Duck Hunt Challenge with Transfer Learning

Adam Kwan
adam876k@gmail.com
Noah Clarke
noahwclarke@gmail.com
Nolan Kurylo
nolankurylo@uvic.ca

University of Victoria
Victoria, Canada

Abstract

Computer Vision is a subset of AI and Machine Learning that is becoming more relevant in technology and expanding to new areas. One of these areas is object detection; the challenge of identifying multiple objects within an image and where they are located. This paper attempts to demonstrate how Computer Vision can be used for object detection by applying it to a variation of the 1984 shooter game, Duck Hunt [3]. The problem with most Computer Vision Tasks such as object detection is that they require massive amounts of data to properly learn a problem space. This paper illustrates how fine-tuning can solve this problem and still produce meaningful results through two well-known models, SSD MobileNet V2 and SSD ResNet50. In combination with a custom frame-displacement algorithm, these transfer learned models are compared in how well they perform in Duck Hunt.

Keywords - computer vision, fine tuning, transfer learning, machine learning

1 Introduction

The Duck-Hunt Computer Vision Challenge presents a problem that humans are adept at and creates an opportunity for the problem to be approached using computer vision methods. The basis for this challenge is to identify and shoot ducks with the game runs through a variety of levels [3]. As the levels progress, the ducks within the image become more obscured by background noise, blurring, changing of colours, and changing of directions. This level progression makes the challenge more difficult to approach from a computer vision perspective. This challenge has two main difficult tasks that need to be considered in a solution. The first challenge is "What computer vision methodologies can be used for this task?", there are many approaches for computer vision problems so finding one that is effective in the Duck-Hunt Challenge is essential. The second difficult task for this challenge is data collection because computer vision approaches to this problem are data-driven. Data collection is a difficult task because there is never enough data and collecting data has proven difficult for those working in the computer vision field.

For the first difficult task of choosing a computer vision methodology, the problem space states that the system will need to identify the location of ducks within an image. This means

that the methodology must be effective in object detection, specifically a methodology that can tell where the object is within the image. The difficulty comes from the fact that more than one object may be located in the image and will need to be identified. To best address this, transfer learning on two different pre-trained models was chosen to compare within this paper, SSD MobileNet V2 and SSD ResNet50. These two were chosen to contrast the effectiveness of a faster SSD MobileNet against a slower SSD ResNet50 architecture.

The second problem of data collection was difficult because there was no previous data set for this problem to build off of so the data collection needed to be started from scratch. Due to the nature of the problem of having to find the location of ducks within an image, the data needed to be annotated with labelled bounding boxes around each region of interest (ducks and dead ducks) before it was used for training. To collect data, images from the game environment were collected and then annotated using an online annotation tool called 'Roboflow'[1], which aided in the splitting of data and proper data format generation as well. Around 500 images were collected and annotated and while this was enough to create a dataset if there was more time allocated to this challenge more data could improve results. Since pre-trained models were chosen, the challenge with the data was more focused on accurate localization while not so much on augmentation variations.

2 Implemented Approach

2.1 Data Collection

To collect data needed for the CNN the Duck-Hunt game was run and 500 frames were saved as images from various levels between level 1 and level 819. Since the solution uses transfer learning of pre-trained models, the number of images collected was limited. Since the pre-trained models already have the ability to generalize a specific type of data, a small subset of new data (Duck Hunt frames) could be used to fine-tune the last layer of the model. After these images were collected, they were then annotated with tight bounding boxes around ducks using a third-party software, Roboflow [1], as seen in Figure 1. Similarly, dead ducks were labelled with their own bounding boxes. This implies data collection led to the decision of two distinct classes: ducks and dead ducks. There were nearly 500 images collected in total with a split of 80% for training, 10% for validation and 10% for testing.

Figure 1: Roboflow [1] Annotated Dataset Image with ducks (yellow) and dead ducks (pink)

2.2 SSD MobileNet V2

The first approach was done using the Single Shot MultiBox Detector (SSD) MobileNet V2 Architecture from TensorFlow's Model Detection API [2]. SSD MobileNet V2 is designed around being a fast detector over having a high mAP. The architecture of SSD MobileNet V2 is a single CNN with the layers shown in Figure 2 below. The benefit of using SSD MobileNet V2 is that it is known to be very fast, however, the downside is that there is a trade-off for accuracy, especially when considering mAP. The inference time is reported as 22ms with an mAP of 22.2 (the worst of all models in the TensorFlow Model Zoo for Object Detection)[2]. SSD MobileNet V2 uses a loss function weighted by the confidence loss from correct classification and a localization loss from the regression of the ground truth bounding box. The classification (1, 2) and localization (3, 4) loss functions are respectively [5]:

