.0005(SSD)	0.138
0.0010(SSD)	0.050
0.0050(SSD)	0.641
0.0100(SSD)	<0.001

4.5 IOU mAP analysis

The training of the six different model configurations had their IOU varied three times (at 0.5, 0.6 and 0.7), and each was run three times. All these models final mAP is reported in figure 17.

A)



B)

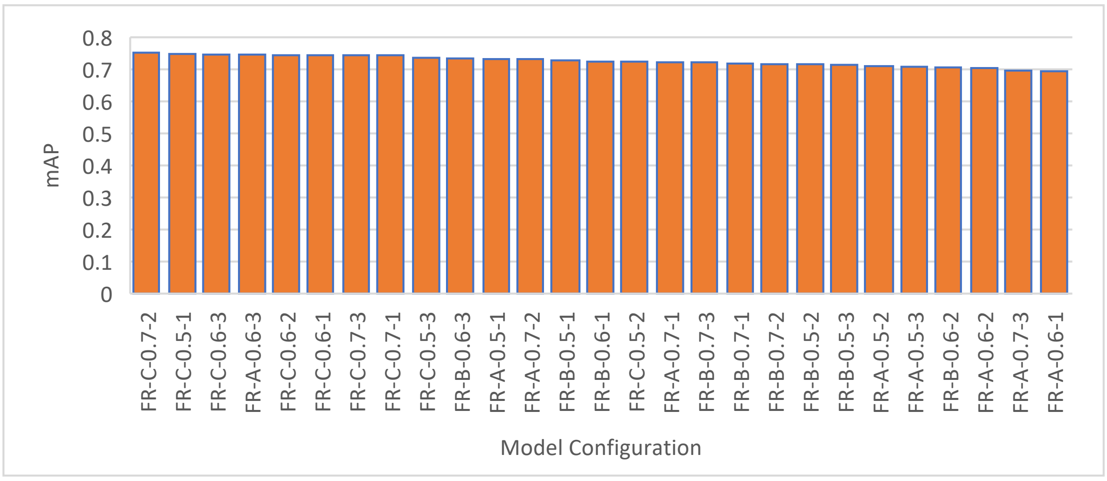


Figure 17. mAP values arranged in ascending order for A) the best SDD models and B) the best FR models, each under three different IOUs (0.5, 0.6 and 0.7) and three replicates.

The three runs for each IOU were then averaged; averaged mAP values for SDD ranged between 0.639 and 0.679, and for FR ranged between 0.713 and 0.745; in all cases FR showed larger mAP values than SDD (Fig. 18). The averaged mAP values indicated which IOU was most effective for each model and image-set combination, and thus these runs were brought forward for evaluation.

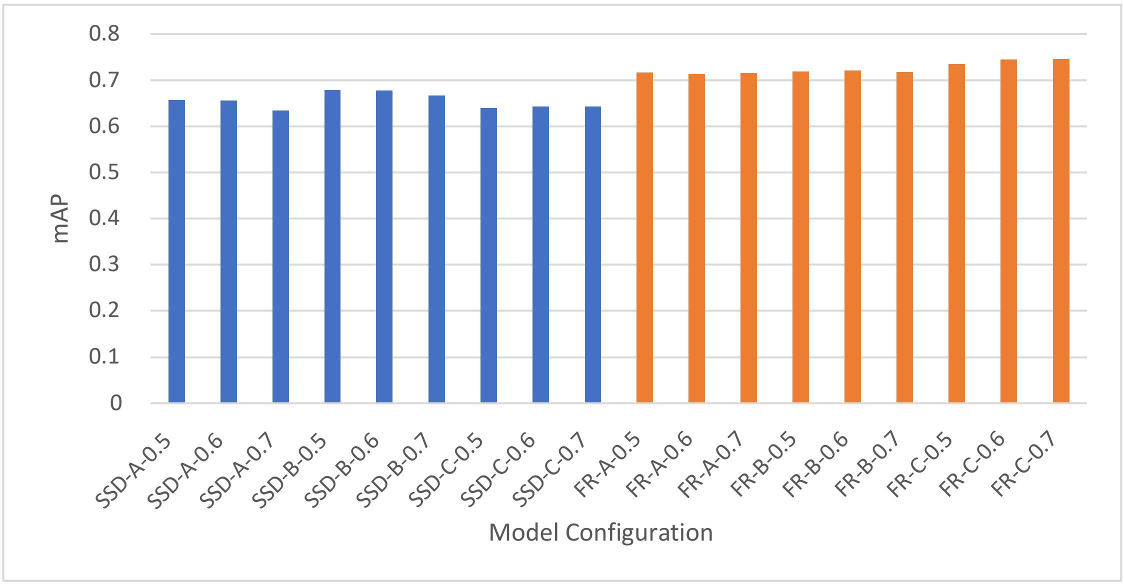
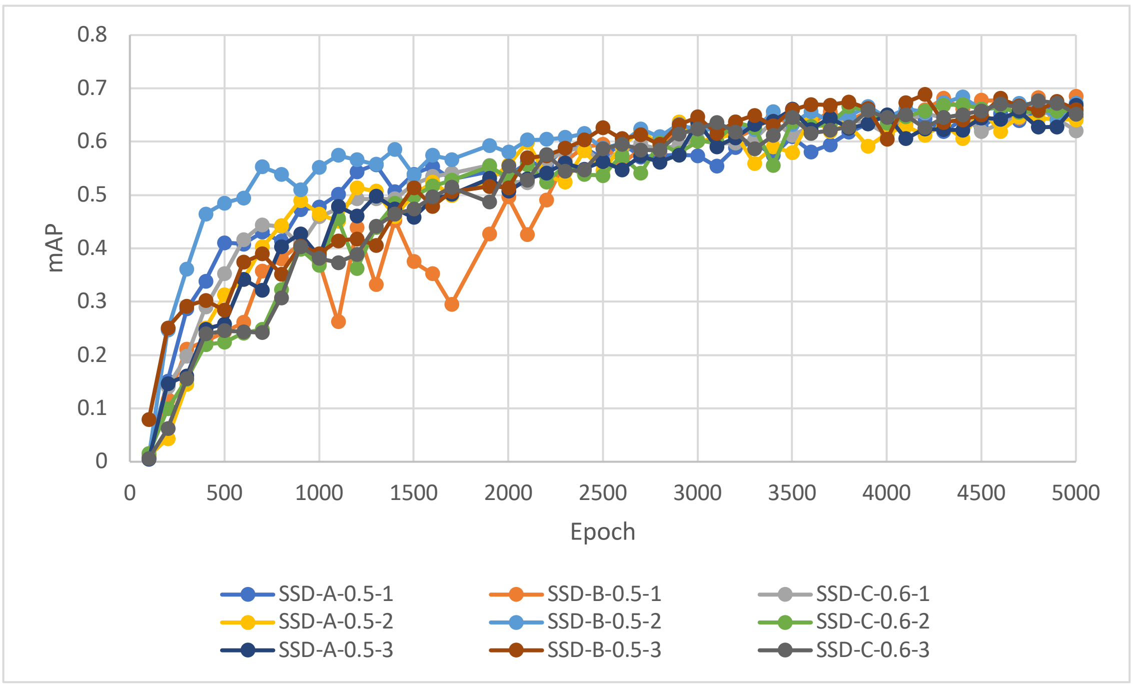


Figure 18. Average mAP values for SDD and FR models, each under three different IOUs (0.5, 0.6 and 0.7) and three replicates.

Figure 19 shows the mAP values for the best performing SSD and FR configurations with the alternating IOU and how they developed over the course of training for the 5000 epochs. After 100 epochs, the SSD models started with mAP values near 0 and increased over time but remained below 0.7. However, the FR models always started with mAP values over 0.4 after 100 epochs, increased rapidly, then levelled off and maintained mAP values above 0.7.

