International Journal of Aviation Science and Engineering
e-ISSN: 2715-6958 p-ISSN: 2721-5342
AVIA

Original Article

Deep Learning Implementation on Aerial Flood Victim Detection System

Khairul Ummah[1], Muhammad Thariq Hidayat[1,*], and Ahmad Yudi Eka Risano[2]

[1] Faculty of Mechanical and Aerospace Engineering, Institut Teknologi Bandung. Labtek II, Jl. Ganesa 10, Bandung, 40132 Indonesia
[2] Department of Mechanical Engineering, Faculty of Engineering, Universitas Lampung. Jl. Prof. Soemantri Brojonegoro no. 1, Bandar Lampung 35145, Indonesia

* Correspondence: muhtariqhidayat@yahoo.com

Received: 14 June 2022; 22 November 2022; 30 December 2022

Abstract. Hydrometeorological hazard such as floods are considered as a regular natural disaster in Indonesia due to its frequent occurrence. To mitigate the risk, search and rescue operations need to be carried out immediately. The sheer magnitude of floods poses a major challenge for responders, and the emerging drone technology could help to alleviate the problem due to its deployment speed and coverage. Automation in drone technology has potential to improve its effectiveness. This paper explores the idea of human detection during floods using a computer vision approach. This approach utilizes a one stage detector model as detection speed is crucial in disaster management case. The dataset used for training consists of 200 labelled and negative images taken from drone point of view. This paper conducted 3 experiments to find out the difference in performance when the model was trained on flood and non-flood dataset, as well as the effect of image input size to the model's performance. The first experiment was trained on non-flood dataset. The second experiment was trained on flood dataset, and the third experiment is the modified version of the second model. The results show that the model trained on flood dataset performed worse than non-flood counterparts with the non-flood mAP reached 90.80% while flood mAP reached 39.15%. In addition, the experiments also conclude that increasing the input size of image during training, will increase the detection performance of the model at the cost of FPS.

Keywords: aerial flood, deep learning, hydrometeorological hazard, victim detection

1. Introduction

Indonesian archipelago, due to its location, has high rainfall events throughout the year. This natural condition resulted in Indonesia to be very vulnerable to flood disaster. Flood is considered as a regular natural disaster in Indonesia due to its frequency and causing socio-economic problems, especially when there are death cases reported [1]. An immediate life-saving response is required to rescue those who are trapped and evacuate survivors since the difference between life and death can be a matter of hours.

Search and rescue operations are often characterized by a similar set of constraints: time is critical and any delay can result in dramatic consequences [2]. Boats and helicopters are generally used during operation to find stranded victims. However, this approach is considered to be inefficient from time and cost aspects, due to the fact that the helicopter needs to fly in from an airport or base to the site. It is estimated that the cost of manned helicopter deployment ranges between \$10,000 to \$15,000 per hour [3]. In addition, the condition around the disaster area might reduce the capability of SAR team, since bad weather and flying at low altitude will increase the risk of the helicopter to crash. Furthermore, search and rescue boat are difficult to be used specially to cover large area of disaster zone like in urban area [4].

The new emerging technology that has experienced rapid development in the last decade is the drone technology. Although it was originally intended for military purposes, the application of drone technology has expanded greatly, including for life-saving operations. This technology could be the solution for a rapid response of SAR operation as drone provides a critical role due to its agility, high endurance, reduced cost, reduced risk, rapid deployment, and flexibility [5].

While the use of drones for this sector has increased recently, it is still operated manually by an operator on the ground. The availability of human resources and limitations of human operators could hinder the rescue effort as it is necessary to train pilots to operate the machine. One solution to the previous problem is to utilize an intelligent autonomous drone equipped with image recognition capabilities to detect and classify objects (victims) which will beneficially assist SAR operations. This is possible mainly due to the research and development in the field of computer and data science. As a result of that, computers are now able to detect certain objects by training them with relevant labeled data.

However, the lack of labeled data during flood will hinder the performance of the model. This research will explore the suitable method for victim detection as well as to determine the performance of the proposed method using different datasets and find out whether input size will influence the performance of the model. It is expected that using the developed algorithm and source code, the drone would be able to detect survivors during flood disaster

2. Materials and Methods

2.1. Data Collection

The first step to train this model is to collect appropriate image data. For this research, the author collected and created a dataset which supposedly contained images of people during flood from drone point of view. The author decided to take data from “Okutama-action” [6] which is a video dataset for aerial view concurrent human action detection, as well as retrieving other images that closely represent the situation from the internet. Total of 200 images were collected in this stage. To prevent the imbalanced of data, negative images (images that have no objects) were also included in the training set. The data obtained is divided into 2 groups, train set and test set with 70% and 30% proportion respectively.