$$L_{conf(x,c)} = -\sum_{i\in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i\in Neg}^{N} log(\hat{c}_i^0) \tag{1}$$

where,

$$\hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)} \tag{2}$$

$$L_{loc(x,l,g)} = \sum_{i\in Pos}^{N} \sum_{m\in\{cx,cy,w,h\}} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m); \tag{3}$$

where,

$$\hat{g}_j^{cx} = \frac{g_j^{cx} - d_i^{cx}}{d_i^w} \qquad \hat{g}_j^{cy} = \frac{g_j^{cy} - d_i^{cy}}{d_i^h} \qquad \hat{g}_j^w = log\frac{g_j^w}{d_i^w} \qquad \hat{g}_j^h = log\frac{g_j^h}{d_i^h} \tag{4}$$

where the overall loss function is given as [5],

$$L(x,c,l,g) = \frac{1}{N}(L_{conf(x,c)} + \alpha L_{loc(x,l,g)}) \tag{5}$$

The classification loss has c: class confidence, p: category and α: tunable parameter via cross validation. The localization loss has l: predicted box, g: ground truth box, cx: center point for x, cy: center point for y, w: width of the box, and h: height of the box. [5]

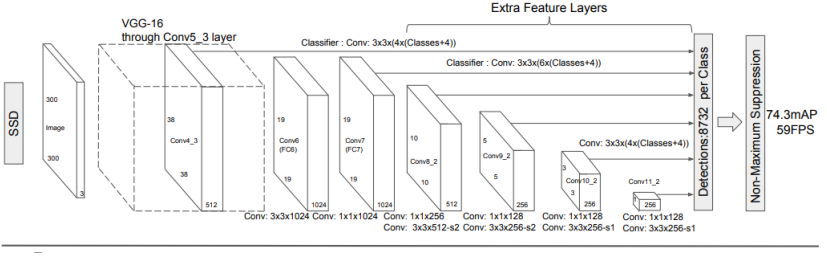


Figure 2: SSD MobileNet V2 Architecture [5]

2.3 SSD ResNet50

The second approach was done using the Single Shot MultiBox Detector (SSD) ResNet Architecture from TensorFlow's Model Detection API [2]. SSD ResNet V2 is designed around having a higher mAP which causes it to be a slower detector. The benefit of this architecture is that the authors report a higher mAP of 34.3 that will aid as a baseline before transfer learning [2]. There is a trade-off, however, where inference time is almost double that of SSD MobileNet V2 with its base taking 46ms to predict on an image [2]. The architecture of SSD ResNet50 is designed with an "identity shortcut mapping" feature, which allows for the preservation of upstream to downstream gradients during backpropagation as seen in Figure 3 [4]. The shortcut connection can also be represented mathematically [4]:

$$y = F(x, \{W_i\}) + W_s x \tag{6}$$

In the equation, we have x: the input vector, y: the output vector, $F(x, \{W_i\})$: the residual mapping that will eventually be learned during training, and the addition of $W_s x$ is the shortcut connection with W_s: a linear projection. [4]

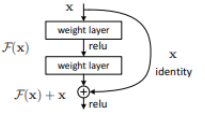


Figure 3: SSD ResNet50 Shortcut Connection Block [4]

2.4 Frame Displacement Algorithm

After testing the initial SSD MobileNet V2 detector, the results were poor. Particularly, the system would attempt to shoot behind where the duck was detected. The issue shooting behind the ducks was due to the inference time of the classifier, meaning that where the duck was in the detected frame was not the same as where the duck was when the shot got taken. This issue was addressed by adding in a prediction function to attempt to predict where the duck would be in the frame when the shot was actually taken after the inference time. The prediction function works by taking two consecutive inferences and analysing the change in position between inferences. It first matches ducks in the current and previous detection by doing a euclidean distance between the centre coordinate of the bounding boxes in the current and previous detection. The centre coordinate of a bounding box is calculated by taking the absolute positions of the top left and bottom right points of the bounding box and averaging them. Once the current ducks were paired with their counterpart in the previous detection, the distance they had moved was calculated by finding the displacement between each duck in the two frames corresponding to the difference in x and y coordinates. The system will then try to take a shot at the current duck's coordinates plus the calculated difference in the x and y directions. This prediction algorithm assumes that the duck is moving at a constant speed in one direction. This assumption can cause issues in later levels when ducks can change directions and move in non-linear paths.

