Optimizing Data Diversity for Accurate Prediction Models: Insights from a Multi-Class Sheep Image Data-Set Analysis towards Autonomous Shepherding

Brandon C. Colelough,
School of Engineering and Information Technology, University of New South Wales, Australia

***Abstract*—Real-time object detection and classification were investigated for the detection and classification of sheep. This project aimed to detect and classify sheep targets from a video stream through a real-time system. The motivation for the use of sheep as targets is so that this research could be further used for the Shepherding of sheep through artificially intelligent drones featuring his shepherding algorithm. An image data set of ˜5000 sheep images was gathered and labelled. This data set was then combined with extracts of the COCO data set. 4 Convolutional neural networks (CNN) were then configured from the Darknet library for object classification and were utilised with the YOLO (You Only Look Once - YOLO) algorithm for object detection. The key findings of this research demonstrated that a combined image dataset, which included a 10-class segment from the COCO dataset, generated the most effective predictive model. However, adding only the sheep category from the COCO dataset to an established dataset of 5000 images decreased the accuracy of the resulting predictive model. The study concluded that enhancing a dataset improves the accuracy of a model trained on it only if the dataset is sufficiently diverse. Furthermore, it was observed that continuing the training process with a set of weights, without adequate information about the optimizer's state after initial training, leads to a less effective model.**

I. Introduction

Object detection and classification is a form of image processing wherein a deep neural network is used to recognise specific items within a defined frame. Machine learning is conducted through the use of an artificial neural network (ANN). Conceptually, an artificial neural network is designed to simulate the interconnected neurons of a mind [1]. Furthermore, CNNs were modelled after the theory of how the mammalian visual cortex detects objects [2]. This specific type of network is very efficient with image processing. A deep neural network is the implementation of any neural network that has many layers between the input and output neurons of that network. Those layers are known as a networks hidden layers [3]. Lastly, YOLO (you only look once) is a method of object detection wherein objects within a frame can be rapidly detected [14]. Through the implementation of the YOLO method coupled with a deep neural network with convolutional traits, a system can rapidly and effectively detect an object within an image frame and then classify that object in real time.

A. Terminology

This report will make use of the following terminology:

Real-Time system: A system that is both logically and temporally correct. To be logically correct is to perform all assigned tasks and functions to the specifications given and to be temporally correct is to be guaranteed that the system will complete these functions within an explicit time-frame.

Artificial Neural Network: Connectionist systems that are vaguely inspired by the biological neural networks that constitute an animal's brain. These networks are an implementation of machine learning.

Convolutional Neural Network: Hierarchical, multi-layered neural network systems that have powerful applications in image processing.

Object Detection: Object detection involves detecting instances of objects from a particular class within the frame of an image.

B. Artificial Neural Network

An ANN consists of an input layer, an output layer and a multitude of hidden layers between them. Each layer consists of a number of nodes which are set to simulate the neurons within a mind [4]. These nodes each hold an activation value. For simplicity, this can be thought of as the amount that a neuron within a brain is lit up. An ANN also contains connections between each of the layers in its network and thus every node at every layer in an ANN is connected to every node in the layers before and after it. Each connection from one node to the next within an ANN has an associated weight. The activation of each neuron in one layer depends upon the weighted sum of each connection and neuron activation from the previous layer. An activation function is applied to this weighted sum to as to compress it to be in range of the upper and lower bounds of the activation limit set for the neurons within a network. A bias can also be added to the weighted sum in order to reduce the noise of a network by eliminating nodes close to the boundary of the network's activation limit. The mathematical model for this process is as follows:

$$A = \sigma(w_1a_1 + w_2a_2 + w_3a_3 + \cdots + w_na_n - \text{BIAS}) \quad (1)$$

This activation process is conducted for each node within a layer of a network. By separating the nodes into layers and

placing connections between them we can take abstract and complex problems and break them into segments of problems from which each layer can handle a part of.

C. Machine Learning

Machine learning within the context of an ANN is the adjustment of weights and biases for the connections within that ANN. This process is done through back propagation. Back propagation aims to lower the cost function an ANN [5].

D. Cost Function:

The cost function of an ANN is a measure of how correctly that network can classify an object. The cost function of a network is the averaged sum of the cost of each layer of the network over many iterations of forward propagation through that network. A common method used to measure the cost of a networks layer is the mean squared error function. This example takes the mean of the squared difference of the found activation and true label for each node in the output layer. The cost function of an ANN using this cost equation can be described by the equation:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - X_i)^2 \tag{2}$$

Where *n* is the number of predictions made by the network. x is the output value vector and y is the output predictions vector. Once the cost function is computed, back propagation can be used to train the network appropriately.