A)



B)

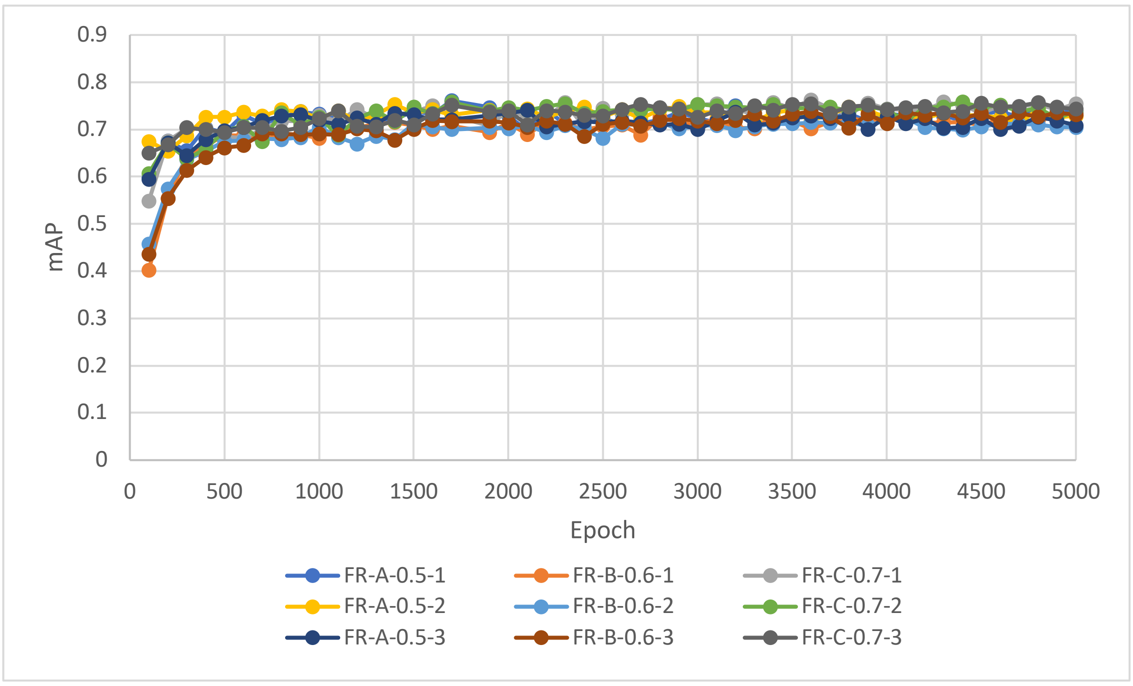


Figure 19. mAP values for the best performing for A) SDD and B) FR models, each under three different IOUs (0.5, 0.6 and 0.7) and three replicates run for 5000 epochs.

4.6 IOU confusion matrix analysis

The accuracy values for all the best performing models across their three runs are shown in Table 15. There was little difference in the values between all the models and their runs. For instance, the worst accuracy score of 0.944 was achieved by the FR-A-0.5 run 1, and the best performing accuracy score of 0.9777 was achieved by configuration SSD-B-0.5, SSD-C-0.6, FR-B-0.6 and FR-C-0.7 on at least one of their runs.

Table 15. Confusion matrix analysis for best performing models.

Model Config	Run	Accuracy
SSD-A-0.5	1	0.967
	2	0.967
	3	0.967
SSD-B-0.5	1	0.956
	2	0.978
	3	0.967
SSD-C-0.6	1	0.978
	2	0.978
	3	0.956
FR-A-0.5	1	0.944
	2	0.967
	3	0.967
FR-B-0.6	1	0.978
	2	0.978
	3	0.978
FR-C-0.7	1	0.978
	2	0.956
	3	0.978

4.7 IOU confidence score analysis

Performing the confidence score evaluation protocols on all the best model configurations showed important differences between SSD and FR (Table 16). The highest average confidence score of 99.99% was achieved by the configuration FR-B-0.6 run 2, however, all the FR models achieved an average confidence score of above 99%. The lowest average confidence score of 80.92% was achieved by configuration SSD-C-0.6 run 3. The highest average confidence score for the SSD was 91.98 achieved by SSD-A0.5 run 2. The minimum confidence score achieved across all configurations was 4% which was achieved by SSD-A-0.5 run 2. The maximum confidence score of 100% was achieved by all configurations. The lowest minimum confidence score for the FR model of 52% was achieved by FR-C-0.7 run 2. The highest minimum score for the FR models was of 95% was achieved by FR-B-0.6 run 3. The highest minimum score for the SSD models was 44% and was achieved by the MC SSD-B-0.5 run 3.

Table 16. Confidence score analysis.

Model Config	Run	Av Conf	Min Conf	Max Conf
SSD-A-0.5	1	89.420	4	100
	2	91.620	13	100
	3	81.000	26	100
SSD-B-0.5	1	86.370	12	100
	2	91.980	39	100
	3	91.170	44	100
SSD-C-0.6	1	86.100	13	100
	2	86.780	5	100
	3	80.920	15	100
FR-A-0.5	1	99.180	64	100
	2	99.460	57	100
	3	99.200	67	100
FR-B-0.6	1	99.430	81	100
	2	99.990	93	100
	3	99.800	95	100
FR-C-0.7	1	99.620	85	100
	2	99.070	52	100
	3	99.540	69	100

4.8 IOU significance evaluation

A nested ANOVA test was used to evaluate for significance of the mAP scores achieved by all model configurations by comparing the two frameworks and how each was influenced by the image set and IOUs (Table 17). There was a significant difference between the two frameworks and among the different image sets; however, there was no significant difference among the different IOUs. This produced an R^2 of 85.09%.

Table 17. Nested ANOVA of mAP values with three levels.

Source	DF	SS	MS	F	P
Framework	1	0.067	0.067	29.790	0.005
Image-Set	4	0.009	0.002	18.398	<0.001
IOU	12	0.002	0.000	0.324	0.980
Error	36	0.014	0.000		
Total	53	0.091			

After removing IOU from the nested design, the nested ANOVA showed significant differences in mAP values between the two models, and among the different image sets within the models (Table 18). This produced an R^2 of 83.48%.

Table 18. Nested ANOVA of mAP values with two levels (Framework & Image set).

Source	DF	SS	MS	F	P
Framework	1	0.066831	0.066831	29.79	0.005
Image-Set	4	0.008974	0.002243	7.18	<0.001
Error	48	0.015006	0.000313		
Total	53	0.090811			

The post-hoc test showed that there was a significant difference for FR with Image Sets A and C and for SSD with Image Sets B and C with with p-values <0.05 (Table 19).

Table 19. Results of post-hoc examination.

Term	P-Value
ImageSet(Framework)	
A(FR)	0.044
B(FR)	0.194
C(FR)	0.001
A(SSD)	0.215
B(SSD)	<0.001
C(SSD)	0.008

5. Discussion

This investigation set out to develop an object detection model that would be able to classify three different chironomid genera (*Rheotantarsus, Cricotopus* and *Eukiefferiella*). To study this, 863 head capsules of three chironomid larvae were photographed and subjected to classification by using an object detection system using the frameworks SSD and Faster-RCNN. As a proof of concept, the results demonstrated that object detection systems and deep learning techniques can be successfully used for the correct identification and classification of chironomid larvae mounted on microscope slides. The use of object detection and deep neural networks is a relatively novel approach for identifying organisms for ecological studies with potential for future implementations for biomonitoring freshwater ecosystems.

To achieve the best performance in terms of classification of the three chironomid genera, several approaches were investigated during the development of the models. This included comparing two object detection frameworks, comparing three different images sets, and comparing the effect of changing two different hyperparameters at separate times, the Learning Rate (LR) and Intersection Over Union (IOU) threshold. By reviewing literature, three hypotheses were proposed for the performance of the different models evaluated based on the MSCOCO mAP scoring system. The first hypothesis proposed that there would be a significant difference in the correct classification of the three genera of chironomid larvae between the two object detection frameworks, SDD and Faster-RCNN. The second proposed that there would be a significant difference in the correct classification of the three genera of chironomid larvae between the three image sets (A, B and C) generated using different augmentation methods. The third proposed that there would be a significant difference in the correct classification of the three genera of chironomid larvae when varying the learning rate and Intersection over union hyperparameters. Moreover, a series of novel post training evaluations were conducted on all the best performing models to see how their post training performance, as an actual detection model for chironomid larvae would compare against each other.

5.1 Object detection framework significance

There were significant differences between the means of both object detection frameworks, SSD and FR. Almost all of the models using FR achieved mAP values over 0.7 with the highest reported value of 0.747, whereas the models using the SSD framework achieved mAP values under 0.7 with the highest reported value of 0.698. This difference between the FR and SSD was seen during the second round of hyperparameter testing where the IOU threshold was varied. Therefore, it can be ascertained that the FR framework is better than SDD, as expected. Previous studies had already described the greater mAP scores for FR than SSD; for example, Arcos-Garcia *et al*. (2018), and Janahiraman and Subuhan (2019).

