CS230

Traffic Sign Localization and Classification

Anton Lok
Department of Computer Science
Stanford University
antonlok@stanford.edu

Kaitlin Peng
Department of Computer Science
Stanford University
kmpeng@stanford.edu

Victoria Wu
Department of Computer Science
Stanford University
vwu2010@stanford.edu

1 Introduction

1.1 All Background

Over the past decade, autonomous driving technology has captured the attention of automakers and the public alike. An important feature for safe autonomous driving is accurate traffic sign localization and classification. Obstacles to perfect localization and classification are numerous, ranging from inconsistent lighting to partial obstruction of the sign to glare in the camera. The goal of this project was to develop a system that can localize and classify traffic signs even under these suboptimal conditions. Attaining a model that can do so is important as studies [1] show self-driving cars could open 2 million employment opportunities for people with disabilities. Additionally, government data identified 94% [2] of automobile crashes to be the result of driver behavior and error. Through this technology, we make a quintessential mode of transportation more accessible and safer for all.

1.2 Literature Review & Pitfalls

The first research on traffic sign recognition (TSR) dates back to 1987 with Akatsuka and Imai [3] utilizing a color-identifying pre-processing step and dedicated hardware for template matching as its classification step. Through reviewing literature on TSR for the past 10 years, methods for TSR can be divded into three categories: (1) color-based methods (such as [3]), (2) shape-based methods, and (3) methods based on machine learning. For (1), approaches are based in color segmentation in various color spaces such as RGB, HSV, HSI, CIELab, CIELUB, CIECAM97s and others [4, 5, 6, 7]. The major pitfall for these methods was that they weren't robust with respect to different lighting conditions. (2) addressed this pitfall, with popular algorithms include Hough transform, geometric matching [8], and SIFT [9]. Dempster-Shafer's evidence theory for shape recognition though brings about a major pitfall: it limits the amount of exclusive classes [10]. (3) includes artificial neural networks (ANN) [11], support vector machine [4, 12], boosting [13] using integral image [14], genetic algorithms [15] and others. These primarily suffer from issue of viewpoint occlusion, physical occlusion, and the issues present in (1) [16]. A mixture between all three methods have been shown to yield the best results, compensating for the pitfalls of each [17]. We chose YOLOv7 as the base for our model because it outperforms most of the state of the art machine learning methods (transformer-based detector SWINL Cascade-Mask R-CNN 509% in speed and 2% in accuracy, and convolutional-based detector ConvNeXt-XL Cascade-Mask R-CNN by 551% in speed and 0.7% AP) and all prior YOLO models [18].

2 Dataset

Our project uses the Mapillary Traffic Sign Dataset [19], which contains over 40,000 hand-verified and hand-annotated images covering over 400 classes of traffic signs. We selected this dataset over other traffic sign databases for three reasons: (1) it is extremely large and comprehensive, covering traffic signs from all around the world; (2) it contains images taken from multiple camera types and perspectives; and (3) it contains images taken in widely varied weather and lighting conditions. A sample image along with an annotated version can be seen below (Figures 1 and 2).

All images were originally provided in their original format from the camera, ranging from approximately 2MP-8MP in size and taking various shapes. We downscaled the longer dimension of the photo to 640 pixels and letterboxed the resulting image to make it square, per YOLO requirements. While this smaller size did result in some labels being as little as 25% their original size, we found that it was necessary to allow the model to train with a reasonable memory footprint and time per epoch.

CS230: Deep Learning, Fall 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Figure 1: Sample Image

Figure 2: Sample Image with Annotations

The dataset as provided by Mapillary had a directory structure and label formats that were incompatible with YOLO, so we wrote a set of Python scripts to convert the labels and to organize the data into 60% training, 20% validation, and 20% test sets. We used these scripts to create two versions of the dataset: one that included all 40,000+ images, and a smaller one that randomly selected 20% of the images.

3 Baseline

Our baseline performance was achieved using a YOLOv5l (large) model for 100 epochs, using default weights and hyperparameters with no further tuning on architecture or hyperparameters. This model outperformed nano, small, medium, and extra large models. Given that YOLOv5 is the direct predecessor to the model we used for our main method, YOLOv7, and given that the model has been used for this task before [20], the baseline model gave us a benchmark against which to compare our experimental performance. The performance data is summarized below. Figure 7 in the Appendix gives the training data metric plots.

