DeepBird: A deep learning pipeline for wildlife camera data analysis

Sarah Harvey
Department of Applied Physics
Stanford University
harveys@stanford.edu

Abstract

Wildlife camera observations of rarely-seen and remote wildlife are vital to our modern understanding of ecosystems. However, the deployment of these cameras at a large scale generates an amount of data that is intractable for humans to parse manually. Here I implement a deep learning pipeline for automatic wildlife camera video analysis for the case of North America bird species. I pursue a two stage approach: first performing object detection on video frames and then performing fine-grained image classification to identify bird species. I find that this two-stage method increases the species identification accuracy and avoids confusion from multiple birds in a single frame of video. The pipeline returns an analysis of the video informing the researcher of unique bird species visiting the location and how long they lingered.

1 Introduction

Over the past 50 years, North American birds have declined in number by 2.9 billion, or by 29% (1). This shocking statistic is the result of painstakingly collected citizen science bird watching data over many years, and is indicative of widespread habitat destruction and ecosystem degradation. Taking this data was possible because some bird species are relatively easy to catalog by bird enthusiasts, however, bird species that are harder to find and identify by humans are less likely to be represented accurately. One might imagine that a network of wildlife cameras (potentially in remote locations) continuously taking video data, 24 hours per day, 356 days per year, may capture a less biased and higher resolution view of ecosystem change on our planet.

Wildlife cameras, often called 'camera traps', are a modern tool to help address the long-standing ecological challenge of estimating animal population sizes, which can be a valuable indicator of overall ecosystem health (2; 3). These cameras provide a transformative mechanism to study rare species and remote environments. However, at a large scale the video data generated is extremely time-consuming for humans to analyze, a problem that is sometimes solved using citizen science methods (4).

The goal of this project is to investigate whether deep convolutional neural networks (CNNs) could be applied to this problem to perform automatic video segmentation and fine-grained image classification. This implementation focuses on the detection and identification of bird species, as several large and high-quality bird image and video datasets are freely available. The pipeline takes an RGB wildlife camera video as input and outputs a list of the unique bird species identified in the video and the time intervals that they spent visiting the camera location. This requires moving beyond the well-studied task of fine-grained image classification on large datasets with one high-quality bird per image, to localizing and segmenting multiple birds per frame and tracking individuals through a video clip. To accomplish this, I implemented a two-stage system, first using the object detection algorithm You-Only-Look-Once (YOLO) (5) to distinguish birds from non-bird objects, and then identifying the species of the detected birds using a ResNet-50 network.

2 Related work

Fine-grained image classification is a well-studied field of computer vision that is concerned with the identification of objects that are visually very similar (6; 7). High-level performance by humans on these classification tasks may only be achievable by trained experts. In the modern days of deep learning, high performing algorithms for these tasks usually involve convolutional neural networks (CNNs) of various different varieties. These often employ non-trivial algorithmic augmentations to standard architectures, such as pose estimation (8), parts models (9), and resolution correction strategies (10), in order to perform at or near state-of-the-art on fine-grained image classification tasks. These approaches do perform highly on datasets that are relevant to this project, but at the cost of much added complexity in implementation. Transfer learning approaches have also been successful, with 2018 study by Cui et. al. (11) demonstrating near state-of-the-art performance on prototypical datasets using more standard models

CS230: Deep Learning, Autumn 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

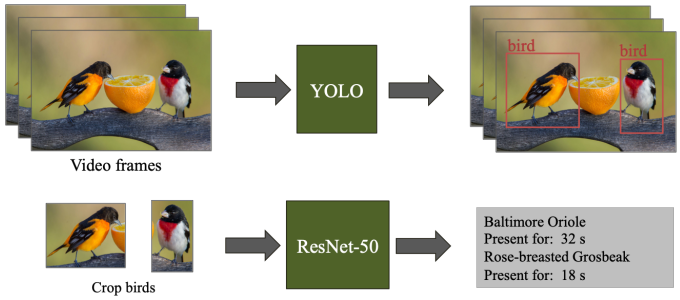

Figure 1: Cartoon schematic of the deep learning pipeline two-stage approach. The system takes as input a video and outputs a list of bird species present in the video and how long they lingered in front of the camera.