E. Back Propagation - gradient descent:

Back propagation is the backbone of training a neural network. Back propagation is the process of optimizing the weights and biases associated with a neural network based on the cost function (function) obtained at each iteration (epoch). Optimising these weights lowers the cost of the network and therefore makes the network more reliable [6]. Gradient descent is the tool used in order to optimise these weights. Gradient descent is a first order iterative optimisation algorithm that can be used in order to find the minimum of a function. "In gradient descent one finds minimum of f(x) by repeatedly computing minima of single variable function $g(t)$" [7]. In a neural network, the activation of each node is determined by the weights and biases associated with the layers previous to that node. Thus, gradient descent must be used to determine the weights and biases that the neural network needs in order to converge the cost function to a local minimum. to do this, the negative gradient of the cost of a layer is determined. This negative gradient is then applied to the weights and biases for that layer. This process can be described by the equation:

$$B = A - \gamma \nabla f(x) \tag{3}$$

Where B is the vector constructed from the next position of the current layers weights and biases, a is the vector constructed from the current layers weights and biases, γ is the step size used for gradient descent and ∇f(x) is the gradient of the cost of the current layer. In this process, the gradient at the output layer is computed and then used to change the weights and biases for the connections associated between the output layer and the layer directly behind it. This process is then completed for each layer in the network working backwards from the output layer. When the Cost function reaches this local minimum then the network has a low error percentage and is therefore trained.

F. Stochastic gradient descent:

Gradient descent takes the whole of the input data-set for a function and computes the steepest gradient over that whole data-set accordingly. This is heavily inefficient in machine learning however as the data-sets for problems are generally very large (in the order of magnitude of hundreds of thousands). Stochastic gradient randomises a data-set and breaks the data-set up into batches of some pre-defined batch size. These batches are then used to train a network. Although the route taken to the minima for the data-set is not optimal, the time taken to find this local minima is much faster.

G. Deep Neural Network

A deep neural network is any type of neural network with a certain level of complexity. Deep neural networks can be any type of neural network with more than 2 layers i.e. a network with more than an input and output layer. The additional layers of a neural network are where the necessary mathematical processes for that specific network are conducted.

H. Object detection

Traditional neural networks such as CNN's and ANN's can only classify an image. This means that, at their output layer, the network can only indicate whether or not something was found within an image. As most image processing neural network are trained with the object to be classified centrally placed in the image and with only one occurrence of that object, the addition of other similar and dissimilar objects within an image will cause the network to fault as the complex feature map for that image will differ from the known feature map for a single object. Similarly, if the object to be classified is not centrally placed or is distant within the image then the complex feature map for this image will again vary which may decrease the likelihood of proper classification as the object may be drowned out by other features [8]. Early detection methods such as the HOG (Person) Detector worked by sliding a context window over an image and then utilising a linear support vector machine to classify objects found within the context window. With the invent of CNN's, they could be re-purposed as object detectors utilising this sliding window method with a much higher accuracy rate than the HOG method. This method was however very computationally costly as it required a pipeline of data to be run individually through a CNN. Next came the approach of a R-CNN (regional CNN). Researcher Ross Girshick and his team proposed a

selective search method wherein 'region proposals' would be identified within an image and bounding boxes placed around them. These region proposals would then be fed through a CNN for object classification. The selective search algorithm worked by grouping similar regions of pixels by colour, texture ETC. The method would take an image and generate initial sub-segmentation's of that image so that it would have many small regions grouped by colour similarity, texture similarity, size similarity, and shape compatibility etc. These regions would then be fed through a CNN to be classified. This method was much faster than the sliding window approach but still could not act as a real time system as even the fastest iteration of the R-CNN (faster R-CNN) can only run at 7 FPS. The first true solution for an object detection system that ran in real time came about with the single shot multibox detector system as the system could run at a maximum of 46 FPS. This system was however beaten by the latest in leading edge of object detection and classification technology which is the YOLO method. This method will be discussed at length throughout this paper.

II. Literature Review

A. Convolutional Neural Network- Image classification

CNNs were inspired by the Research performed by D. H. Hubel and T. N. Wiesel [9]. They proposed that cells in the visual cortex can be classified as simple, complex, or hypercomplex and that complex responses from complex cells are generated from more simplistic signals from simple cells. i.e. mammals use a hierarchical system of cells within their visual cortex to perceive the world around them [10]. Yann LeCun, a French computer scientist, took this theory for the mammalian visual cortex and developed a neural network, leNet5, based on its principles. LeCun implemented within his neural network the principles of local neuron connections, hierarchical cell layering and spatial in-variance through methods of convolution operations, shared weights and pooling/sub-sampling. In 2012, Alex krizhevsky released AlexNet which took LeCuns' CNN and further improved it through the use of rectified linear units (ReLu), its use of dropout technique which was first proposed by Geoffrey Hinton. AlexNet also used the work done in 2010 by Claudiu Ciresan and Jurgen Schmidhuber for the implementation of a CNN with the use of GPUs. Whilst CNN's can be used in other domains such as speech recognition and time series forecasting, their implementation is most efficient with image processing as they have been designed to handle such data. This design is shown through the implementation of local neuron connections through LeCuns implementation of convolutional operations in his neural network. A convolutional operation is taking two inputs and combining them to retrieve a single output. In regards to object classification, convolution is used to, at a simple level, to detect simple things such as edges and at a complex level, detect things such as faces through the hierarchical detection of simple objects. As talked about earlier, ANN have weights and biases associated with connections between nodes in a network that can be trained. CNNs' also contain these weights but they are represented as filters which are values in a 2d matrix. These filters are parsed over an image to detect certain features of that image. As one image may have many simple features that can be picked up by a simple filter, the weights of a CNN can be shared. When these filters are parsed over an image, small parts of the image are systematically analysed. A convolutional operation is applied between the input square currently being analysed and the weighted matrix (filter) that is currently being used for this operation. The end product of this convolutional operation repeated for the whole image is a feature map. This operation is then repeated for the depth of an image or, the amount of colour channels that image has. This operation is described by the following equation:

Convolution theorem for two general continuous functions:

$$h(x) = f \circledast g = \int_{-\infty}^{\infty} f(x)g(x-u)du = F^{-1}(\sqrt{2\pi}F(f)F(g)) \tag{4}$$

Convolution theorem applied for 2D discrete image data:

$$\text{Feature map} = input \circledast filter \tag{5}$$

$$= \sum_{y=0}^{columns} \left(\sum_{x=0}^{rows} inputs(x-a, y-b) \cdot filter(x,y) \right) \tag{6}$$

$$= F^{-1}\left(\sqrt{2\pi}F(input) \cdot F(filter)\right) \tag{7}$$

Where:

$\circledast$ is a convolutional operator

F is a Fourier transform

$\sqrt{2\pi}$ is the normalisation constant

Once all the simple filters of a network have been applied to all the colour channels of that image, more complex filters can be parsed over that image to detect more complex things and thus, the neural network can hierarchical structure of this neural network can be seen. After the convolutional operations are performed, the feature map is run through a Rectified linear unit (ReLu) operation. This operation essentially removes any negative values from the feature map in order to create non linearities within decision function found at the softmax layer of the network without affecting the receptive fields found within the convolution layers. This is done through the following non saturation activation function:

$$f(x) = \max(x, 0) \tag{8}$$

Once the feature map has had a ReLu function applied to it, it is passed through a sub-sampling/spatial pooling layer. This layer reduces the size of a feature map whilst maintaining its most importing information. The spatial pooling operation has many types like Max, Min, Average, Sum etc. This function defines a spatial neighbourhood and then takes the parameter defined (Max,Min etc) applies it to that neighbourhood in order to reduce the spatial neighbourhood into a single product which is defined by the spatial pooling type. For example,

a Minimum spatial pooling operation with a spatial neighbourhood of a 2x2 window of the feature map and take the minimum value for that 4x4 pixel square as its product. This compresses the image and reduces the computational power needed for its classification. The equation for this operation varies dependant on the spatial pooling type. These three operations, convolution, ReLu and Spatial pooling, are then repeated in order to learn more abstract features. Through repeating these processes the network is applying feature learning to an input image. Once the network has identified an images features, it will then classify that image based on its features. The classification process begins with the matrix that was used for convolutional operations being flattened into a single column that can be used in a traditional ANN. This flattened layer is then connected fully to a classification layer wherein the nodes will become activated dependant on the likeliness that a certain object stored within that node is detected. Lastly a SoftMax activation function is applied in the output layer of the network in order to rate the probabilities of detection for the objects known by the network. Training for a CNN is done through the same method of back propagation that an ANN goes through. The dimensions of the parameters are higher for the function in a CNN as the weights found for nodal connections are represented by matrices but the applied maths is the same.

B. YOLO - Object detection

The YOLO (You Only Look Once) method for object detection works by splitting an image into a SxS grid from which region proposals can be made from. Each cell in the SxS grid is responsible for predicting B bounding boxes. Each bounding box consists of 5 predictions which are the x and y co-ordinates of the centre of the bounding box, the width and height of the box and the confidence score for the object that is being detected in that bounding box [11]. The reason that this method is so fast is because a single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation rather than multiple parses as was done in detectors like R-CNN and SSD. YOLO the intersection over union method between the ground truth of an object and a its associated anchor box to predict its bounding box.

III. Methodology

The aim of this project is to detect a specified target (sheep) in a video stream through image processing and the utilisation of an artificially intelligent system. I have decided to build my project upon the Darknet Framework and utilise the YOLO object detection algorithm. In order for a CNN to be sufficiently trained in the detection of a specific object, a substantial image data-set (in the magnitude of thousands of images) of that object must first be obtained. After this data-set is labelled to specify where in the image the object that the CNN is being trained to detect, the labelled data-set can be utilised to train a set of weights for a configured CNN for detection of similar objects. In short, a labelled data-set of thousands of images of a single object must be obtained to train a CNN how to detect that object.

A. Obtaining a data-set

For this purpose, an android application was developed for the efficient capture of sheep images. This application worked by utilising burst shot and video to rapidly take many photos of one target from multiple angles and varying the contrast of the photo to diversify the data-set. The images captured by the burst shot and frames captured by the video were then broken up and saved as individual, labelled photos. With this application, a data-set of over 5000 images of sheep was collected.