YOLOv5l Full Dataset Evaluation Metrics

Train			
Precision	Recall	mAP@0.5	mAP@0.5:0.95
0.567	0.119	0.0944	0.0687
Test			
Precision	Recall	mAP@0.5	mAP@0.5:0.95
0.58	0.148	0.115	0.087

4 Methods

Our research leading into this project indicated that the YOLOv5 models that have been previously trained for this task are minimally tuned and use the default hyperparameters [20]. Furthermore, we did not find any published projects using YOLOv7 for this task, with the exception of one simple model posted on YouTube. Thus, we applied a new model to an old problem.

4.1 YOLOv7 Design

4.1.1 Network Architecture

YOLOv7 is the most recent version of the YOLO family of single stage object detection models. This version improves on previous versions of YOLO by employing ELAN-

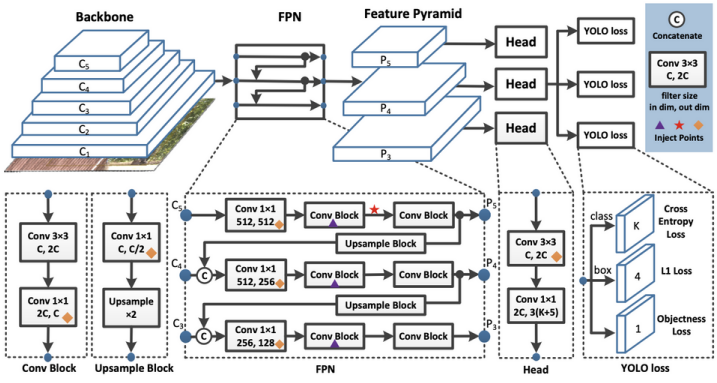

Figure 3: YOLOv7 Architecture [21]

based layer aggregation, intelligent model scaling, module-level re-parameterization, and supervision throughout the layers of the model [21].

4.1.2 Loss function

The original YOLOv7 loss function can be broken into three separate instances of mean-squared losses: (i) bounding box size and coordinate prediction, (ii) the confidence score, and (iii) the class prediction (see Figure 4).

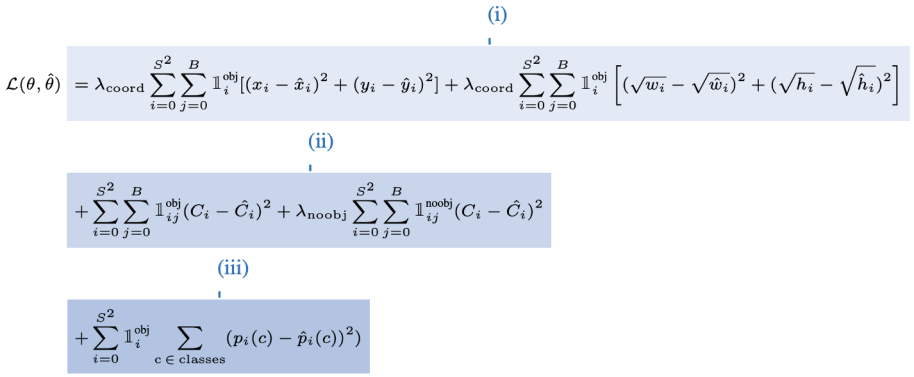

Figure 4: YOLOv7 Loss Function [22]

4.2 Experiments & Modifications

4.2.1 Vanilla YOLOv7

We first ran YOLOv7 with default hyperparameters for 100 epochs and a batch size of 16 to get a baseline YOLOv7 performance. Our selection of 100 epochs was based on a balance of runtime and performance.

In our error analysis step, we used wandb and clearML to extract incorrectly labeled training images. We found that majority of the erroneous images had small labels (see Appendix, Figure 8). We also found occlusion misclassifications, lighting causing false positives, and poor illumination misclassification.

To address these issues, we extracted bounding box label coordinates to find all labels <15 pixels in size and found 18.3% of all labels to fit that criteria. Since our starting mAP was so poor, and the vast majority of our errors were small labels, we decided it was worth to hone in on solving the small label problem since the other error patterns were marginal compared to it.