like ResNets and Inception networks pre-trained on large, domain-similar datasets. Two important and well-studied datasets for the task of bird identification are the Caltech-UCSD Birds 200-2011 (12) and the larger NAbirds dataset (13). The state-of-the-art for classification accuracy on these datasets is currently 90.4% and 89.2% respectively, which was achieved by the aforementioned parts model approach (9) and resolution-correction approach (10). In contrast, expert human performance on the CUB dataset is estimated to be between 93% and 95.6% (14). However, these implementations trained on large bird datasets are not typically well-suited to the real-world application of processing wildlife camera data, as the data can contain many species in each video frame. Previous work in the object detection (localization and classification) field is dominated by the YOLO (You Only Look Once) (5), which is designed to perform both object localization and classification in one pass through a network. YOLO is notable for its speed and accuracy, and has seen several improvements since it was first introduced in 2015 (15**?**). However, most implementations of YOLO are not designed for fine-grained image classification, as is needed for distinguishing bird species.

3 Dataset and Features

Three major datasets were used in the creation of this project: the Caltech-UCSD Birds 200-2011 dataset (CUB) (12) the NAbirds dataset (13), and the VB100 Video Bird Dataset (16). These CUB and NAbirds are both large databases of avian images, labeled by species. The smaller CUB dataset was mainly used for development, with a total of 11,788 images and 200 categories it was the fastest to train. The NAbirds dataset is a more recent, higher-quality image dataset consisting of only North American birds containing over 48,000 species-labeled images over 555 fine-grained categories often with over 100 images per species, as well as hierarchical categories and bounding boxes. This dataset was used to train the final ResNet-50 used for species classification, as it resulted in a more practically useful network that can identify species common in the California environment. The VB100 dataset consists of 1416 video clips of 100 bird species taken by expert bird watchers, and was used as a source of 'camera trap' videos to test my pipeline.

4 Methods

The bird identification stage depicted in figure 1 (bottom row), which is inspired by the state-of-the-art fine-grained classification results reported in Cui et. al. (2018) (17), is based on a ResNet-50 model with weights initialized on the ImageNet dataset in Tensorflow / Keras (18; 19). This core enables transfer learning with variable trainable / frozen layers and network architecture configuration using additional layers in the Keras framework. From the original network, the output layer was removed, and the last global max pooling layer was directed into a set of fully-connected layers of a user-specified width, followed by a final fully-connected output layer with softmax activation. The model was then trained using the categorical crossentropy loss function on either the CUB and NAbirds datasets using a Nvidia GeForce GTX 1080 Ti GPU.

The bird detection aspect of this project depicted in figure 1 (top row) adapts an existing implementation of the YOLOv3 algorithm by Huynh Ngoc Anh, which can be found in the author's GitHub repository (20), with help from a blog post by Jason Brownlee (21). This model is pre-trained on the COCO dataset of 220,000 images (22), which contains a bird object category. Browsing through the COCO images used to train this network reveals that COCO contains a wide variety of bird shapes and sizes, and even representations of birds such as paintings, so it seemed plausible that this network could be adapted to localize the large diversity of birds in the VB100 dataset. This YOLO implementation uses a DarkNet architecture as the base network.

5 Experiments/Results/Discussion

Initial experiments were centered around tuning the ResNet-50 classification network to achieve the highest possible performance. Previous reports (23) suggest that it should be possible to achieve a classification accuracy around 80-84% using ResNet-50 on the CUB dataset.

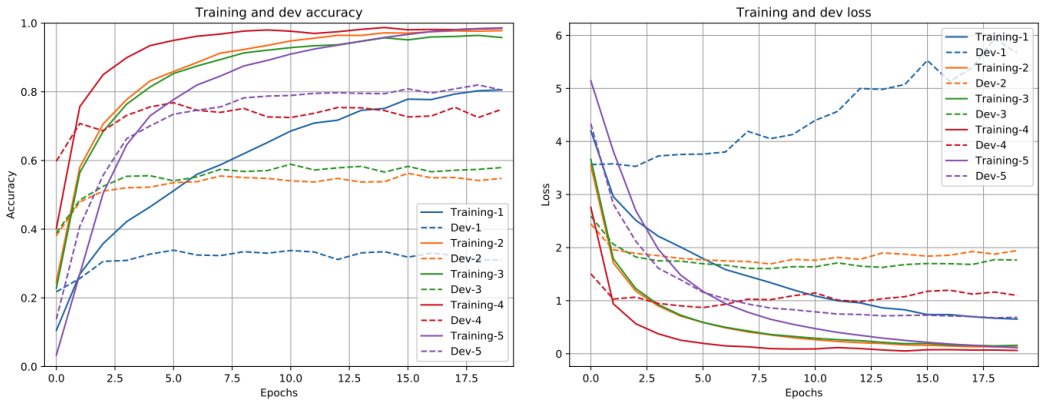

Figure 2: Performance of ResNet-50 transfer learning tests on Caltech-UCSD Birds-200-2011 dataset.