B. Labelling the Data-set

This data-set of 5000 sheep images then needed to be properly labelled. This labelling process included drawing a bounding box around every instance of the target within the image. This bounding box data along with the objects ID was then saved to a text file. This labelling process had to be done by hand (there is not yet an automation process for this) for the entire data-set. An example of a labelled image is shown below.

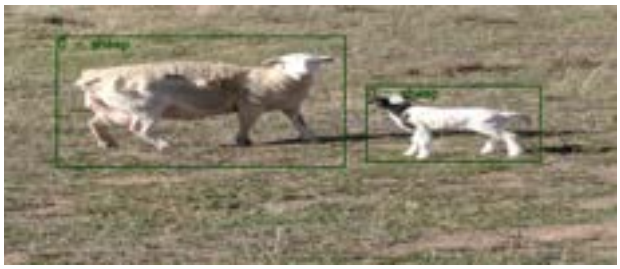


Fig. 1: example of a labelled sheep target

TABLE I: Just sheep CNN parameter configuration

Parameters	Values
Classes	1
Batch-size	64
Sub-division size	64
Channels	3
Momentum	0.9
Decay	0.0005
Angle	0
Saturation	1.5
Exposure	1.5
Hue	1
Learning rate	0.001
Burn in	0.001
Max Batches	2000
Steps	1600, 1800
Filters	18

C. Convolutional Neural Network

The CNN utilised for by the Darknet framework for the YOLO algorithm was first configured for use with a data-set with only one image. The cloned CNN from the source page

utilised the COCO (Common Objects in COntext) image data-set with 80 object classes. The last two layers in the YOLO CNN was stripped from the network and replaced with a layers written for the detection of only one object. The last layer was a fully connected SoftMax layer which worked to identify which object (if any) the CNN was currently detecting and the probability that the object was actually being detected. This layer was replaced with a layer only containing two nodes – sheep or not sheep. A configuration file was then made for the new CNN wherein the specified classes was set to 1 and the filters for each of the convolutional layers within the network was lowered for the new number of classes. The labelled sheep data-set was then run through this newly configured YOLO CNN. The images were trained with the following parameters. Table I summarises the configuration parameters utilised in training of a CNN for just sheep.

D. Optimisation with COCO data-set

After an initial testing of the weights generated from this data-set, it was decided that more data was needed from a more diverse data-set. To accommodate for this, extracts of the COCO data-set were used in order to optimise the data-set. The COCO image data-set contains over 300K images for 171 classes (200K of which are labelled). From this, 1596 images of sheep were extracted and the annotations from the COCO data-set were used in an automated process to properly label these images for use in the Darknet YOLO structure. This combined data-set of roughly 7000 images was then run through the CNN configured for just the sheep class using the same parameters.

E. Addition of 9 classes

Whilst the weights generated from this data-set gave a much more optimised result for sheep detection, it was hypothesised that a much larger data-set with similar objects of different classes would generate weights that would much more effectively classify sheep targets. The following classes were hence extracted from the COCO data-set; bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe. Over 25000 images from these classes were extracted from the COCO data-set. This data-set was again labelled through an automated process that utilised the COCO data-set annotations. A new CNN was configured for this data-set with the same process as mentioned above but with 10 classes instead of 1. Table II summarises the changes made for the configuration parameters utilised for the training of 10 classes

TABLE II: Change of parameters for addition of 9 classes

Parameters	Values
Classes	10
Max Batches	20,000
Steps	16,000, 18,000
Filters	45

F. Use of pre-trained weights

Lastly, the optimised data-set of 7000 labelled sheep images was used in an attempt to optimise the already trained weights provided by the Darknet library which was trained on 80 classes of the COCO data-set. These weights were converted into a form where they could be used as a basis for training and a new convolutional neural net was configured for 80 classes. Table III summarises the changes made for the configuration parameters utilised for the optimisation of pre-trained weights obtained from the Darknet library.

Parameters	Values
Classes	80
Max Batches	200 000
Steps	200,400,600,800,100K, 150K
Filters	255

TABLE III: Change of parameters pre-trained weights optimisation

Figure 2 shows the loss over time for training the optimised CNN trained with the optimised data-set of 10 classes with ˜30K images.

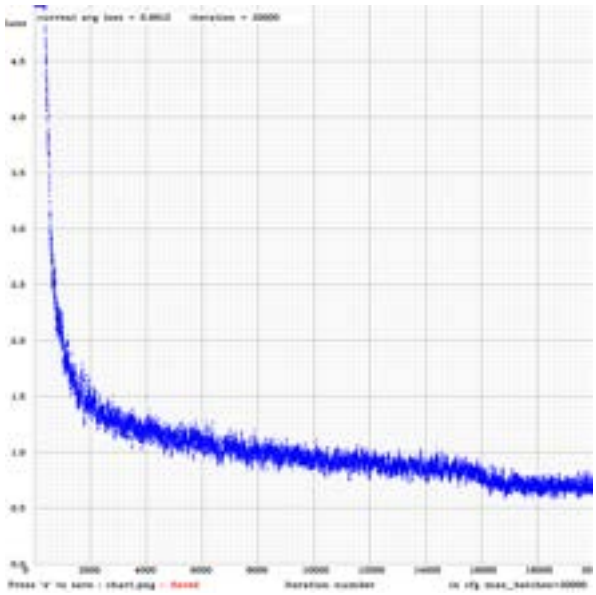


Fig. 2: Loss value VS training time of optimised CNN using 30k image dataset

IV. Experimental Design

The four sets of weights and their corresponding configured CNNs were all tested against a set of images from the imageNet data-set. Four data-sets were extracted from the imageNet library. These data-sets and their corresponding Wordnet ID's are:

- sheep: n02411705
- domestic sheep: n02413131
- rocky mountain: sheep n02415577

- Dall sheep: n02415253.

A data-set of roughly 700 sheep images per data-set was extracted from the imageNet image data-set. These images were then run through the trained weights and CNNs. A set of pre-trained weights from the Darknet library was also tested against these images as a benchmark for the generated weights. During these tests, the following parameters were observed:

- of sheep correctly detected
- Average clarity
- Average probability
- Average unclear probability
- Average time

The number of sheep correctly detected was the number of images wherein a sheep was detected. The average clarity was the average number of times a sheep was mislabelled as another class, like for example a horse or a dog. The average probability was the predicted probability that the class detected was a sheep. The average unclear probability was the predicted probability that the class detected was any class other than sheep (for the mislabelled sheep targets) and the average time was the time in milliseconds taken by the CNN to detect a sheep object in a frame. From these data-points, the mAP (mean average precision) and FPS (frames per second) was calculated. The mean average precision was calculated from two parameters. The first parameter was the precision of the results (number of targets found / number of targets presented). The second parameter was the intersection over union (IoU). An example of a ground truth and prediction box is shown figure 3. For the area of the ground truth and the prediction boxes for the detected object, the intersection over union was calculated using the formula:

$$IoU = \frac{\text{Area of intersection}}{\text{Area of union}}$$

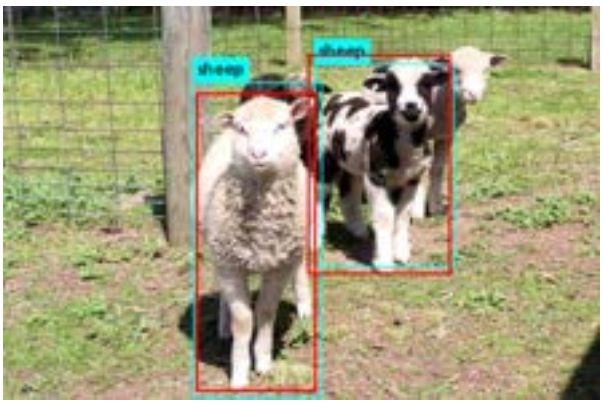


Fig. 3: Example of ground truth bounding box (Red) VS prediction bounding box (blue)

V. Results

A. Sheep detected

Figure 8 shows that Across the 5 weights and CNN's tested, it was found that the optimised data-set trained on 10 classes had, on average the greatest number of sheep detected across the 4 data-sets tested on. The next best configuration was the original data-set of just 5000 images of sheep. It is interesting to note here that the addition of 2000 images from the COCO data-set found within the optimised data-set configuration did not, on average give a higher number of targets detected. The lowest number of targets detected came from the benchmark configuration (provided by Darknet library).

B. clarity

As shown in figure 8, the original data-set and optimised data-set of course gave the best clarity as they were only trained to detect sheep and could therefore only classify sheep and could not mislabel the target being detected. The optimised pre-trained weights data-set also had a clarity of 0. I will explore this further in the discussion. The optimised data-set trained on 10 classes beat the benchmark CNN and weight configuration which is to be expected as the benchmark was trained on 80 classes meaning it would have 8X the chance of classifying an object incorrectly based on the number of choices it has to choose from alone. This clarity is highlighted in figures 10 and 11 wherein the target was detected by both configurations. However, the target was classified twice as different objects in both figure 10a and figure 11a whereas it was only classified once and as classified correctly in figure 10b and 11b.

C. Probability and unclear probability

Figure 8 shows the certainty in detection given by the configurations of CNN and weights was highest in the optimised data-set trained on multiple classes. There was an observed 10 % jump in certainty of target detection for sheep and a drop of more than 40 % for the certainty that it was incorrectly detecting other classes. The original data-set and optimised weights configurations gave a spread between the benchmark and the highest scoring data-set in their certainty in detection but the optimised data-set configuration which contained the additional 2000 images went down in clarity. There was an observed 0 across these three configurations for unclear probability as these CNN's and weights were not trained on classes other than sheep and therefore could not mislabel a target.