3 Experiments & Evaluation

3.1 Experiments

Early testing relied on relatively poorly trained models via fine-tuning of SSD MobileNet V2. The initial goal was to have a working end to end test. While early trained versions of the model were able to play the game at early levels (i.e. levels 1 through 10), performance deteriorated quickly as the game progressed and became more difficult. Early tests also revealed an issue where the shooter would trail moving ducks slightly and never shoot them as the duck moved a sufficient difference during the inference time of the model to escape being shot. This issue was effectively remedied with the prediction algorithm outlined in section 2.4.

A second, somewhat expected issue, was the AI's tendency to shoot dead ducks. Due to the high degree of similarity between the sprites of dead and alive ducks dead ducks were being unnecessarily shot. Using the initial model, approximately 130 new samples were captured with dead ducks and a second class for dead ducks was created. Then the dataset was re-annotated on Roboflow [1] for dead ducks. Once the model was retrained dead ducks no longer posed a significant issue. To this point, the AI was able to play the game on early levels (approximately 1 through 30) with little difficulty shooting most of the ducks on each level.

Another issue with the early versions of the trained models was that they hadn't "fully" learned what a duck looked like yet. In Figure 4, the earliest version of the fine-tuned MobileNet predicts ducks with 30-40% confidence. This was due to a small custom dataset which was improved with more images and training for longer.

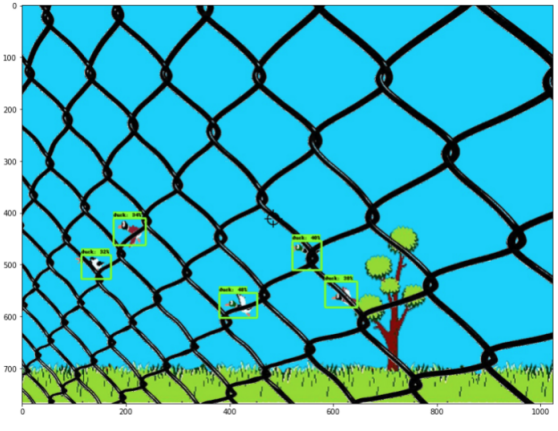


Figure 4: Early SSD MobileNet V2 Trained Model Prediction on Test Image

The next issue to address was accuracy at higher levels. The game's difficulty increases significantly after level 500 with little contrast between ducks and the background as well as overlays that partially obscure ducks. To combat this the training dataset was expanded significantly from approximately 250 to almost 500 images, many of which were taken from higher levels of the game. This new dataset was used to retrain the model to help improve its ability to shoot ducks at higher levels. Moreover, we were able to improve accuracy by normalizing how images in the dataset were tagged. Roboflow provides some advice on how

images should be tagged [6]. Additionally, a set of standards for how images should be tagged were established:

- Ducks should be bounded with a square box.
- The bounding box should be as small as possible while still including the entire duck.
- If the duck is partially occluded only tag the visible portion of the duck.
- If the duck is partially occluded and visible in two distinct regions (i.e. only the middle portion of the duck is occluded) tag the duck as if it is not occluded.

(a) (b) (c)

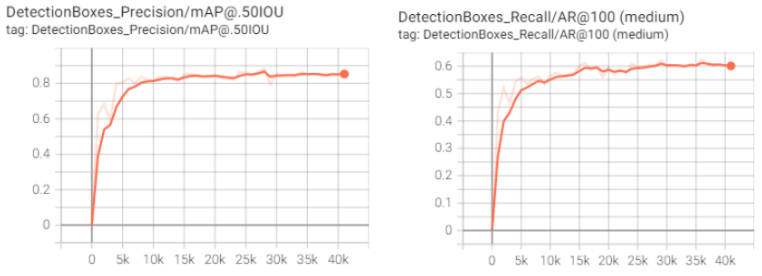
Figure 5: Examples from the training dataset (a) partially occluded duck (b) partially occluded duck visible in two distinct regions (c) overlapping ducks

Additionally, once new images were tagged by one group member their work was reviewed to ensure accuracy. The institution of these changes made the model, if not more accurate, at least more predictable.

3.2 SSD MobileNet V2 Evaluation

The evaluation of the SSD MobileNet V2 model was tricky because the mAP was reportedly around 22.2 [2]. This means the threshold that the shooting system considered a duck was quite low and often led to shots being taken that were not directed at actual ducks. Fine-tuning the threshold to accept any detection with precision above 0.25 helped minimize these errant shots but raising the threshold any higher would result in the shooting system not trying to shoot actual ducks. This is where the idea to train a new model with a reportedly more accurate starting architecture in the transfer learning came in. The new model chosen was SSD ResNet50 which had a reported mAP of 34.3, significantly higher than SSD MobileNet V2's mAP of 22.2 [2].