2.2. Data Preprocessing

The raw data that will be fed into our model needs to be processed. The first step is to label objects in our input images by using a software called Labeling [7]. This step requires us to create bounding boxes or ground truths on the objects that we want to detect and define the corresponding classes for those objects in order to classify them in an image. The output of this step is a .txt file for each input image that contains image classes as well as the bounding box coordinates in YOLO format.

Once we are done with labeling, these images will be resized according to the desired resolution. In this research, the resolutions are ranging from 416 by 416 up to 448 by 448. Another method that is utilized in this step is data augmentation. This technique will increase the variability of images in order to improve the generalization of the model training. Both resizing and augmentation data were done by using software from Roboflow [8].

2.3. Training

In these experiments, there are few hyperparameters that need to be defined, as well as the Colab settings as shown below:

Table 1. Colab Related Settings

Parts	Value
GPU count	1
GPU	Tesla T4
Compute capability	750
CUDA-version	10010
CuDNN-version	7.6.5
CuDNN_HALF	1
OpenCV-version	3.2.0

Table 2. Hyperparameter Setup

Parameter	Value
Batch size	64
Subdivision	16
Width x Height	416
Channel	3
Learning rate	0.001
Max batches	6000
Steps	4800, 5400
Filters (Before YOLO layer)	18
Classes (in YOLO layer)	1

Transfer learning is a technique in which a pretrained model from one task is utilized to train a new model for another task. Computer vision field, especially in CNN, usually make a good use of this technique and in practice, it is rare to train the model from the scratch [9]. The pretrained model is commonly trained on a large dataset (e.g., ImageNet, MS COCO) and take days or weeks to complete. Transfer learning is popular since it helps to train models using fewer labeled data and tremendously reduce the training time compared to model that is trained from the scratch. This technique will replace and retrain the classifier using a new dataset or go even deeper by fine-tuning the weights in CNN layers. In this research, pre-trained weights for convolution layers from AlexeyAB's repository [10] are used to help the model to converge faster and more accurate during training phase.

2.4. Evaluation Metrics

In order to evaluate the performance of the model, the following evaluation metrics are discussed. Those metrics are confusion matrix, precision, recall, F1-score, intersection over union (IOU), mean average precision (mAP), and frame per second (FPS).

There are 4 categories in confusion matrix, those are True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). In object detection task, TP means that the model is able to detect and classify an object and its prediction is correct relative to ground truth. FP is either the model predicted an object even though there is no object there or the predicted box IOU to ground truth is lower than the threshold or simply that the model classifies the object wrong. FN means that the model is unable to detect an object even if there is an object in an image. The last one, TN, means that the model does not detect and predict any object, and the image has no object in it.

Precision measures how accurate is your prediction. Recall measure how good we detect all the object in an image. F1 score is the harmonic mean of the precision and recall. IOU measures the overlap area between the predicted bounding box and the ground truth bounding box over the union area of these two boxes. The mAP is generally defined as the area under the smoothed precision-recall curve. FPS refers to the number of images that can be detected per second.

3. Results and Discussion

3.1. Experiment Results of a Non-Flood Dataset

This experiment uses default hyperparameters in Tab.3, with input size of 416 by 416, and subdivision 16. Based on the results from Tab. 3 and Fig. 1, the mAP fluctuated in around 85% and reached a peak of 90.43%. What stands out in this figure is that the mAP plunged to the value of 23% and rose again. As for the loss graph, there has been a sharp drop in early iterations and gradually reaching a plateau. This could be a sign of overfitting.