This difference in object detection performance could be due to the way that the frameworks develop over their specific training sessions. For instance, SSD frameworks took more time to reach the highest mAP scores, and the first reported mAP value was near 0 at epoch 100. However, the FR frameworks almost instantly reached a mAP of over 0.4 by epoch 100, and some models even reached mAP values over 0.6 and near 0.7. Interestingly, and to the best of the author’s knowledge, there is no literature that explains why either framework would develop in such ways. A possible explanation could be that the finetuning checkpoint that was created with the FR framework was able to produce a set of features and weights that were more suited towards the classification of chironomid larvae, and their respective features needed to make an accurate prediction when training was conducted.

5.2 Image sets significance

When focusing on the models where the LR was varied, the nested ANOVA indicated that there was no significance between the image sets (Table 12), although the image sets did have some impact on the performance of the models. When the image set group level is included within the nested ANOVA, the R^2 was 81.87%. When the image set level is removed, the R^2 was 68.20%. This would indicate that while it was not significant, the image sets did have bearing on the overall relationship of the different levels of the model.

When reviewing the results from models from the IOU analysis, the results do suggest that there is significance within the values obtained from the image sets. By observing Table 18, the p-value was <0.001 which would indicate it being strongly significant. By observing Table 19, a post hoc examination indicated that there was a relationship between the two frameworks and the three image sets. However, there was no consistency in which image sets have produce significant results. The SSD framework was significant when paired with image set A and C while the FR framework was significant when paired with Image sets B and C. Image set C, the largest of the three image sets agrees with discussed literature, however, it was somewhat surprising that Image Set C did not have a greater effect with the two architectures and Learning Rates. Previous studies indicate that the larger the image or data set, the more relationships within the network itself that can potentially be constructed (Srivastava *et al.* 2014; Shijie *et al.* 2017; Hussain *et al.* 2017; Wong *et al.* 2016; Valan *et al.* 2019), yet this did not produce the strongest evidence of significance.

Examples of image set numbers from the literature indicate that there is a general lack of consistency when deciding on the total number of images to include for training (Krizhevsky *et al.* 2012; Xia *et al*. 2018; Chulu *et al.* 2019). While it is generally considered that when training a model from scratch, there is a need for a large image set to allow a build-up of features and weights (Kermany *et al*. 2018), this can be alleviated by using a finetuning checkpoint, thus negating the need for a large image set. However, a lack of consistency has created a lack of guidance that could instruct researchers on how to utilise their specimens to create their own image sets to eventually create their own automated classifiers. For example, the original chironomid collection used in this investigation could be seen as both small in size and imbalanced between the classes. The collecting of chironomids was part of several surveys across a time frame of several years (Vega *et al. 2021*) and several factors (ecological and/or methodological) could have led to the resulting imbalanced and small amounts of chironomid specimens available for this investigation (in relevance at least to the numbers needed for the training of a computer-based model). Similar small and imbalanced specimen numbers can be observed in similar chironomid surveys (Johnson and Bligh, 1995; Orendt 1999; Lencioni *et al.* 2012) and this is mirrored in surveys of other animal groups (Smart *et al.* 2005; Avenant 2011). This suggests that if another researcher within this field wanted to create their own automated classifier, they would most likely be faced with similar imbalanced specimens' groups and a small number of specimens to work with. The inconsistency between the literature and the results obtained within this investigation indicate that perhaps with the use of finetuning checkpoint along with a correct framework is more important to the performance than the number of images required and whether they are augmented not when attempting to optimise a model.

5.3 Hyperparameter significance

When focusing on the hyperparameter testing groups, the combination of Learning Rate (LR) and architecture showed significant relationships. Significant effects were found when the SSD framework was paired with LR 0.01, and when the FR framework was paired with LR 0.0005. Previous studies show no guidance or universal LR value (Chudzik *et al.* 2020), indicating that each model and its respective neural network will require an optimisation of its own individual LR. This suggests that the LR rates that performed best with their respective frameworks were indeed the most optimum out of the four LR trailed. When reporting the SSD framework, Liu *et al*. (2016) set the learning rate at 0.001 and Ren *et al.* (2016) set the LR at 0.003 when reporting the FR frameworks, further reinforcing the idea that this value is independent to the individual model; this also means that LR must be finetuned on a modelby-model basis to achieve the best possible results from the training sessions.

There was no significant relationship between the different IOU values trialled and mAP values. However, there was a small effect of model performance (1.61% difference in the strength of the relationship with and without IOU). Thus, all together, alternating the IOU threshold hyperparameter value could be considered negligible in the general performance output of the models in terms of its mAP score. When observing the individual mAP that was obtained from each individual model configuration, image set, IOU and run, an IOU of 0.5 did increase the mAP value of the best performing FR model, however this increase was only by 0.02%.

5.4 Statistically reviewing the performance of a whole model

A nested ANOVA was chosen for model evaluation because it would allow to see if there was variation in mAP scores between the two architectures and within nested subgroups of image sets and hyperparameters. The nested ANOVAs showed significant differences in mAP between the two frameworks, as expected from previous studies, however, any significance between the remaining factors and variables within the model had not been explored previously. A nested ANOVA can effectively be broken down into its individual levels where each could be considered a one-way ANOVA (Bentler and Satorra 2010). By following the protocols of a one-way ANOVA (Doncaster and Davey 2007), a minimum of three repetitions is needed to review the means of a group for significance, therefore justifying that the number of repetitions of each respective model chosen was enough to meet the requirements and thus reinforce the significant values obtained. However, this justification and the significance would have been greater if there were more repetitions performed. Unfortunately, due to the length of time required to train each model and the time frame available, only three replicate runs were possible.

Furthermore, previous studies on object detection and computer vision models either do not carry out replications or examine the variances among replicate runs while reporting overall performance. Within the life sciences, replication is fundamental for confirming that observed patterns are not due to chance (Cassey *et al*. 2006). Yet within the computer sciences, replication appears to be unreported. Without replication, statistical examinations and significance tests cannot be performed. As mentioned in the introduction, almost all object detection models exclusively report one single mAP value (see for example Girshick *et al*. 2014; Liu *et al*. 2016; Ren *et al*. 2016; Ren *et al*. 2017; Arcos-Garcia *et al*. 2018; Shi *et al*. 2019; Bose and Kumar 2020). mAP scores are typically ranked against models of different architecture construction, yet the replication of a single object detection model has yet to be observed within the computer science literature. By observing the repetition of runs within this study, the results obtained showed important variation and did not produce exact results each run. By observing the range of mAP values obtained from multiple runs of all the model configurations, there were differences between 2% and 10%. Achieving a small gain in mAP performance is often considered an achievement; for example, Xia *et al*. (2018) reported a 1.7% improvement from their LR finetuning experimentation and a 3.72% performance potential of their custom model when reviewed against a VGG16 CNN model. It appears perplexing that such an opportunity to exploit the possibility of performance gains is not reported or investigated.

5.5 mAP values

Using the MSCOCO metric system, the model that produced the highest mAP value (0.751) was configuration framework FR, Image set C, Learning rate 0.005 and IOU 0.7. If following the protocols set out by the majority of literature within computer science, this configuration would be considered the most optimum (Girshick *et al*. 2014; Liu *et al*. 2016; Ren *et al*. 2016, 2017; Arcos-Garcia *et al*. 2018; Shi *et al*. 2019; Bose and Kumar 2020). The MSCOCO mAP results obtained from this investigation were high when compared to other investigations. For instance, the mAP obtained by the TensorFlow team when training the Faster-RCNN and SSD finetuning checkpoints reported final MSCOCO mAP values of 0.28 and 0.24 respectively. Janahiraman and Subuhan (2019) reported mAP scores of 0.28 and 0.22 for the Faster-RCNN and SSD frameworks, respectively. Lui *et al*. (2016) produced a mAP value of 0.28 when first reporting the SSD. Ren *et al*. (2016) produced a mAP of 0.22 when first reporting the FasterRCNN framework. This are all lower than any of the mAP values obtained here from these same frameworks when using the MSCOCO metric. This could indicate that the results obtained here are due to an error produced from within the training protocols. However, the mAP obtained in those other studies have used the MSCOCO metric in unison with the MSCOCO image set as a way to benchmark their model analysis. This investigation used a custom image set which has unknown expectations and is also built on top of the work of these large image sets. Thus, it is entirely possible that the mAP values obtained here for chironomid genera detection are correct for the combination of model configurations along with the weights already produced within the finetuning checkpoint. There is unfortunately a lack of investigations that utilise their own image sets with object detection frameworks to make direct comparisons against reported MSCOCO mAP values.