4.2.2 Focal Loss

We first tried changing the loss function. The intuition behind this was since our model was performing very poorly, we decided to experiment with a major architecture-level change. Because YOLO uses square error, it is not stable for training in the beginning, but reaches a good performance on the end [22]. Focal loss was said to be more stable [23]. Additionally, focal loss performs especially well in cases where there is class imbalance, which is very much the case here with 400 classes of traffic signs where signs like regulatory-maximum-speed-limit and other-sign dominating signs like information–telephone.

$$FL = -(1 - P_t)^{\gamma} \log(P_t)$$

Figure 5: Focal Loss function

What influenced our decision to try focal loss the most though was our results from the error analysis step, where we understood that our model was failing "hard" cases (small label, occlusion, etc.) and working well in "easy" cases where the sign was fully visible and easy to read. Since the hyperparameter γ in focal loss allows hard-to-classify examples to be penalized more heavily relative to easy-to-classify examples, we decided to try it. Through research, we found 1.5 was the best value for γ [24].

4.2.3 Attention - CBAM

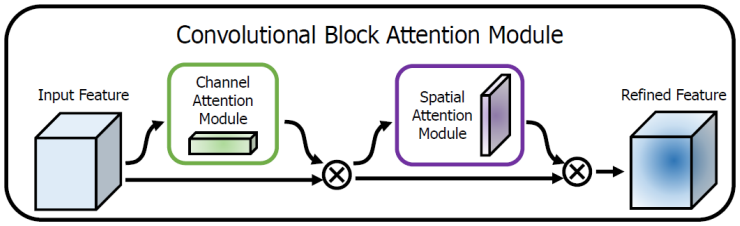

Figure 6: CBAM Structure [25]

Second, we tried adding the Convolutional Block Attention Module (CBAM) to YOLOv7 to improve our results since many of the traffic signs in our images were very small, a technique suggested by our TA and shown to improve performance in other YOLOv7 object detection research papers [26]. The idea of attention mechanisms in general are to make CNNs learn and focus on the important features of an image and reduce its learning for useless background information [27].

CBAM is a lightweight and general attention module that is composed of a channel attention module (CAM) and a spatial attention module (SAM) (see Figure 6) [25]. The CAM module helps the network focus on what is meaningful in an input image, while the SAM module helps it focus on where the meaningful object is. Adding this to our YOLOv7 architecture, in theory, should improve our model as it should be able to detect the small signs much better.

4.2.4 Non-Maximum Suppression

We also tried adding NMS to our CBAM model after the CBAM trial finished as we found that the model was paying a bit too much attention: almost all images now displayed a pattern of false positive "ghost" boxes with confidence scores between 0.001 and 0.002 (see Appendix, Figure 9). We remembered from lecture that NMS would discard all boxes whose confidence scores were under a specific threshold and then discard all remaining boxes with a greater IoU than a certain threshold. Since these boxes were all overlapping and numerous, we believed NMS would suppress them.

We did two rounds of testing: first, we implemented agnostic NMS and only allowed for single-labels. This would do the conf-box threshold and suppress all but the highest conf box for each sign detected. This actually performed so poorly that we killed training and iterated on another NMS approach. For our second round, we solely did agnostic and NMS with multi-label and normal IoU suppression (released papers say multi-label, though it creates false positives, still increases mAP score) [28]

4.2.5 Adam

We then switched YOLOv7's default stochastic gradient descent (SGD) to Adam. This was done right after we met with a TA to get some intuition as to why our model was doing so poorly, who said that poor performance was typical based on how large our dataset was and how many classes we had. While Adam is generally considered to perform worse than SGD for these tasks [29], we believed that it might improve our performance due to the large size of our dataset and difficulty in converging rapidly [29]. Additionally, based on what we learned in class, we believed that Adam would help combat the large oscillations we were observing in precision and recall during training due to its momentum component (see Appendix, Figure 11). Adam also updates the learning rate for each network weight individually unlike SGD, and though we found papers saying Adam performs worse on classification and recognition tasks [29], we found a slew of other sources that said Adam had been adapted as a benchmark for deep learning papers and recommended as a default optimization algorithm. [30, 31, 32, 33]

4.2.6 Hyperparameter tuning

We decided against hand-tuning hyperparameters due to the complexity and abundance of parameters in YOLOv7. We instead started hyperparameter tuning using YOLO's genetic algorithm [34] and initiated it using 100 10-epoch evolutions as a previous CS230 project suggested [35]. However, this sums to 1,000 total epochs run, which requires more than 17 GPU days, so we were unable to produce optimal parameters in time.

5 Results and Discussion

YOLOv7, as expected, outperformed our YOLOv5 baseline. Our only experiment that improved on the performance of the Vanilla model though was changing the optimizer to Adam. Below are our results for all the experiments.