Table 3. Experiments #1 Results (Non-Flood)

Iteration	1000	2000	3000	4000	Best	Last
TP	162	161	163	167	172	167
FP	38	37	23	24	19	19
FN	32	33	31	27	22	27
Precision	0.81	0.81	0.88	0.87	0.90	0.90
Recall	0.84	0.83	0.84	0.86	0.89	0.86
F1-Score	0.82	0.82	0.86	0.87	0.89	0.88
Avg IOU	56.55%	56.64%	62.06%	60.52%	62.80%	63.76%
mAP	83.99%	81.34%	84.18%	86.67%	90.43%	87.92%

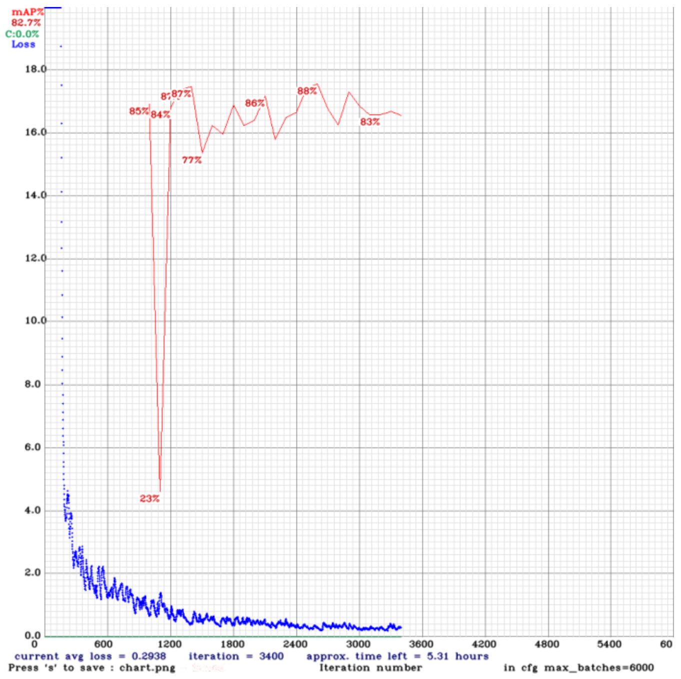


Figure 1. Experiment #1 Loss and mAP Chart

3.2 Experiment Results for Flood Dataset

This experiment uses default hyperparameters in Tab. 4, with input size of 416 by 416, and subdivision 16. Based on the results from Tab. 4 and Fig. 2, the mAP fluctuated in around 38% and reached a peak of 39.15%. Similar to the loss graph from previous experiment, there has been a sharp drop in early iterations and gradually reaching a plateau [9].

Table 4. Experiments #1 Results (Flood)

Iteration	**1000**	**2000**	**3000**	**4000**	**Best**	**Last**
TP	155	141	154	143	150	150
FP	177	115	111	116	105	105
FN	177	191	178	189	182	182
Precision	0.47	0.55	0.58	0.55	0.59	0.59
Recall	0.47	0.42	0.46	0.43	0.45	0.45
F1-Score	0.47	0.48	0.52	0.48	0.51	0.51
Avg IOU	31.57%	37.89%	39.14%	37.73%	40.14%	40.14%
mAP	33.90%	37.58%	37.34%	35.48%	39.15%	39.15%