5.6 Post training performance

Within object detection there is little observable post training performance protocols described in the literature. Again, while most studies report the mAP value, which is obtained during the training phase, there is little attention given to how the models perform in real time once training has ceased and the model is in its finished form. Because of this, it was decided to conduct a series of novel post training evaluations as a way of highlighting and comparing the performance potentials, which was ultimately the goal of this investigation. Firstly, a script was written that would allow the finished models to classify a batch of 90 images which consisted of 30 images from all three genera, one after the other. The results of this were then input into a 3x3 confusion matrix which could then be used to work out a series of evaluation metrics.

Typically, confusion matrix analysis is used within classification model and not within an object detection model (Xu *et al*. 2019). Classification models are designed to only make one prediction per image whereas an object detection model can classify multiple objects at any one time (Lui *et al*. 2016; Rawat *et al*. 2017). To allow this to still work effectively, it was decided to negate any multiple predictions and only accept a single prediction with the highest confidence value as the correct prediction. To make sure that all images could be classified, the confidence threshold was decreased until all subjects within each image were successfully classified. From this the accuracy score could be produced.

Interestingly, there was very little difference between any of these models in terms of accuracy. All models were able to positively classify the majority of the test images. All models achieved an accuracy between 94.4% - 97.8%. The most mistakes any model made was a total of 4 images incorrectly predicted, and the least mistakes any model made was 2 images incorrectly predicted. Moreover, almost all of the incorrect predictions across all models were the same images. two of the images belonging to *Eukieferriella* were incorrectly classified by all the models as either *Cricotopus* or *Rheotanytarsus* (Fig. 20). However, one specific configuration (SSD-A-0.005) did correctly classify the image on the right.

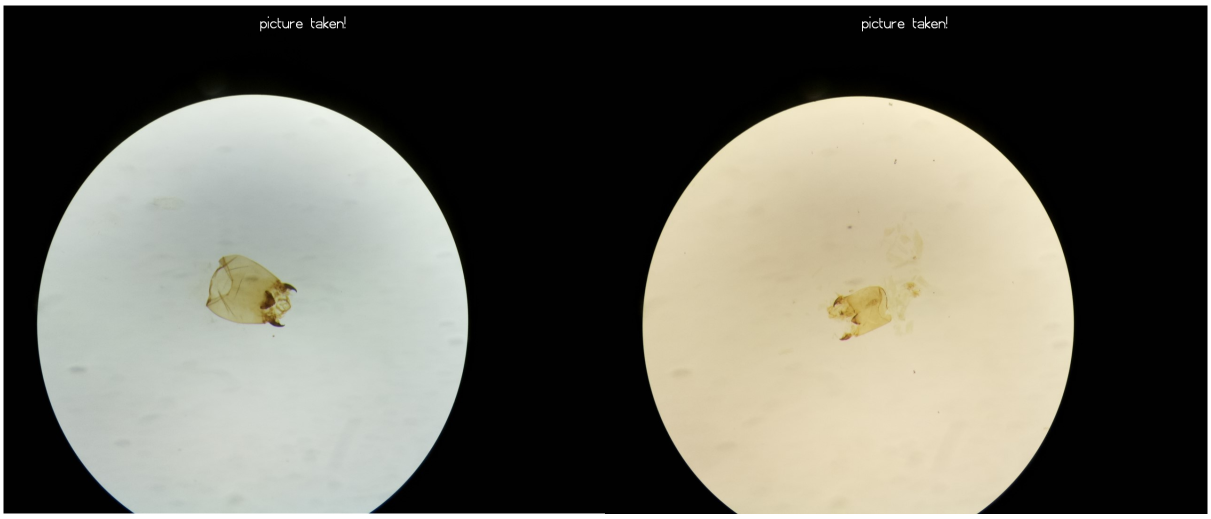


Figure 20. Images of *Eukiefferiella* incorrectly classified by all models. From left to right, image A classified as *Cricotopus*, and image B classified as *Rheotanytarsus*.

The configuration SSD using image set A and using a LR value of 0.005 gave an extremely low confidence value of 0.3, 0.7 and 0.14 across the three runs when correctly predicting the image. The fact that all 36 model configurations incorrectly classified one image as the same incorrect genus, and 33 model configurations incorrectly classified another image as the same incorrect genus, suggest that there could have been an issue with these images themselves. All these images were identified by a third party using visual methods at a previous date. Literature indicates that this method is prone to mistakes (Wiederholm 1983; Haase *et al.* 1998), so it is logical to suggest that perhaps these images were misidentified during visual identification and the models are in fact attempting to identify these correctly.

For the second phase of the post training evaluation, a review of the confidence values attributed to each image prediction was conducted. Again, any incidences where there was more than one prediction only included the prediction with the highest confidence score and all other would be negated. Using this method, three metrics were devised: 1) the average confidence score where all confidence scores for all predictions, either correct or false, were averaged out, 2) the highest confidence scores were then recorded for each model, and 3) the lowest confidence score was then recorded for each model. However, there is a difference between the model’s lowest confidence scores and the average confidence scores obtained. All the FR models across both groups achieved an average confidence score of above 99%. However, the SSD models achieved average values between 80% and 90%. The SSD models produced several extremely low confidence scores values with several models producing a value <10%, whereas the lowest confidence score produced by an FR was 42%.

These novel methods for reviewing the post-training performance of the models have shown many differences between them. It should be noted that not all the models trained were put through these evaluation tests. Only the models that achieved the best performing hyperparameter settings and their respective runs were put through post-training performance evaluation. This meant that although the configuration of FR-A-0.001 achieved the highest mAP value out of all models, it was not included within the evaluation process because the other runs within this series produced low mAP values, thus lowering the average mAP. Therefore, the models put through this evaluation were only the ones that produced the highest average mAP across the three runs, which were then used to calculate the most optimum hyperparameter setting for each framework and image set combination.

5.7 Potential use as a biomonitoring tool

For a biomonitoring tool to assist or even replace current methods, one would expect this tool to work on par or better than current methods. Chironomids can be used exclusively for the biomonitoring of ecosystems, but this method relies on the correct identification to as high of a taxonomy level as possible (García and Añón Suárez 2007; Rawal *et al*. 2018). At present, the two lead methods for the identification of chironomids are by visual identification (Wiederholm 1983; Haase *et al.* 1998) and through genetic barcoding (Valentini *et al.* 2009). To identify chironomids visually, an expert taxonomist and a microscope is required. The current best method for viewing under a microscope is to remove the head capsule of a specimen and to display it as flat as possible on a microscope slide so that all features of the head (mandibles, mentum, antennae, position of eyes, and other morphological characters) can be observed (Beckett and Lewis 1982). This method can be time consuming and is prone to mistakes (Wiederholm 1983; Haase *et al.* 1998), even from experts, however the microscope slides can be stored, reused and reanalysed for many years to come.

Identification using genetic barcoding can be done on single specimens using DNA barcoding and from environmental samples using eDNA metabarcoding. DNA barcoding involves the identification of specimens using the mitochondrial CO1 gene which is currently the best gene for the identification of invertebrates (Valentini *et al.* 2009). For chironomid larvae, this requires the destruction of the specimen to extract DNA for molecular amplification and DNA sequencing. This process does require skill to use the scientific instruments correctly, and the scientific instruments and reagents can be expensive, even if pooling of samples is possible. However, if done correctly, this process can identify easily up to the taxonomy level of species. eDNA metabarcoding can identify genera and/or species of chironomids from an environmental source, such as water meaning that specimens do not have to be collected and harvested to extract readable genetic makeup (Deiner *et al*. 2015). Again, this requires the use of expensive scientific instruments and reagents, it is time consuming and requires the use of scientific expertise.

In its present state, the proposed method using object detection and deep learning techniques created within this investigation requires chironomids to be collected on a site, euthanised and have their head capsules placed on microscope slides. These slides then need to be viewed under a microscope slide and images taken of these slides. These images then need to be transferred to a computer where they can be examined by the object detection models which will classify the chironomid head capsule to one of three genera. Three genera severely underrepresent the total 20,000 estimated species of chironomid across the globe and the 600 species estimated within the UK. To perform an accurate and meaningful biomonitoring examination using strictly chironomid larvae, each species would need to be first identified, the same object detection and deep learning technique needs to be applied, evaluated and tested, and then it can be automated for classifying new samples. . Therefore, in its current state, the proposed method of using an object detection model to classify three genera of chironomids by using images of head capsules displayed on microscope slides is unable to perform as an effective biomonitoring tool. However, as a proof of concept towards a biomonitoring tool this approach shows potential.