	Train		Test	
Model	mAP@0.5	mAP@0.5:0.95	mAP@0.5	mAP@0.5:0.95
0. Baseline	0.0944	0.0687	0.115	0.087
1. Vanilla	0.1426	0.1031	0.132	0.095
2. Focal gamma	0.01975	0.01245	0.0397	0.0278
3. CBAM	0.07603	0.05406	0.0652	0.0462
4. CBAM + NMS	0.02891	0.01076	0.0587	0.0337
5. Adam	0.1391	0.1019	0.153	0.107

From our results, we see that Adam (5) performed the best. In training, the Vanilla model and Adam were comparable, but in testing, Adam performed better. In direct comparison to the baseline, Adam performed 33.04% better on mAP@0.5 and 22.99% better on mAP@0.5:0.95. We believe this is due to our original theory: Adam performs better on larger datasets with a larger number of classes [29]. Additionally, as discussed above, we believe the momentum component of Adam helped with the precision/recall oscillations. A pattern that spread across all our trials was that precision was often exchanged for recall; that is, to have a high precision meant a low recall, and as the model continued training, precision would decrease and recall would increase, since the metrics we were using were mAP@0.5 and mAP@0.5:0.95 so our model was trying to optimize for that. Also as discussed before, the fact that Adam has adaptive learning rates for different parameters meant that we probably had better tuning of our weights over the epochs.

Focal gamma (2) performed the worst. In direct comparison to the baseline, it performed 65.48% worse on mAP@0.5 and 68.04% worse on mAP@0.5:0.95 We think this may be due to the simplicity of the focal loss function: Referring back to Figure 5 in section 4.2.2, the function only has two parts: $-(1-P_t)^\gamma$ which penalizes when the true label is 0 and $log(P_t)$ which penalizes when the true label is 1 with a γ modulating factor. This γ can go all the way up to 5 and increases in such essentially "extend" or "relax" the criteria of a well-classified example [36]. Unfortunately, we found this focal loss doesn't capture enough information, with bounding box coordinates not being taken into account at all. Our next consideration to this is how we can incorporate focal loss into the YOLOv7 loss function itself to account for such lack of information. Then we can truly test if our intuition in penalizing hard-to-classify images more than easy-to-classify images actually works or not.

With CBAM (3), we also ended up getting worse results that performed worse than the baseline, which went against our intuition. It performed 43.30% worse than baseline on mAP@0.5 and 46.90% worse on mAP@0.5:0.95. Manual error analysis showed that we were outputting "ghost" images at a higher rate in this run than other runs, which may partly be the case. We don't think this contributed that much to the poor performance though because of the 0.001 and 0.002 confidence scores on these boxes, which squared with a true value of 0 in the YOLO loss function (i.e. $(0-0.001)^2$), we would get a marginal increase in loss. We think this may be the case for our run. Even though prior research and the TA's suggestion led us to believe an attention model would perform well for this task, we believe our dataset size and class count impaired performance.

For CBAM + NMS (4) the model significantly underperformed our expectations. It performed 48.95% worse than the baseline on mAP@0.5 and 61.26% worse on mAP@0.5:0.95. We ran with a confidence threshold of 0.35 and an IoU of 0.45, thresholds that we thought were reasonably small enough after looking through output images. Unfortunately, when looking at outputs of this run, we saw that some labels on the actual signs were getting removed and there were a number of photos where there was no label output on the images at all. So, our limiting thresholds could be one possible reason for the poor performance. Another reason may be the greedy method in which traditional NMS does clustering with a fixed distance threshold, which forces a trade-off between recall versus precision. [37] This though doesn't really seem like the case since both recall and precision were low, so we think it is more due to the former than the latter.

6 Future Work

Our biggest challenge during this project was achieving reasonable runtimes within the memory and performance constraints inherent in the EC2 instance, and within the constraints of our allocated AWS GPU credit. As mentioned before, we were unable to determine the point of weight convergence due to the sheer computational demands of the dataset (i.e. threshold for overfitting). We used 100-epoch trials, balancing a relatively reasonable runtime of 41 hours with a long enough run to allow some convergence and plateauing in the loss. In the future, we would be interested in expanding on this project through longer runtimes or parallel work on multiple GPUs. Bringing the runtime down or removing the time constraint would allow us to experiment more deeply with the YOLOv7 architecture.

7 Contributions

Anton and Vicky set up AWS, Anton and Kaitlin did dataset reformatting, Kaitlin wrote the dataset python scripts, and Vicky did the YOLOv5 testing. We all did research on YOLOv7, implemented the modifications to YOLOv7, and wrote up this report.

8 Code

All code associated with this project can be viewed at https://github.com/kmpeng/mapillary-yolo.