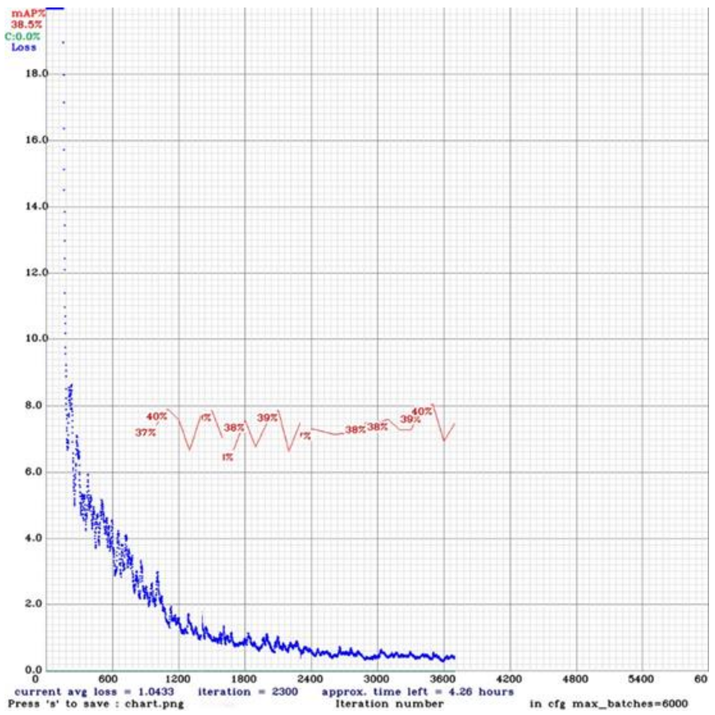


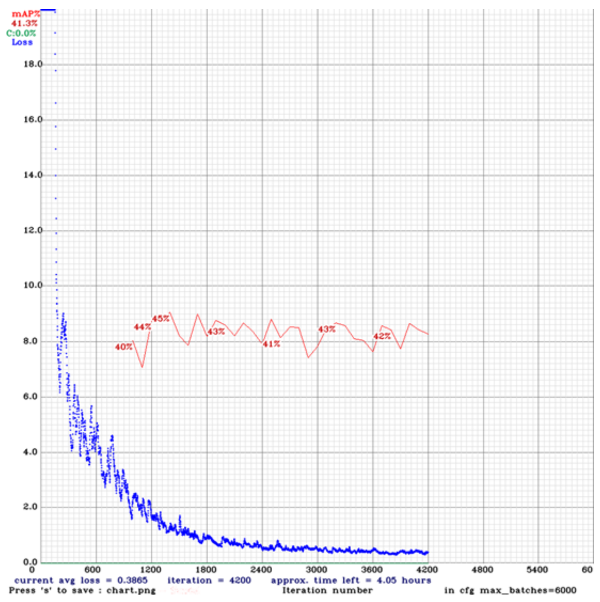
Figure 2. Experiment #2 Loss and mAP Chart

3.3 Modified Model

This experiment was trained with flood dataset, similar to Experiment #2: Flood dataset. However, the hyperparameters in Tab. 2 was modified, with input size of 448 by 448, and subdivision 32. Based on the results from Tab. 5 and Fig. 3, the mAP fluctuated in around 42% and reached a peak of 44.76%. Again, for the loss graph, there is a sharp drop in early iterations and gradually reaching a plateau. When loss no longer decreases and the performance is lower than the previous iteration, it could be a sign of overfitting.

Table 5. Experiments #3 Results (Modified Model)

Iteration	**1000**	**2000**	**3000**	**4000**	**Best**	**Last**
TP	178	159	150	154	165	160
FP	198	119	93	87	125	105
FN	154	173	182	178	167	172
Precision	0.47	0.57	0.62	0.64	0.57	0.60
Recall	0.54	0.48	0.45	0.46	0.50	0.48
F1-Score	0.50	0.52	0.52	0.54	0.53	0.54
Avg IOU	32.61%	38.86%	42.25%	44.35%	38.42%	41.91%
mAP	41.35%	43.31%	39.45%	43.32%	44.76%	41.07%



current avg loss = 0.3865 iteration = 4200 approx. time left = 4.05 hours
Press 's' to save : chart.png in cfg max_batches=6000

Figure 3. Experiment #2 Loss and mAP Chart

3.4 Performance Comparison of the Non-Flood and Flood Dataset

This subsection will compare and analyze experiments performance with respect to the training dataset. Experiment #1 and experiment #2 will be compared as shown in table 6 below.

Table 6. Non-Flood and Flood Dataset Performance Comparison

Experiment	**#1 [Non-Flood]**	**#2 [Flood]**
TP	172	150
FP	19	105
FN	22	182
Precision	0.90	0.59
Recall	0.89	0.45
F1-Score	0.89	0.51
Avg IOU	62.80%	40.14%
mAP	90.43%	39.15%
FPS	39.3	39.3

Based on the results in Tab.6, Experiment #1: Non-Flood dataset have higher performance in all aspects compared to Experiment #2: Flood dataset. It shows that the model has difficulty to detect objects from the flood dataset as can be seen with the low mAP of 39.15%. This is most likely due to the size of objects from flood dataset much smaller than the non-flood counterparts as well as the simplicity of the non-flood dataset. The flood dataset is more complex due to debris and other background objects that can be mistakenly classified as a human. Due to the same input size for both models, the FPS are the same.

3.5 Experiment #2 (Flood) and Modified Model

This subsection will compare and analyze experiments performance with respect to input size. Experiment #2 and experiment #3 will be compared as shown in table 7 below.