Once trained, despite their individual configurations and settings, each of the models were able to accurately classify between the three genera, where all the models were able to correctly classify between 86/90 and 88/90 images. It should be noted that all the specimens used for both training and testing were originally identified by visual methods, meaning that at present the models created have not surpassed or are on par with the potential of an expert taxonomist. The model's potential to factor in more genera or to identify to species level was hindered by the lack of usable data in the form of images, so there is unfortunately no way of knowing the full potential of this current method. The models were only able to work the taxonomy level of genus and not of species so there is no way of knowing whether this type of method is even capable of distinguishing at the species level. Nonetheless, there is also currently no maximum number of classes that can be added within a model, so it is entirely feasible to consider that a model could one day be trained to encompass the whole of the 20,000 species within Chironomidae.

As well as a general biomonitoring tool, this type of identification method has potential for other uses within the life sciences. Although the process of putting specimens onto a microscope slide can be laborious, it is still a frequently used method for the storage of invertebrate specimens. The Natural History Museum (NHM, London) currently houses around 80 million specimens where 2.4 million are on microscope slides where a large portion of these have not been identified (Allan *et al*. 2019). The NHM is also in the process of taking digital images of all these specimens. Having an automatic tool that is trained to identify and classify through a large selection of images of microscope slides could drastically shorten the time frame needed to identify these. Again, however, this relies on a more competent model being trained than what has been created within this investigation, but as such this project demonstrates that the automatic object detection and classification using deep learning techniques of chironomid larvae mounted on microscope slides is feasible.

5.8 Limitations

Object detection is a relatively new subject in terms of its creation and use within the scientific community. Because of this, the literature is still limited and inconsistent, especially when compared to other forms of computer vision. For example, image classification has been in development since 1989 (Cun *et al*. 1989) and contains less coding components than object detection dictating its use and finetuning. When creating object detection models, the various components work in an almost modular approach which can be constructed by different object detection frameworks, their internal convoluted neural networks, and the evaluation protocols. Each will require its own specific finetuning requirements, and this has created a range of hyperparameters that can only be finetuned by a trial-and-error process. As pointed out, there are several important hyperparameters that have no universal value, and these can be instrumental in a model performing at its optimum. The integration of object detection in other disciplines suffers even greater than the study of it within its own discipline as there is even less literature and implementation guidance available. Computer science literature has created its own evaluation processes that have been adopted within its own field but have apparently moved away from evaluation processes used by other disciplines. The computer sciences also make little effort to report how their models perform once fully trained when used in real world instances and therefore there are no evaluation processes to accompany this. Geirhos *et al*. (2020) suggest that this is due to a phenomenon called ‘shortcut learning’ wherein computer scientists have created their own benchmarking systems but have invested little attention outside of these systems and how real-world performances of their object detection models actually compare against the scoring systems created.

Within this investigation, there were many inconsistencies within the results obtained, particularly from the significance examinations. While there was clear evidence to support that the FR frameworks reliably produced a significant higher mAP value, there is little evidence to justify the impact that the different image sets had on the models. While there was justification for the learning rate values that were used, there was little evidence for the effect that the different IOU values created. When considering the hyperparameters picked for this investigation, a small selection of values was trialled, however, there is a large selection of numerical values that could have theoretically been input. There are also many other hyperparameters that could have been examined for finetuning (*e.g*. batch size and epoch), however, the time needed to investigate this would have been long.

This investigation used an object detection system to classify single subjects set within an image exclusively. While object detection can classify images, it also localises them within the image itself which gives it the potential to classify multiple subjects at any one time and can do so in real time over a live video feed. However, none of these extra features of an object detection system were utilised within this investigation and could have been completed using an image classification system. Image classification systems use a CNN much like an object detection system but then do not have to perform a localisation task. There is unfortunately a lack of research into whether the localisation factor could dimmish the potential of an object detection to classify on par with an image classification system classifying on the same data set. As mentioned, there is also a lack of post-training performance analysis for object detection systems and their ability to correctly classify. Therefore, this investigation had to adopt and alter a technique used for image classification analysis. Despite all this, the approach using the object detection systems within this investigation did create models that had a high classification accuracy and rate.

It should be noted that this investigation did not take on ecological assessments using the final models. While these models have the potential to classify different genera of chironomids, the fact that only three genera were trained dramatically decreases the model's potential to be a viable tool within ecological studies. A model with a significantly larger class data base has the potential to not only classify a larger selection of chironomids but can also be utilised for ecological assessments. However, this study can be seen as a preliminary approach or pilot study to the use of object detection and deep learning for classifying chironomid larvae, with potential for future developments and implementations in ecology.

6. Conclusion

To conclude, this investigation set out to see if an object detection model could be used to distinguish between three genera of chironomid larvae. This was intended as a proof of concept to show whether the techniques were feasible for future use by ecologists and taxonomists. To achieve this, several different deep learning techniques were used to create an optimum model that could classify the three genera. This included the comparison of two object detection frameworks, SSD and Faster-RCNN, the creation of three different image sets which followed the recommendations of various literatures and the varying of two important hyperparameters, the learning rate and the intersection over union threshold. To ascertain which configuration was most optimum, three sets of evaluation protocols were used. Firstly, the standard computer science metric system used to evaluate object detection models called the mean average precision score following the MSCOCO protocol was used. This was followed by a statistical analysis which was performed by running each model multiple times. Finally, a series of novel tests were devised to evaluate a model when used in real time.

Using the mAP system, the model configuration that produced the highest mAP value was Faster-RCNN, Image set C, learning rate 0.005 and IOU 0.7 with a value of 0.751. Moreover, the object detection framework Faster-RCNN performed generally better than the SSD framework, however, the image sets and hyperparameters showed inconsistencies and produced evidence of nonsignificance. Using this approach, all models performed almost equally high (95.5% - 97.7%) in accuracy when used in real time, however, there were major differences between the two object detection frameworks when considering their confidence scores given to each prediction, wherein the Faster-RCNN framework produced average confidence scores of >99% while the SSD framework produced average scores of <90%. By combining all these factors, the most important was the choice of object detection framework.

As expected, the Faster-RCNN framework performed better than SSD in almost all evaluations. However, it must be noted that both frameworks performed almost identically when being utilised as a simple detector. The Faster-RCNN framework is considered more accurate than other frameworks but requires more computational power to utilise. The SSD, while not considered as accurate as Faster-RCNN, it requires extensively less computational power to run (it is considered as 'lite' programming) and can be streamlined to run on low powered devices. Both frameworks therefore create different avenues for this type of project to move into. If one wishes to create a powerful, confident, and accurate automatic classifier, then the use of a powerful workstation would be required to run the Faster-RCNN framework. However, if a more mobile device or lite approach was required, so that assessments could be utilised on site for example, then the SSD framework could be used which while still accurate, is less confident in its own predictions.

The intention of this study was to create a cost-effective and fast-working computer-based model that could act as an identification tool to aid or replace more traditional methods such as the visual identification through morphology or by using molecular methods of identification. Visual identification has been used for many years and is still being used today. However, it can take a human many years to gain the necessary visual and taxonomic skills to be effective and be considered an expert within the field. Even then, this method is considered slow, laborious, and prone to misidentification. DNA barcoding has the potential to produce highly accurate results if voucher specimens and a DNA library are already available. However, the instruments needed for DNA barcoding require experience to use, the many stages involve long time to complete, and the whole process incurs many different monetary requirements.

The new deep learning method proposed here, which involves a trained object detection model, can classify images in <1s, can be executed simply with little computer training, and can do the identification automatically with high accuracy (>97% accuracy). While the initial stages require the use of a costly workstation and requires an expert to create optimum training conditions, once completed the model could potentially be used by anyone with access to a computer. This automatic computer model, when paired with a camera device, such as an affordable USB camera, could be used in real time and could identify chironomid larvae specimens just by passing them in front of the camera feed rather than having to work exclusively with digital images.

However, it should be mentioned that this demonstration only uses a very small fraction of the total chironomid diversity, where only three genera were utilised for detection out of an estimated 200+ genera worldwide and did not attempt to distinguish between species taxonomy level, where there is an estimated 20,000+ species worldwide. There are other models already in existence that can distinguish between large amounts of individual subjects (*i.e*. PlantSnap with its database of 610,000+ individual possible detections), which suggests that this type of approach could potentially be upscaled to include the whole diversity of genera and species of the taxonomically rich family Chironomidae.