9 Appendix

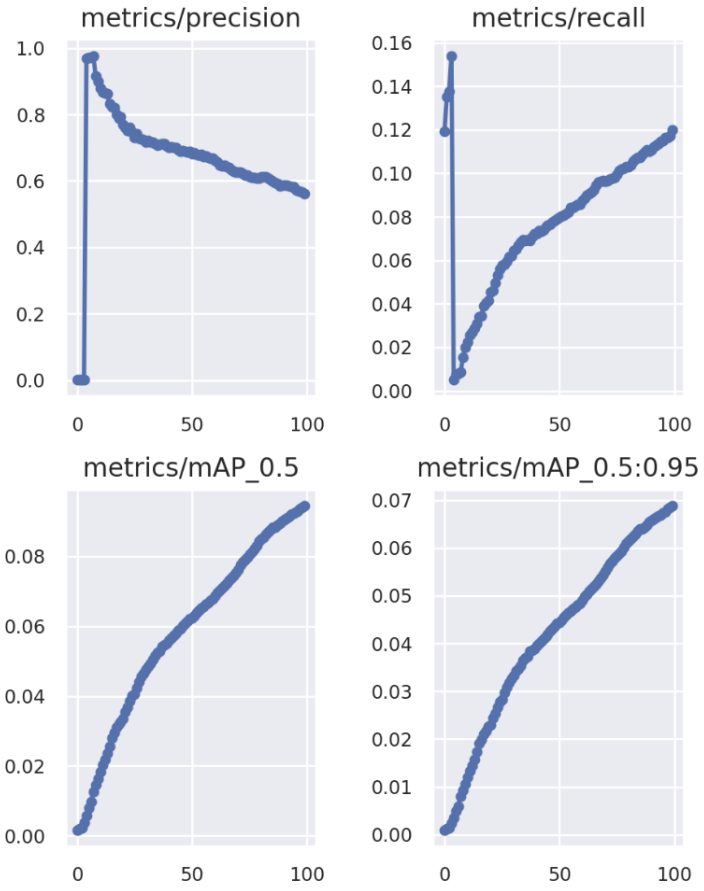

Figure 7: Training Graphs for Baseline

Figure 8: Error Matrix: classifying majority of small labels as "other-sign"

Figure 9: CBAM with "ghost" boxes

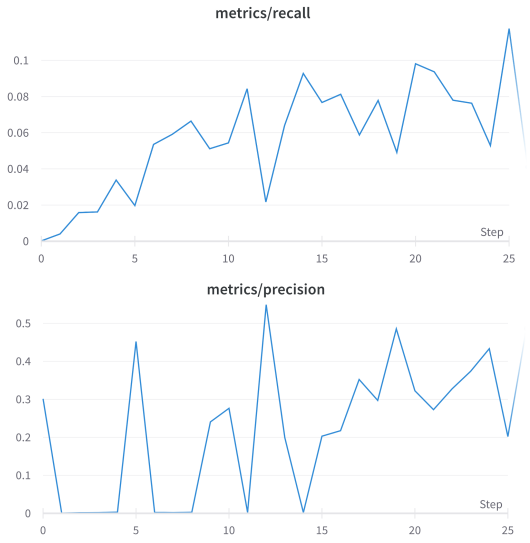

Figure 10: Sample output of noisy precision-recall exchanges

References

[1] Henry Claypool, Amitai Bin-Nun, and Jeffrey Gerlach. Self-driving cars: The impact on people with disabilities. Technical report, Ruderman Family Foundation, January 2017.

[2] Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Government Report DOT-HS-812-506, National Highway Traffic Safety Administration, March 2018.

[3] Hidehiko Akatsuka and Shinichiro Imai. Road signposts recognition system. In *SAE Technical Paper Series*. SAE International, February 1987.

[4] David Soendoro and Iping Supriana. Traffic sign recognition with color-based method, shape-arc estimation and svm. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pages 1–6, 2011.

[5] Tang Jin, Liang Xiong, Xie Bin, Chen Fangyan, and Liu Bo. A method for traffic signs detection, tracking and recognition. In *2010 5th International Conference on Computer Science Education*, pages 189–194, 2010.

[6] King Hann Lim, Kah Phooi Seng, and Li Minn Ang. Intra color-shape classification for traffic sign recognition. In *2010 International Computer Symposium (ICS2010)*, pages 642–647, 2010.