Table 7. Influence of Input Size

Experiment	#2 [416x416]	#2 [448x448]
TP	150	165
FP	105	125
FN	182	167
Precision	0.59	0.57
Recall	0.45	0.50
F1-Score	0.51	0.53
Avg IOU	40.14%	38.42%
mAP	39.15%	44.76%
FPS	39.3	38.8

Based on the results in Tab.7, Experiment #3: Modified model has a slightly better performance compared to Experiment #2: Flood dataset. It shows that increasing the input size will lead to a better performance as can be seen in the table above, where the mAP increases from 39.15% to 44.76%. On the other hand, the FPS of modified model has smaller value than the original model with the value of 38.8 and 39.3 respectively. This is logical since the larger the input size, the more detailed information available from the input image with the cost of the time. Hence, the model was able to detect smaller objects better. However, this result is still not desirable due to the fact that the Recall value is just 0.50, which means that the model was unable to detect some people in the image [10].

3.6 Visual Results Comparison

The models were tested and compared between each other using images in this subsection. The author decided to use IoU threshold of 0.3 for all the testing. At first, each model was visualized with the training image data, and their detection results of the training data. This step is to provide context only and does not have impact to the experiment, as presented in figure 4 and figure 5.

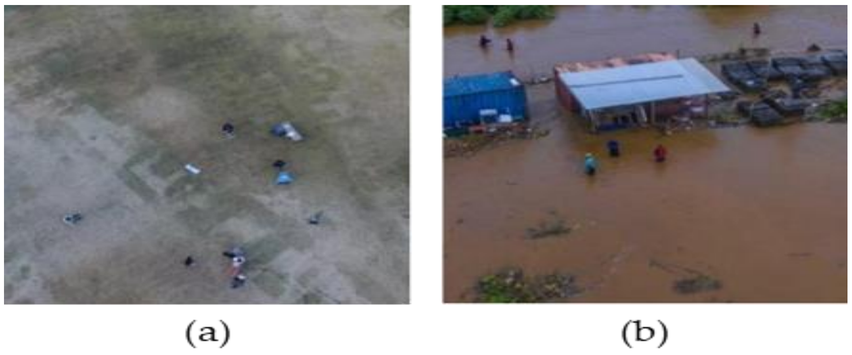


Figure 4. Dataset Input for (a) Experiment #1 and (b) Experiment #2 and #3

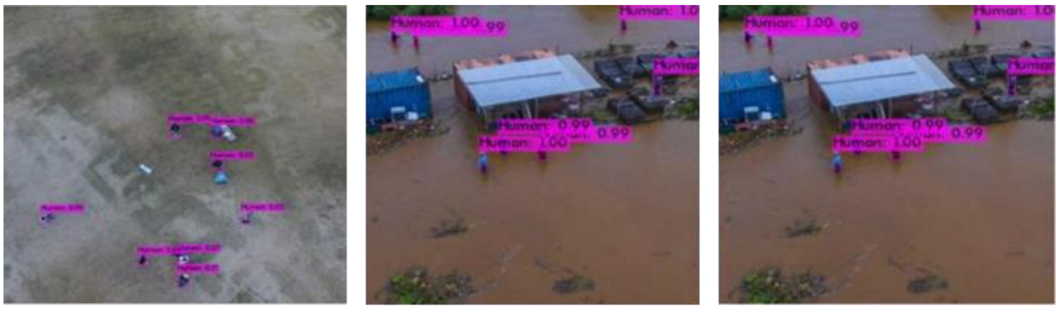


Figure 5. Tested with Training Data (a) Experiment #1, (b) Experiment #2, and (c) Experiment #3

These models were further tested with images that were not included in training dataset. The point is to see whether these models can generalize and detect objects on images that have never been seen by the models before. From the Fig.6 below, it can be seen that model #2 missed many objects (people) which indicates a higher degree of false negative. The other two models were able to detect most of objects, where model #1 performed slightly better than model #3 since it was able to detect more objects than the latter model. However, none of these models were able to detect all of the people in the image. For our case, minimum value of false negative is more desirable since it will reduce the probability of undetected victims.