Computer vision models, and in particular, the use of object detection using deep learning techniques is still in its infancy for their use in ecological sciences. However, this study is a relevant demonstration on how it can be applied for the rapid identification of taxonomically challenging organisms. It is envisaged that future work in object detection will open new opportunities for biological diversity and biomonitoring, not only of chironomids but also other group of freshwater organisms.

7. References

Abraham, A. (2005). Artificial neural networks. In: Sydenham PH, Thorn R (eds). *Handbook of measuring system design*. John Wiley and Sons. USA

Afif, M., Ayachi, R., Said, Y., *et al. (*2020). An evaluation of RetinaNet on indoor object detection for blind and visually impaired persons assistance navigation. *Neural Process Lett*, *51*, 2265–2279.

Al-Azzo, F. Taqi, A. Milanova, M. (2018). Human related-health actions detection using Android Camera based on TensorFlow Object Detection API. *Int J Adv Comput Sci Appl*, 9, 9-23.

Aldridge, C., Jones, P., Gibson, S., *et al.* (1977). Automated microbiological detection/identification system. *J Clin Microbiol, 6*, 406–413.

Allan, E., Livermore, L., Price, B., *et al*. (2019). A novel automated mass digitisation workflow for natural history microscope slides. *Biodivers Data J,* 7: e32342.

Al-Saffar, A., Tao, H., Talab, M. (2017). Review of deep convolution neural network in image classification. *International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET),* 26-31.

Amer, M., Maul, T. (2019). A review of modularization techniques in artificial neural networks. *Artif Intell Rev*, 52, 527-561.

Anaconda (2020). Anaconda website. Available at https://www.anaconda.com. Accessed: 1st November 2020.

Arcos-García, Á., Álvarez-García, J., Soria-Morillo, L. (2018). Evaluation of deep neural networks for traffic sign detection systems. *Neurocomputing*, 316, 332-344.

Avenant, N. (2011). The potential utility of rodents and other small mammals as indicators of ecosystem "integrity" of South African grasslands. *Wildl Res*. 38, 626-639.

Batmaz, Z., Yurekli, A., Bilge, A., *et al*. (2019). A review on deep learning for recommender systems: challenges and remedies. *Artif Intell Rev*. 52.

Bebis, G., Egbert, D., Shah, M. (2003). Review of computer vision education. *IEEE Trans Educ*. 46, 2-21.

Beckett, C. and Lewis, P. (1982). An Efficient Procedure for Slide Mounting of Larval Chironomids. *Trans Am Microsc Soc*. 101, 96.

Benson, D., Cavanaugh, M., Clark, K., *et al*. (2013). GenBank. *Nucleic Acids Res*. 41(D1), 36-42.

Bentler, P. and Satorra, A. (2010). Testing Model Nesting and Equivalence. *Psy Meth*. 15, 111–123.

Bergstra, J., Bengio, Y. (2012). Random search for hyper-parameter optimization. *J Mach Learn Res*. 13, 281-305.

Beyene, A., Addis, T., Kifle, D. *et al.* (2009). Comparative study of diatoms and macroinvertebrates as indicators of severe water pollution: Case study of the Kebena and Akaki rivers in Addis Ababa, Ethiopia. *Ecol Indic*. 9, 381-392.

Biggs, J., Williams, P., Whitfield, M., Fox, G. & Nicolet, P. (2000). Biological techniques of still water quality assessment. 3. Method development. Environment Agency R & D Technical report E110. Environment Agency. Bristol. UK.

Bloice, M., Stocker, C., and Holzinger, A. (2017). Augmentor: an image augmentation library for machine learning. *J Open Source Softw*, 2, 432.

Bondi, E., Fang, F., Hamilton, M., *et al.* (2018). Spot poachers in action: Augmenting conservation drones with automatic detection in near real time. *32nd AAAI Conf Artif Intell AAAI,* 7741-7746.

Bose, S., Kumar, V. (2020). Efficient inception V2 based deep convolutional neural network for real-time hand action recognition. *IET Img Proc*, 14, 688-696.

Cao, X., Chai, L., Jiang, D., *et al.* (2018). Loss of biodiversity alters ecosystem function in freshwater streams: potential evidence from benthic macroinvertebrates. *Ecosphere*, 9, e02445.

Cárdenas, R., Beltrán, C., Gutiérrez, J. (2019). Small face detection using deep learning on surveillance videos. *Int J Mach Learn Comput*, 9, 189-194.

Cassey, P., Tim, P., Blackburn, P. (2006). Reproducibility and repeatability in ecology. *BioScience*, 56, 958–959.

Chudzik, P., Mitchell, A., Alkaseem, M., *et al*. (2020). Mobile real-time grasshopper detection and data aggregation framework. *Sci Rep*, 10, 1150.

Chulu, F., Phiri, J., Nkunika, P., *et al*. (2019). A convolutional neural network for automatic identification and classification of Fall Army Worm Moth. *Int J Adv Comput Sci Appl,* 10, 112-118.

Coates, A., Baumstarck, P., Le, Q., *et al*. (2009). Scalable learning for object detection with GPU hardware. *IEEE/RSJ Int Conf Intell Robot Syst IROS,* 4287-4293.

Costa, L., Zalmon, I., Fanini, L., *et al*. (2020). Macroinvertebrates as indicators of human disturbances on sandy beaches: A global review. *Ecol Indic,* 118, 106764.

Cun, Y., Guyon, I., Jackel, L., *et al*. (1989). Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning. *IEEE Com Mag*. *27*,41-46.

Deiner, K., Bik, H., Mächler, E., *et al.* (2017). Environmental DNA metabarcoding: Transforming how we survey animal and plant communities. *Mol Ecol,* 26, 58725895.

Deiner, K., Walser, J., Mächler, E., *et al.* (2015). Choice of capture and extraction methods affect detection of freshwater biodiversity from environmental DNA. *Biol Conserv,* 183, 53-63.

Doncaster, C.,.and Davey, A. (2007). *Analysis of variance and covariance: how to choose and construct models for the life sciences*. Cambridge University Press. UK.

Everingham, M., Van Gool, L., Williams, C., *et al.* (2009). The pascal visual object classes (VOC) challenge. *Int J Comput Vis*, 88, 303-338.

Favorskaya, M. and Pakhirka, A. (2019). Animal species recognition in the wildlife based on muzzle and shape features using joint CNN. *Proc Comput Sci*, 159, 933-942.

Fegraus. E., Lin, K., Ahumada, J. *et al.* (2011). Data acquisition and management software for camera trap data: a case study from the TEAM Network. *Ecol Inform,* 6, 345–353.

Ferdoush, S. and Li, X., (2014). Wireless sensor network system design using Raspberry Pi and Arduino for environmental monitoring applications. *Procedia Comput Sci,* 34,103110.

Ferrington, L. (2008). Global diversity of non-biting midges (Chironomidae; Insecta-Diptera) in freshwater. *Hydrobiologia*, 595, 447–455.

Frouz, J., Matěna, J., Ali, A. (2003). Survival strategies of chironomids (Diptera: Chironomidae) living in temporary habitats: A review. *Eur J Entomol,* 100, 459465.

García, P.E., Añón Suárez, D. (2007). Community structure and phenology of chironomids (Insecta: Chironomidae) in a Patagonian Andean stream. *Limnologica,* 37, 109117.

Geirhos, R., Jacobsen, J., Michaelis, C., *et al.* (2020). Shortcut learning in deep neural networks. *Nat Mach Intell*. 2, 665-673.

Gerbeaux P., Finlayson C., van Dam A. (2016). Wetland Classification: Overview. In: Finlayson C. et al. (eds). *The Wetland Book*. Springer, Dordrecht, The Netherlands.

Girshick, R., Donahue, J., Darrell, T., *et al*. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit,* 580-587.

Goldberg, C., Turner, C., Deiner, K., *et al.* (2016). Critical considerations for the application of environmental DNA methods to detect aquatic species. *Methods Ecol Evol,* 7, 1299-1307.

Goldsborough, P. (2016). A Tour of TensorFlow. Available at http://arxiv.org/abs/1610.01178. Accessed: 1st November 2020.