The hyperparameter tuning process was completed using the CUB dataset with input image sizes of (224, 224). Aside from the learning rate, optimization algorithm, regularization, and epoch number, the additional hyperparameters to tune in the transfer learning case include the number of add-on layers, number of neurons in these layers, and the number of layers in the base model to retrain. To evaluate the performance of the model, the 11,788 images in the CUB dataset were divided into 90% training and 10% development sets. The development set was held out during training and used to evaluate the performance of the model on unseen data after every training epoch. For all of the experiments displayed below, the Adam optimization algorithm was used (with default parameters) to minimize the categorical cross-entropy loss function.

I first implemented a model with two additional fully connected layers (using 512 then 256 neurons) with a softmax activation function output layer and frozen base model weights (figure 1, training-1 and dev-1), using a mini-batch size of 64. The left panel of figure 1 (training-1 and dev-1), shows that this model performs at only approximately 32% accuracy on the development set, and this level appears to saturate after about 5 epochs. The model is clearly underperforming: in particular, it has high bias (as it is not even able to overfit the training set), and the test/dev accuracy split of over 40% also indicates that there is a variance problem To mitigate the variance problem, I tried simplifying the model. Eliminating the two fully-connected layers of the network between the global average pooling and the output softmax layer resulted in a model still able to overfit the data, with a training accuracy of 97% and improves the dev set accuracy to 55% (figure 1, training-2 and dev-2). I also implemented L_2 regularization of the weight matrices in the model with two fully-connected layers before the output layer, which slowed training but did not seem to improve dev set performance even over a range of regularization parameter values.

Moving forward with the simplified model of training only the output layer with softmax activation, I experimented with increasing the image input size. Unsurprisingly, increasing the size of the inputs slowed down training and

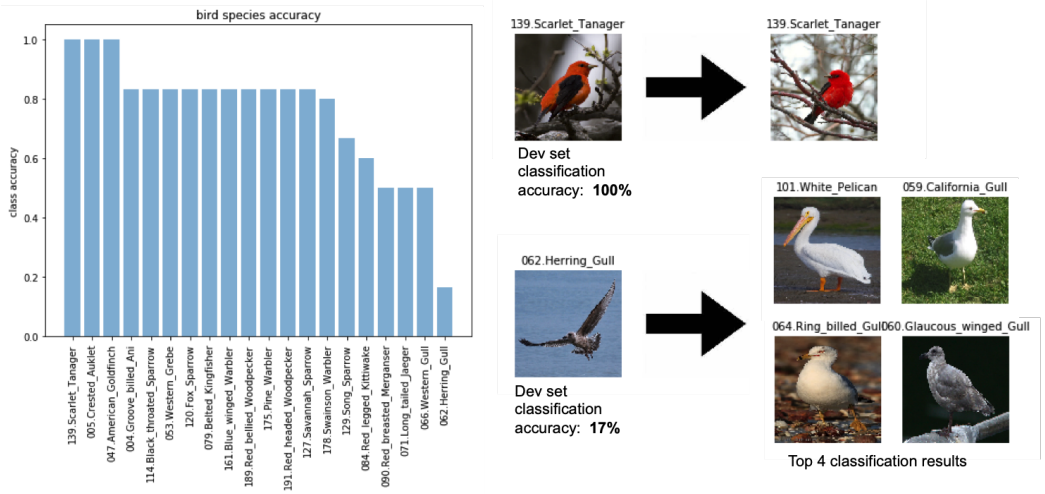

Figure 3: Classification accuracy on the dev set broken down in to species categories. Some species are correctly classified every time they appear in the dev set, and some are more difficult.