D. mAP

Figure 8 shows the highest mean Average precision was given by the optimised data-set trained on 10 classes configuration which was followed closely by the original data-set. The optimised data-set configuration which had the additional 2000 images from the COCO data-set decreased in mAP and had the lowest score among the 5 configurations tested. The additional precision in prediction bounding box overlap with ground truth boxes is highlighted in figures 11 and 12 where it can be seen that the benchmark CNN and weights configuration often cuts out parts of a target when it is detected whereas the optimised weights and CNN trained on 10 classes has bounding boxes that encapsulate all of the target being detected.

	Benchmark	Original Dataset	Optimised Dataset	Optimised dataset - multiple classes	Optimised pre-trained weight
Number of Images	725	725	725	725	725
Sheep found	635	649	585	668	595
average clarity	1.946280992	0	0	0.361878453	0
average probability	64.48898072	64.05939227	50.87845304	69.53867403	61.47375691
average unclear probability	113.0688705	0	0	18.27624309	0
average time (ms)	21.00484298	20.42485912	19.87487017	19.91932182	21.04206492
mAP	80.5862069	77.51724138	74.68965517	84.13793103	69.06896552
FPS	47.60806835	48.95994603	50.31479409	50.20251236	47.523853

Fig. 4: results for testing of regular sheep imageNet testing set

	Benchmark	Original Dataset	Optimised Dataset	Optimised dataset - multiple classes	Optimised pre-trained weight
Number of Images	774	774	774	774	774
Sheep found	668	731	622	724	716
average clarity	1.984496124	0	0	0.296248383	0
average probability	64.69509044	76.01681759	56.98706339	74.87192755	75.13712807
average unclear probability	15.57751938	0	0	16.30271669	0
average time (ms)	20.98963824	20.42526261	20.43685899	20.48700259	21.05457309
mAP	81.30490956	85.44444444	76.36175711	79.54005168	78.50645995
FPS	47.64255527	48.95897884	48.9311983	48.81143524	47.49561987

Fig. 5: results for testing of domestic sheep imageNet testing set

	Benchmark	Original Dataset	Optimised Dataset	Optimised dataset - multiple classes	Optimised pre-trained weight
Number of Images	747	747	747	747	747
Sheep found	578	649	498	646	620
average clarity	1.065683646	0	0	0.466487936	0
average probability	57.2613941	64.95442359	41.40348525	67.57908847	64.43565684
average unclear probability	60.74262735	0	0	25.27211796	0
average time (ms)	20.93093298	20.22356702	19.63626139	20.392937	20.95026944
mAP	65.37617135	72.88085676	58.66666667	80.47925033	70.99866131
FPS	47.77617898	49.44726115	50.92619109	49.03658557	47.73208302

Fig. 6: results for testing of rocky mountain sheep imageNet testing set

	Benchmark	Original Dataset	Optimised Dataset	Optimised dataset - multiple classes	Optimised pre-trained weight
Number of Images	630	630	630	630	630
Sheep found	473	535	418	546	495
average clarity	1.009538951	0	0	0.287758347	0
average probability	54.27503975	64.0317965	42.22009569	69.31796502	60.47058824
average unclear probability	58.51510334	0	0	14.52623211	0
average time (ms)	20.96686804	20.3157806	20.447563	20.51133704	21.16033545
mAP	74.07936508	79.92063492	61.34920635	76.66666667	71.57142857
FPS	47.69429549	49.22281942	48.90558352	48.75352581	47.25823001

Fig. 7: results for testing of Dall sheep imageNet testing set

	Benchmark	Original Dataset	Optimised Dataset	Optimised dataset - multiple classes	Optimised pre-trained weight
Number of Images	719	719	719	719	719
Sheep found	588.5	641	530.75	646	606.5
average clarity	1.501499928	0	0	0.35309328	0
average probability	60.18012625	67.26560749	47.87227434	70.32691377	65.37928251
average unclear probability	61.97603015	0	0	18.59432746	0
average time (ms)	20.97307056	20.34736734	20.09888839	20.32764961	21.05181072
mAP	75.33666322	78.94079438	67.76682132	80.20597493	72.53637884
FPS	47.68019052	49.14640717	49.75399538	49.19407895	47.50185212

Fig. 8: average results across the previous 4 result sets

E. cluster detection

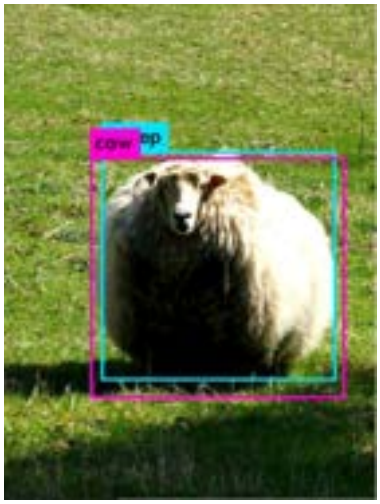
Figures 9 to 12 highlight the major differences between the benchmark CNN and weights configuration and the optimised weights and CNN trained on 10 classes. These figures show the effect of the increased clarity, mAP and detection given by the optimised weights and CNN trained over 10 classes when compared to the benchmark tests. Figures 10 and 12 highlights well the tendency of the benchmark configuration to mislabel targets and cut their bounding boxes short. Figure 12 highlights the extended precision and clarity of the optimised CNN trained on 10 classes as it has detected more targets and labelled them correctly as opposed to the benchmark configuration which did not label a single target properly. Figures 9-12 show the importance of these characteristics when classifying targets in a cluster formation as many herds of sheep are.