Gavrila, D., & Munder, S. (2007). Multi-cue pedestrian detection and tracking from a moving vehicle. *Int J Comput Vis,* 73, 41–59.

Haase, P., Murray-Bligh, J., Lohse, S, *et al*. (2006). Assessing the impact of errors in sorting and identifying macroinvertebrate samples. *Hydrobiologia,* 566, 505-521.

Hall, D., Dayoub, F., Skinner, J., *et al*. (2020). Probabilistic object detection: Definition and evaluation. *Proc - 2020 IEEE Winter Conf Appl Comput Vision, WACV*. 1020-1029.

Heidarinia, N., Harounabadi, A. (2014). An intelligent anti-money laundering method for detecting risky users in the banking systems. *Int J Comput Appl*, 97, 35-39.

Holmes, D., Moody, P., Dine, D., *et al*. (2016). *Research methods for the bioscience*s. Oxford University Press, Oxford, UK.

Hoque, S., Azhar, M., Deravi, F. (2011). Zoometrics-biometric identification of wildlife using natural body marks. *Int J Bio-Science Biotechnology*, 3, 45-54.

Howard, A., Sandler, M., Chen, B., *et al*. (2019). Searching for mobileNetV3. *Proc IEEE Int Conf Comput Vis*, 1314-1324.

Howard, A., Zhu, M., Chen, B., *et al.* (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:170404861.

Huang, J., Rathod, V., Sun, C., *et al*. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *Proc - 30th IEEE Conf Comput Vis Pat Rec, CVPR,* 296-3305.

Hughes, J. (2019). *Freshwater ecology and conservation: Approaches and techniques*. Oxford University Press, Oxford, UK.

Hussain, Z., Gimenez, F., Yi, D., *et al*. (2017). Differential Data Augmentation Techniques for Medical Imaging Classification Tasks. *AMIA. Annu Symp proceedings AMIA Symp*. 979-984.

Ikeuchi, K. (2014). *Computer Vision: A Reference Guide.* Springer Nature, New York Switzerland.

Jain, S., Vaibhav, A., Goyal, L. (2014). Raspberry Pi based interactive home automation system through E-mail. *Int Conf on Relia Opt Inf Tech (ICROIT)*, 277-280.

Janahiraman, T., Subuhan, M. (2019). Traffic light detection using tensorflow object detection framework. *2019 IEEE 9th Int Conf Syst Eng Technol ICSET 2019 - Proceeding*. 108-113.

Kermany, D., Goldbaum, M., Cai, W., *et al.* (2018). Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell,* 172, 1122-1131.

Khan, A., Sohail, A., Zahoora, U., *et al.* (2020). A survey of the recent architectures of deep convolutional neural networks. *Artif Intell Rev*. 1-70.

Krizhevsky, A., Sutskever, I., Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. *Adv Neur Inf Proc Sys*, 25.

Landgrebe, T., Paclik, P., Duin, R., *et al*. (2006). Precision-Recall Operating Characteristic (P-ROC) curves in imprecise environments. *Proc - Int Conf Pattern Recognit*. 4, 123127.

Lang, B., Medeiros, A., Worsley, A., *et al*. (2018). Influence of industrial activity and pollution on the paleoclimate reconstruction from a eutrophic lake in lowland England, UK. *J Paleolimnol*. 59, 397-410.

Laurindo da Silva, F., Pinho, L., Wiedenbrug, S., *et al*. (2018). Family Chironomidae. *Thorp and Covich's Freshwater Invertebrates: Ecology and General Biology*;. Elsevier Science Publishing Co Inc, USA.

Lencioni, V., Marziali, L., Rossaro, B. (2012). Chironomids as bioindicators of environmental quality in mountain springs. *Freshw Sci*, 31, 525-541.

Lim, J., Salakhutdinov, R., and Torralba, A. (2011). Transfer learning by borrowing examples for multiclass object detection. *Adv Neural Inf Process Syst 24 25th Annu Conf Neural Inf Process Syst 2011,*1-9.

Lin, T., Zitnick, C., Doll, P. (2015). Microsoft COCO: Common Objects in Context. *ECCV*, 1-15.

Liu, J., Zhang, S., Wang, S., *et al*. (2016). Multispectral deep neural networks for pedestrian detection. *Br Mach Vis Conf 2016, BMVC 2016*. 73, 1-73.

Liu, W., Anguelov, D., Erhan, D., *et al.* (2016). SSD: Single Shot MultiBox Detector. *Comp Vis*, 21-37.

Luoto, T. (2011). The relationship between water quality and chironomid distribution in Finland - A new assemblage-based tool for assessments of long-term nutrient dynamics. *Ecol Indic*. 11, 255-262.

Ma, Z., Liu, Y., Liu, X., *et al*. (2019). Privacy-Preserving Outsourced Speech Recognition for Smart IoT Devices. *IEEE IOT J*, *6*, 8406–8420.

Meier, R., Shiyang, K., Vaidya, G., *et al*. (2006). DNA barcoding and taxonomy in Diptera: A tale of high intraspecific variability and low identification success. *Syst Biol*. 55, 715-728.

Morse, J., Bae, Y., Munkhjargal, G., *et al*. (2007). Freshwater biomonitoring with macroinvertebrates in East Asia. *Front Ecol Environ*. 5, 33-42.

Nadjla, C., Zineb, B., Lilia, F., *et al*. (2013). Environmental factors affecting the distribution of chironomid larvae of the Seybouse wadi, North-Eastern Algeria. *J Limnol*. 72, 203214.

Naidoo, R., Fisher, B., Manica, A., *et al*. (2016). Estimating economic losses to tourism in Africa from the illegal killing of elephants. *Nat Commun*. 7, 1-9.

Nam, N., Hung, P. (2018). Pest detection on traps using deep convolutional neural networks. *ACM Int Conf Proceeding Ser*. 33-38.

Nazarova, L., Pestryakova, L., Ushnitskaya, L., *et al.* (2008). Chironomids (Diptera: Chironomidae) in lakes of central Yakutia and their indicative potential for paleoclimatic research. *Contemp Probl Ecol*. 1, 335-345.

Nazarova, L., Subetto, D., Syrykh, L., *et al*. (2018). Reconstructions of Paleoecological and Paleoclimatic Conditions of the Late Pleistocene and Holocene according to the Results of Chironomid Analysis of Sediments from Medvedevskoe Lake (Karelian Isthmus). *Dokl Earth Sci*. 480, 710-714.

Nicacio, G., Juen, L., (2015). Chironomids as indicators in freshwater ecosystems: An assessment of the literature. *Insect Conserv Divers*. 8, 393-403.

Obermeyer, Z., Emanuel, E.J. (2016). Predicting the Future - Big Data, Machine Learning, and Clinical Medicine. *N Engl J Med*, *375*, 1216–1219.

Opelt, A., Pinz, A., Zisserman, A. (2006). A Boundary-Fragment-Model for Object Detection, *ECCV 2006*.

Orendt, C. (1999). Chironomids as bioindicators in acidified streams: A contribution to the acidity tolerance of chironomid species with a classification in sensitivity classes. *Int Rev Hydrobiol*. 84, 439-449.

Pagnutti, M., Ryan, R., Cazenavette, G., *et al*. (2017). Laying the foundation to use Raspberry Pi 3 V2 camera module imagery for scientific and engineering purposes. *J Electron Img,* 26.

Pang, B., Nijkamp, E., and Wu, Y. (2020). Deep Learning with TensorFlow: A Review. *J Educ Behav Stat*. 45, 227-248.

Perez, F., Vasconcelos, C., Avila, S., *et al*. (2018) Data Augmentation for Skin Lesion Analysis. *ISIC 2018.* 11041

Pickek, S., Heuser, A., Jovic, A., *et al.* (2019) The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations. *IACR TCHES* 1, 1-29.

Pinder, L. (1986). Biology of freshwater Chironomidae. Ann Rev Ento 31, 1-23

PlantSnap (2020). PlantSnap website. Available at https://www.plantsnap.com. Accessed: 14th October 2020.

Probst, P., Boulesteix, A., Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *J Mach Learn Res*. 20, 1-22.

Raiman, J., Zhang, S., and Wolski, F. (2019). Long-Term Planning and Situational Awareness in OpenAI Five. *Mach Learn CS*, 2-9.

Raspberry Pi (2020) Raspberry Pi website. Available at pihttps://www.raspberrypi.org/about/9.11.20. Accessed: 1st November 2020

Rawal, D., Verma, H., Prajapat, G. (2018). Ecological and economic importance of Chironomids (Diptera). *J Emerg Technol* 5

Rawat W, Wang Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput*. 29, 2352-2449.