[7] Xiaohong W. Gao. A collection of benchmark images for traffic sign research. In *2011 17th International Conference on Digital Signal Processing (DSP)*, pages 1–6, 2011.

[8] S. Xu. Robust traffic sign shape recognition using geometric matching. *IET Intelligent Transport Systems*, 3(1):10, 2009.

[9] Merve Can Kus, Muhittin Gokmen, and Sima Etaner-Uyar. Traffic sign recognition using scale invariant feature transform and color classification. In *2008 23rd International Symposium on Computer and Information Sciences*, pages 1–6, 2008.

[10] B. Besserer, S. Estable, B. Ulmer, and D. Reichardt. Shape classification for traffic sign recognition. *IFAC Proceedings Volumes*, 26(1):487–492, 1993. 1st IFAC International Workshop on Intelligent Autonomous Vehicles, Hampshire, UK, 18-21 April.

[11] Song Zhenghe, Zhao Bo, Zhu Zhongxiang, Wang Meng, and Mao Enrong. Research on recognition methods for traffic signs. In *2008 Second International Conference on Future Generation Communication and Networking*. IEEE, December 2008.

[12] Min Shi, Haifeng Wu, and Hasan Fleyeh. Support vector machines for traffic signs recognition. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 3820–3827, 2008.

[13] Claus Bahlmann, Ying Zhu, Visvanathan Ramesh, M. Pellkofer, and Thorsten Köhler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. pages 255 – 260, 07 2005.

[14] Peter Tarabek. Fast license plate detection based on edge density and integral edge image. In *2012 IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 37–40, 2012.

[15] Lunbo Li and Guangfu Ma. Recognition of degraded traffic sign symbols using pnn and combined blur and affine invariants. In *2008 Fourth International Conference on Natural Computation*, volume 3, pages 515–520, 2008.

[16] Yiqiang Wu, Zhiyong Li, Ying Chen, Kenkoh Nai, and Jin Yuan. Real-time traffic sign detection and classification towards real traffic scene. *Multimedia Tools and Applications*, 79, 07 2020.

[17] Štefan Toth. Difficulties of traffic sign recognition. 02 2012.

[18] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.

[19] Mapillary Research. Mapillary traffic sign dataset. `https://www.mapillary.com/dataset/trafficsign`, 2020.

[20] Junghwan Yim. Traffic-sign-yolov5-mapillary. `https://github.com/JacobYim/Traffic-Sign-Yolov5-Mapillary`, January 2022.

[21] Jacob Solawetz. What is yolov7? a complete guide., July 2022.

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015.

[23] Saptashwa Bhattacharyya. A loss function suitable for class imbalanced data: "focal loss": Deep learning with class imbalanced data, February 2021.

[24] adldotori. Efficientdet inference (better than yolov5), 2021.

[25] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[26] Kailin Jiang, Tianyu Xie, Rui Yan, Xi Wen, Danyang Li, Hongbo Jiang, Ning Jiang, Ling Feng, Xuliang Duan, and Jianjun Wang. An attention mechanism-improved yolov7 object detection algorithm for hemp duck count estimation. *Agriculture*, 12(10):1659, Oct 2022.

[27] Gianni Brauwers and Flavius Frasincar. A general survey on attention mechanisms in deep learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[28] Brad Dwyer. Launch: End to end multi-label classification, April 2022.

[29] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning, 2017.

[30] Ayush Gupta. A comprehensive guide on deep learning optimizers, October 2021.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[32] Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of adam and adagrad, 2020.

[33] Wei Yuan and Kai-Xin Gao. Eadam optimizer: How impact adam, 2020.

[34] Ultralytics. Hyperparameter evolution, 2022.

[35] Margaret Daly and Carlota Parés-Morlans and Mohammed Salman. Auto-detection of southern sea otters (enhydra lutris nereis) from aerial imaging on the monterey peninsula. `http://cs230.stanford.edu/projects_fall_2021/reports/103120671.pdf`, 2021.

[36] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2017.

[37] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. A convnet for non-maximum suppression, 2015.

[38] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[39] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd, 2017.

[40] iscyy. yoloair. `https://github.com/iscyy/yoloair`, August 2022.

[41] Junetae Kim, Sangwon Lee, Eugene Hwang, Kwang Sun Ryu, Hanseok Jeong, Jae Wook Lee, Yul Hwangbo, Kui Son Choi, and Hyo Soung Cha. Limitations of deep learning attention mechanisms in clinical research: Empirical case study based on the korean diabetic disease setting. *Journal of Medical Internet Research*, 22(12):e18418, December 2020.