F. Overall observations

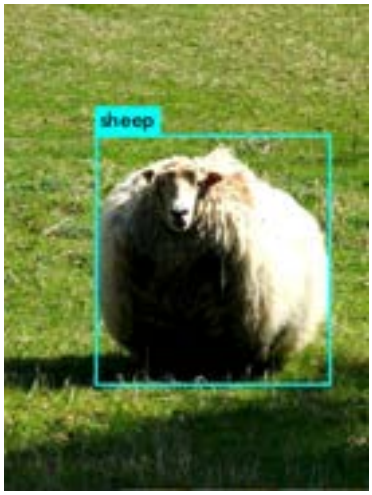
On average it was found that the optimised data-set which was trained on 10 classes had the best characteristics across all the parameters tested (disregarding 0 values). It was also observed that each configuration tested outperformed the benchmark tested for detection of sheep. Lastly, it was observed that the addition of 2000 images sourced from the COCO data-set which transitioned the original data-set to the optimised data-set actually decreased the detection results and other parameters tested.

VI. Discussion

It was found in this experiment that the addition of more data to a small data-set will improve its accuracy in detection as every configuration tested outperformed the benchmark on detection. The benchmark was trained on only 2000 images of the target to be detected which is a relatively small data-set. This correlation to data-set size and detection rate did not



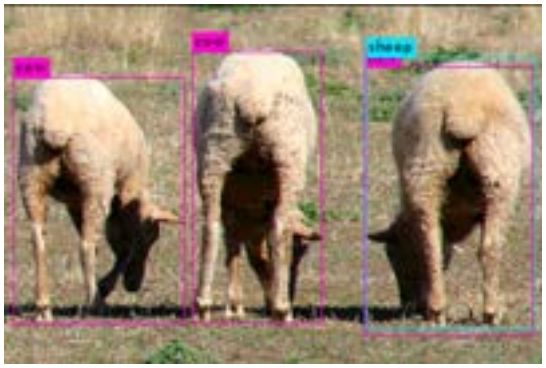
(a) Result from benchmark weights & CNN



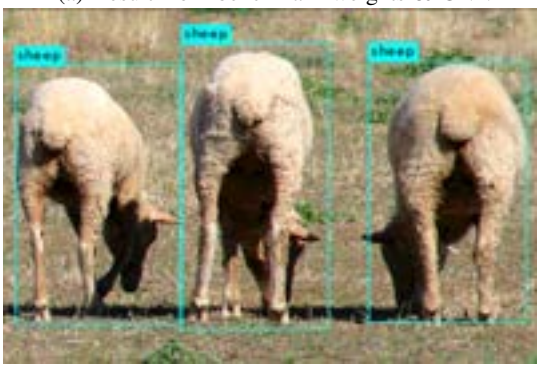
(b) Result from optimised weights & CNN

Fig. 9: Single target detection

however hold when the data-set was increased to a size of over 5000. It was found that when the original data-set was increased from 5000 to 7000 images of the target, that the detection rate, mAP and certainty all went down. I theorise that this decline in accuracy was a result of the filters (weights) used in the convolutional operations for detection became over-fitted to the objects being detected and could not handle dis-similar data such as strange or odd-looking sheep or the view of sheep in a cluster. This theory was confirmed as the same data-set of 7000 sheep images was then combined with 23000 images of 10 similar classes (such as cats, dogs etc) and this optimised data-set of 10 classes gave the best results from across all the parameters tested on. This means that a larger data-set will give more accurate results but only if that data-set contains a wide variety of diverse images. This will ensure that the convolutional filters used for detection do not over-fit to a certain shape and will therefore be better at detecting a wider range of similar targets over a more diverse field of vision. The end result of this is the more accurate detection



(a) Result from benchmark weights & CNN



(b) Result from optimised weights & CNN trained with 10 classes

Fig. 10: Multiple target detection

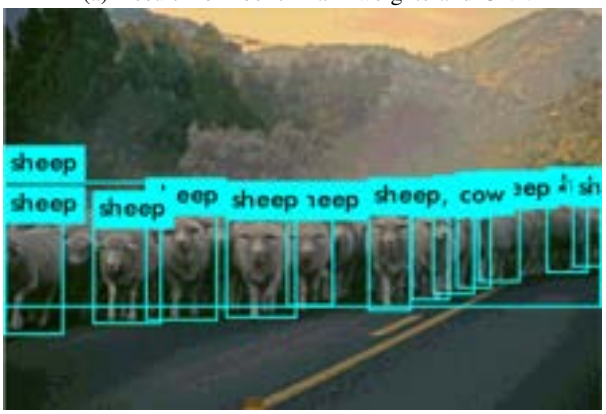
(a) Result from benchmark weights & CNN

(b) Result from optimised weights & CNN trained with 10 classes

Fig. 11: Clustered target detection

and classification of targets in a wider range of environments. The optimised pre-trained weights CNN and weights resulted in a 0 value for both average clarity and average unclear probability. This configuration was set up for 80 classes and therefore should have at some point received a value larger than 0 for these parameters. I believe that this issue arose from the optimiser used in the back-propagation process for training starting at a point wherein it had no knowledge of the previous optimisers' momentum and LR. [15]. This class was trained using the same data-set as the optimised data-set (7000 images,