Redmon, J., Farhadi, A. (2018) YOLO v.3. *Tech Rep*. 1-6.

Ren, S., He, K., Girshick, R., *et al.* (2017). Object detection networks on convolutional feature maps. *IEEE Trans Pattern Anal Mach Intell*. 39, 1476-1481.

Reynaud, D., Mishler, B., Neal-kababick, J., *et al*. (2015). The Capabilities and Limitations of DNA Barcoding of Botanical Dietary Supplements. 1-13.

Ríos-Touma, B., Acosta, R., Prat, N. (2014). The Andean biotic index (ABI): Revised tolerance to pollution values for macroinvertebrate families and index performance evaluation. *Rev Biol Trop*. 62, 249-273.

Ruppert, K., Kline, R., Rahman, M. (2019). Past, present, and future perspectives of environmental DNA (eDNA) metabarcoding: A systematic review in methods, monitoring, and applications of global eDNA. *Glob Ecol Conserv*. 17

Russakovsky, O., Deng, J., Su, H., *et al.* (2015). ImageNet Large Scale Visual Recognition Challenge. *Int J Comput Vis*. 115, 211-252.

Rybski, D. Huber, D. D. Morris and R. Hoffman, "Visual classification of coarse vehicle orientation using Histogram of Oriented Gradients features," *2010 IEEE Intt Veh Sym*, 921-928,

Sabottke, C., Spieler, B. (2020). The Effect of Image Resolution on Deep Learning in Radiography. *Radiol Artif Intell*. 2.

Salkind, N. (2010). Random selection. *Ency res des*. 1, 1214-1216.

Sandler, M., Howard, A., Zhu, M., *et al*. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit*. 4510-4520.

Sharma, K., Chowdhary, S. (2011). Macroinvertebrate assemblages as biological indicators of pollution in a Central Himalayan River, Tawi (J & K). *Int J Biodivers Conserv*. 3, 167-174.

Shendure, J., Balasubramanian, S., Church, G., *et al*. (2017). DNA sequencing at 40: Past, present and future. *Nature*. 550, 345-353.

Shi, W., Bao, S., Tan, D. (2019). FFESSD: An accurate and efficient single-shot detector for target detection. *Appl Sci*. 9.

Shijie, J., Ping, W., Peiyi, J., *et al*. (2017). Research on data augmentation for image classification based on convolution neural networks. *Chinese Autom Congr CAC.* 4165-4170.

Smart, R., Whiting, M., Twine, W. (2005). Lizards and landscapes: Integrating field surveys and interviews to assess the impact of human disturbance on lizard assemblages and selected reptiles in a savanna in South Africa. *Biol Conserv*. 122, 23-31.

Srivastava, H., Hinton, G., Krizhevsky, A., *et al*. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J Mach Lrn Res (1929-1958

Stager, C., Davis, J. (1992). Automated systems for identification of microorganisms. *Clin Microbiol Rev*. 5, 302-327.

Stein, E., Martinez, M., Stiles, S., *et al*. (2014). Is DNA barcoding actually cheaper and faster than traditional morphological methods: Results from a survey of freshwater bioassessment efforts in the United States? *PLoS One*; 9.

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications (1st. ed.).* Springer-Verlag, Berlin, Heidelberg.

TensorFlow (2020a). TensorFlow GitHub page. Available at https://github.com/tensorflow. Accessed: 1st November 2020.

TensorFlow (2020b). TensorFlow GitHub Model Zoo. Available at https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md. Accessed: 1st November 2020.

Tensorflow (2020c) Tensorflow Website. Available at https://www.tensorflow.org. Accessed: 1st November 2020.

Thenmozhi, K., Srinivasulu Reddy, U. (2019). Crop pest classification based on deep convolutional neural network and transfer learning. *Comput Electron Agric*. 164, 104906.

Thermofisher, (2020). Available at https://www.thermofisher.com/order/catalog/product/K0722#/K0722. Accessed: 10th October, 2020.

Tzutalin (2015). LabelImg. Git code. Available at https://github.com/tzutalin/labelImg. Accessed: 1st November 2020.

Uherek, C., Pinto Gouveia, F. (2014). Biological monitoring using macroinvertebrates as bioindicators of water quality of maroaga stream in the maroaga cave system, Presidente Figueiredo, Amazon, Brazil. *Int J Ecol*.

Valan, M., Makonyi, K., Maki, A., *et al*. (2019). Automated Taxonomic Identification of Insects with Expert-Level Accuracy Using Effective Feature Transfer from Convolutional Networks. *Syst Biol*. 68, 876-895.

Valentini, A., Pompanon, F., Taberlet, P. (2009). DNA barcoding for ecologists. *Trends Ecol Evol*. 24, 110-117.

Varma, R., Bressler, N.M., Doan, Q.V., *et al*. (2014). Prevalence of and risk factors for diabetic macular edema in the United States. JAMA Ophthalmol. 132, 1334–1340.

Vega, R, Brooks, S.J., Hockaday, W., *et al*. (2021). Diversity of Chironomidae (Diptera) breeding in the Great Stour, Kent: baseline results from the Westgate Parks Non-biting Midge Project. *J Nat Hist*. 55, 11-12

Verma, G., Gupta, P. (2018). Wild Animal Detection Using Deep Convolutional Neural Network. *CVIP*, 704.

Waithe, D., Brown, J. M., Reglinski, K., Diez-Sevilla, I., Roberts, D., & Eggeling, C. (2020). Object detection networks and augmented reality for cellular detection in fluorescence microscopy. *J Cell Bio*, *219*.

Wiederholm, T. (1983). Chironomidae of the Holoarctic region. Keys and diagnoses. Part. 1. Larvae. *Entomolo- gica Scandinavica Supplement,* 19, 1–457.

Wong, S., Gatt, A., Stamatescu, V., *et al*. (2016). Understanding Data Augmentation for Classification: When to Warp? *2016 Int Conf Digit Image Comput Tech Appl DICTA 2016.*

Wong, W., Su, X., Li, X., *et al.* (2014). Global prevalence of age-related macular degeneration and dis- ease burden projection for 2020 and 2040: a systematic review and meta-analysis. *Lancet Glob*. 2, 106–116.

Xia, D., Chen, P., Wang, B., *et al*. (2018). Insect detection and classification based on an improved convolutional neural network. *Sns (Switzerland)*. 18, 1-12.

Xu, X., Zheng, H., Guo, Z., *et al*. (2019). SDD-CNN: Small data-driven convolution neural networks for subtle roller defect inspection. *Appl Sci*. 9.

Yadav, S., and Shukla, S. "Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification. *IACC*, 2016, 78-83.

Yamashita, R., Nishio, M., Do, R., *et al*. (2018). Convolutional neural networks: an overview and application in radiology. *Insig Imag*. 9, 611-629.

Yan, J., Wang, H., Yan, M., *et al* (2019). IoU-adaptive deformable R-CNN: Make full use of IoU for multi-class object detection in remote sensing imagery. *Remote Sens*. 11, 1-22.

Yang, J., Zhao, Y., Chan, J., *et al*. (2016). Hyperspectral image classification using twochannel deep convolutional neural network. *Int Geosci Remote Sens Symp*. 11, 5079-5082.

Yu, X., Wang, J., Kays, R., *et al*. (2013). Automated identification of animal species in camera trap images. *Eurasip J Imag Vid Proc*. 52

Yue, S. (2017). Imbalanced Malware Images Classification: a CNN based Approach. 2017, *arXiv,* 3-7.

Zeybek, M. (2017). Macroinvertebrate-based biotic indices for evaluating the water quality of Kargı stream (Antalya, Turkey). *Turkish J Zool*. 41. 476-486.

Zhang, Z., Wang, Y., Zhang, J., *et al*. (2019). Comparison of multiple feature extractors on Faster RCNN for breast tumor detection. *8th Int Symp Next Gener Electron ISNE 2019*. 1-4.

Zhao, Z.Q., Zheng, P., Xu, S.T. and Wu, X., 2019. Object detection with deep learning: A review. *IEEE Trans Neural Netw Learn Syst,* 30, 3212-3232.

Živić, I., Živić, M., Bjelanović, K., *et al*. (2014). Global warming effects on benthic macroinvertebrates: A model case study from a small geothermal stream. *Hydrobiologia*, 732, 147-159.