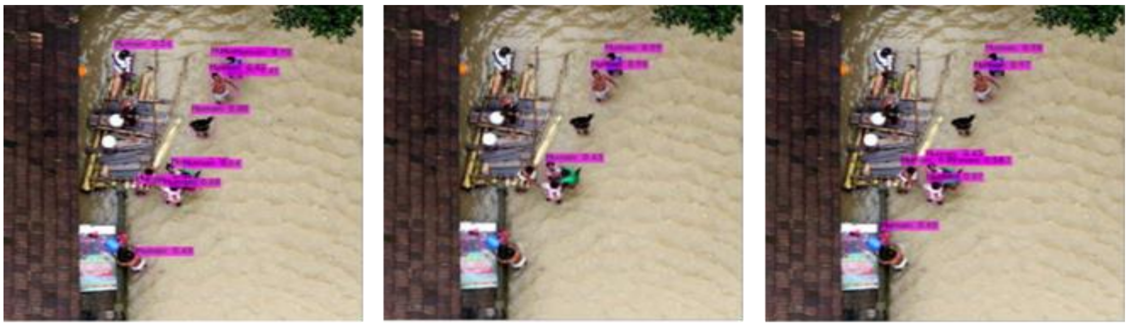


Figure 6. Tested with New Data for (a) Experiment #1, (b) Experiment #2, and (c) Experiment #3

Since model #3 performed better than model #2, the author decided to discard the model #2 from further comparison and test the remaining models with another never-seen image. To see the generalization of these models, the test image was the training image from different model. From the Fig.7 below, both models performed well and both missed one object from the image. The model #1, however, have more false positives where it classified debris on water as human.

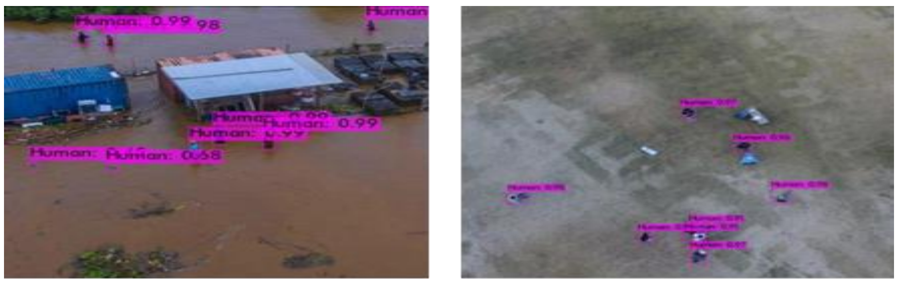


Figure 7. Tested with Swapped Training Data for (a) Experiment #1 and (b) Experiment #3

4. Conclusions

Based on the work done in this research, there are several conclusions that can be taken. These conclusions that the YOLOv4 is the most suitable method, and can be utilized to detect flood victims. The parameters that affect the performance are the training data and the input size. Increasing the input size will lead to a better performance where the mAP increases from 39.15% to 44.76%. By Increasing the input size, the model is able to extract more features due to more pixels. This is important because the objects taken from drone's view are most likely to be small which decreases the probability of model to detect victims. However, increasing the resolution decreases the FPS, where there is a chance that the model is unable to detect in real-time when the resolution is too large. Furthermore, the model trained on flood dataset performed worse than non-flood counterparts, in which the model achieved mAP of 39.15% for flood and 90.43% for non-flood.

References

[1] I. Narulita and W. Ningrum, IOP Conf. Series: Earth and Env. Sci. (2018)

[2] S. Waharte and N. Trigoni, Int. Conf. on Emerging Security Tech. (2010)

[3] R. Murphy, Drone Save Lives in Disasters When They're Allowed to Fly. https://www.space.com/30555-beginning-with-katrina-drones-save-lives-in-disasters.html (Accessed 11 May 2020)

[4] A. S. Hashim and M. S. M. Tamizi, Int. Jour. of Eng. & Tech. **7**, 9-12 (2018)

[5] C. D. Rodin, L. N. de Lima, F. A. d. A. Andrade, D. B. Haddad, T. A. Johansen, and R. Storvold, *Int. Joint Conf. on Neural Networks.* (2018)

[6] M. Barekatain, M. Marti, H.-F. Shih, S. Murrah, K. Nakayama, Y. Matsuo, and H. Predinger, arXiv. (2017)

[7] Tzutalin, Github Repository. (2015)

[8] Roboflow. https://roboflow.com

[9] Stanford University. https://cs231n.github.io/transfer-learning/

[10] A. Bochkovsky. https://github.com/AlexeyAB/darknet