As seen in Figure 6a, MobileNet's training resulted in an acceptable convergence of validation 0.5 IOU mAP of 0.82. However, Figure 6b shows that the medium threshold AR was quite a bit lower at 0.6. Both figures show convergence after around the 15kth step in the training process, with little to no improvements be seen afterwards. This goes to show that although MobileNet was able to learn the custom dataset, there is still much room for improvement.

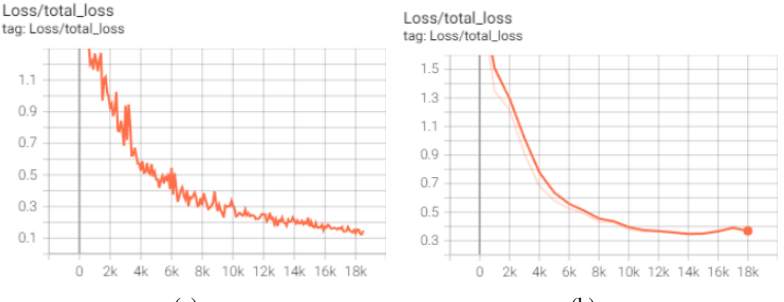


(a) (b)

Figure 6: SSD MobileNet V2 Validation Metrics: (a) mAP @ 0.5 IOU (b) AR @ 100

3.3 SSD ResNet50 Evaluation

ResNet was used in transfer learning on a custom dataset of nearly 500 images. These images normally had multiple ducks and dead ducks in the frame so plenty of variation was available for the model to fine-tune to. As seen in Figure 7a the training of ResNet gradually converged on the custom dataset. With an initial warm-up period for the learning rate, the training loss smooths out as it settles into a more optimal local minimum. The choppiness seen at the beginning of the training loss plot argues this point, where improvements were no longer being seen. The validation loss in Figure 7b follows a similar trend and eventually starts to level out around the 12kth step of the training process. The training for fine-tuning ResNet took nearly 6 hours on a Google Colab GPU so it was stopped at this point. However, it could have been continued to potentially optimize the model even more. This can be seen in how the validation loss begins the trend back down nearing the end of the plot. The training resulted in a medium-ranged thresholded mAP of 0.88 and mAR of 0.7 on the validation set. This was quite an improvement on the mAP of 0.82 and mAR of 0.6 that was achieved from MobileNet. The trade-off was a larger inference time for ResNet and thus, MobileNet was the preferred model. Inference speed needed to be minimized so as many shots as possible could be taken throughout a level of the game.



(a) (b)

Figure 7: SSD ResNet50 Losses: (a) Training (b) Validation

4 Conclusion

After evaluating the experiments for MobileNet and ResNet it was shown that ResNet was able to achieve a higher mAP of ~ 0.88 than MobileNet which achieved a mAP of ~ 0.82. The increase in mAP came at the cost of increasing the inference time from X to $\sim 1.5s$ from MobileNet to ResNet. The increase in inference time ended up being significant because the game logic dictated that there would be X number of ducks in the frame, and when a duck left the frame and new duck would be created in the frame to keep the same number of ducks in the frame. This meant that the longer the inference takes the fewer ducks the system would be able to shoot, and the fewer ducks would get shot meant that fewer new ducks would be created. This led to a lower maximum score that could be achieved and a worse performance when the inference time was high. Due to the inference time increase in ResNet, the final system was implemented with MobileNet for better performance.

There are a few future improvements that could be made to the system if more time and resources became available. The first improvement to consider would be to continue expanding the dataset. Currently, the dataset has approximately 500 samples, which is a quite small amount of samples for training most CNNs. When expanding the dataset there should be a focus on collecting dead duck images as there were significantly more ducks than dead ducks annotated in the dataset. Another future recommendation for this system would be to consider using a different model to base the transfer learning. The two models that were evaluated in this paper were SSD detectors so it could be worthwhile to base the transfer learning on a different model that is not using SSD. Comparing MobileNet and ResNet with a non-SSD model would provide insight into how effective the SSD approach to the Duck Hunt Challenge was.

References

[1] Roboflow. URL https://roboflow.com/.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[3] Lauren Boisvert. The truth behind duck hunt's secret ending, Jun 2021. URL https://www.looper.com/305828/the-truth-behind-duck-hunts-secret-ending.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In

Computer Vision – ECCV 2016, pages 21–37. Springer International Publishing, 2016. doi: 10.1007/978-3-319-46448-0_2. URL https://doi.org/10.1007%2F978-3-319-46448-0_2.

[6] Joseph Nelson. Seven tips for labeling images for computer vision, Oct 2021. URL https://blog.roboflow.com/tips-for-how-to-label-images/.