(a) Result from benchmark weights and CNN



(b) Result from optimised weights & CNN trained with 10 classes

Fig. 12: Clustered target detection

5000 from my original data-set and 2000 from COCO data-set) but it gave differing results from the optimised data-set which means that the weights produced by the configured CNN were different to the weights produced by the optimised data-set. Therefore, starting the training with the pre-trained weights from the Darknet library did have an effect on the training of the CNN and the end result for the weights produced but, this effect actually hindered the target detection of the weights produced and did not allow for the recognition of the other classes that the pre-trained weights were trained on. The conclusion drawn from this is continuing training from a set of weights without the information surrounding the training for those weights results in an end product that is not anywhere near as accurate or useful as if the weights were produced from the original data-set. The CNN and weights produced from the optimised data-set trained on 10 classes had the highest detection rate. This configuration also gave the best clarity and precision. This clarity and precision in classification is a vital component of detection. The uncertainty in classification would lead to further issues with autonomous algorithms if this system was utilised in an autonomous herding drone as is intended. It is important for the system to produce reliable and accurate data so that this data can then be utilised by a separate autonomous system to produce the right decisions.

VII. Conclusion

In this study it was found that the addition to a data-set increases the accuracy of the model trained from that data-set only of the data-set is diverse. It was also found that the addition of excess data to a data-set that is not diverse can cause over-fitting and ultimately decrease the effectiveness of the prediction model produced from the training over that data-set. Lastly, it was found that continuing training from a set of weights with no information on the state of the optimiser at the conclusion of the training of that set of weights will lead to an ineffective model and it a model will therefore be more effective if it is trained from the original data-set and not as a continuation of the weights produced from it.

References

[1] Rich, E. (1983). *Artificial Intelligence.*.New York: McGraw-Hill.
[2] Y LeCun, Y Bengio (1995). *The handbook of brain theory and neural networks*.
[3] Y LeCun, Y Bengio, G Hinton (2015). *Deep Learning = nature*.
[4] Samarasinghe, S., and Shanmuganathan, S. (2016). *Artificial Neural Network Modelling (1st ed 2016.)*. Cham: Springer International Publishing: Imprint: Springer.
[5] Mitchell, T. M. (1997). *Machine learning.*. New York: McGraw-Hill.
[6] Orr, G. Muller, K. (1998). *Neural Networks: tricks of the trade.* Springer, 1998
[7] Wooley, M. (2019). *A Tour of OPtimisation*, Lecture Notes, Engineering Research 2A ZEIT2901. UNSW Canberra (ADFA), delivered March 4, 2019.
[8] Wu, J. M., and Rehg, J. M. (2012). *Object detection. In Ensemble Machine Learning: Methods and Applications*, (pp. 225–250).Springer US.Available at: https://doi.org/10.1007/9781441993267_8 (Accessed: 5 April 2019).
[9] Yamashita, R., Nishio, M., Do, R.K.G. and Togashi, K. (2018). *Convolutional neural networks: an overview and application in radiology. Insights into Imaging*, 9(4), pp.611–629.
[10] Hubel DH, Wiesel TN (1968) *Receptive fields and functional architecture of monkey striate cortex. J Physiol* , 195:215–243.
[11] Redmon, J. Divvala, S. Girshick, R. Farhadi, A. (2016) *You Only Look Once: Unified, Real-Time Object Detection*. Available at: https://pjreddie.com/media/files/papers/yolo.pdf (Accessed: 13 April 2019).
[12] Redmon, J. Divvala, S. Girshick, R. Farhadi, A. (2016) *You Only Look Once: Unified, Real-Time Object Detection*. Washington: University of Washington. Available at: https://pjreddie.com/media/files/papers/yolo.pdf (Accessed: 13 April 2019).
[13] Redmon, J. Farhadi, A. (2017) *YOLO9000: Better, Faster, Stronger*. Washington: University of Washington. Available at: https://pjreddie.com/media/files/papers/YOLO9000.pdf (Accessed: 14 April 2019).
[14] Redmon, J. Farhadi, A. (2018) *YOLOv3: An Incremental Improvement*. Washington: University of Washington. Available at: https://pjreddie.com/media/files/papers/YOLOv3.pdf (Accessed: 14 April 2019).
[15] Glenn-jocher, *Resume training from official yolov3 weights*. (2018) [Online]. Available: https://github.com/ultralytics/yolov3/issues/2 (Accessed: 15 May 2019).