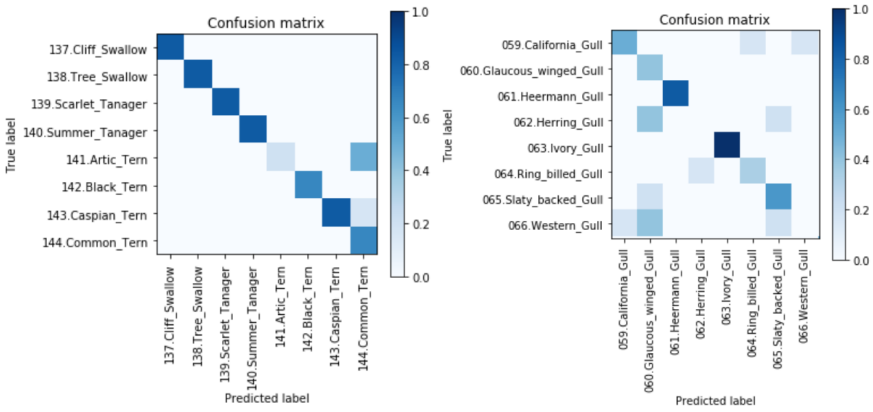

Figure 4: Subsets of the confusion matrix for the bird classification task on the CUB dataset. The left panel shows a region containing mostly easy-to-classify birds, including the scarlet tananger. This is contrasted with the right panel, which shows that the gulls are often misclassified as other erroneous gulls.

required a decrease in the batch size, but also increased dev set accuracy by about 2% (figure 1, training-3 and dev-3), suggesting that the fine detail contained in higher resolution images is helpful for this classification task. Lastly, I also experimented with allowing the layers of the base model to be trainable, which is essentially using the ImageNet classification weights as initialization for this model re-trained exclusively on bird data. This increased dev set performance to over 70% (figure 1, training-4 and dev-4). After tuning the learning rate, I find that this model achieves approximately 80% dev set accuracy after 13 epochs (figure 1, training-5 and dev-5), which is starting to approach what one should expect from the literature.

To gain insight into the performance of the model, we can examine the dev set accuracy subdivided into species categories, as shown in the bar plot in figure 3. This figure, abbreviated to show a random sampling of the 200 species categories, shows that some bird categories are more error-prone than others. Gulls appear to be a particularly difficult species to classify, however, when a gull is incorrectly classified, it is often classified as another species of gull. This seems to suggest that the gulls are closely clustered in some latent space and that the network is learning intuitive semantic features of birds to distinguish species. We can also see this represented in the confusion matrix subsets in figure 4, from which it is apparent that gulls are largely misclassified as other gulls. This may also imply that the images are simply not detailed enough (once they are resized for input to the network) for the network to learn such a fine distinction.

The ResNet-50 implementation was also trained on the NAbirds dataset with minimal additional hyperparameter tuning, as it became very time-consuming to fit this model to the larger dataset. The final performance of the model trained on the two datasets after hyperparameter tuning is summarized in figure 5.

Dataset	Precision	Recall	F1-score	Accuracy	Top-5 Accuracy
CUB-200-2011	0.85	0.83	0.83	0.83	0.96
NAbirds	0.83	0.80	0.80	0.80	0.95

Figure 5: Summary of the performance of the fine-grained classification network on two popular bird species datasets (subsets held out from training).

The YOLO stage of the video analysis pipeline was implemented and applied to the VB100 dataset. This algorithm has varying success detecting birds frame-by-frame, and its ability to detect birds is significantly impacted by bird movement (blur artifacts) or irregular poses. Depending on the species and video quality, it tends to detect the correct number of birds correctly for between 30% and 99% of the frames. Much of this variation can be corrected by smoothing out the output. One can safely assume that birds do not enter and leave between one or two frames (given a reasonably fast frame rate), so if a long string of bird detections occur separated by one or two non-detections in the middle, we can treat this as a spurious error. I can then report the length of the block of detections as one ‘visit’ of this bird. The detected birds are then cropped from the frames and sent to the ResNet-50 for image classification, using an input image size of (310, 310) pixels. This gives as an output the number of birds detected in each frame of the video (as a time series) as well as the species of each of those birds at that timestep. This data can then be used for further analysis.

For the majority of VB100 videos evaluated using this pipeline, cropping the birds using the YOLO detections before classification improves the classification accuracy. However, there are problematic instances of the YOLO bounding box cutting off portions of the bird, which could be a source of misclassifications. Out of the 31 videos passed through

the pipeline so far, the majority vote species classification of all of the frames achieved 84% accuracy. Additionally, using YOLO detection to crop the video frames before submitting images for classification generally suppressed the number of unique identifications among all the frames of the video (figure 6) compared with classifying un-cropped (raw) frames of video directly with the ResNet-50 network.

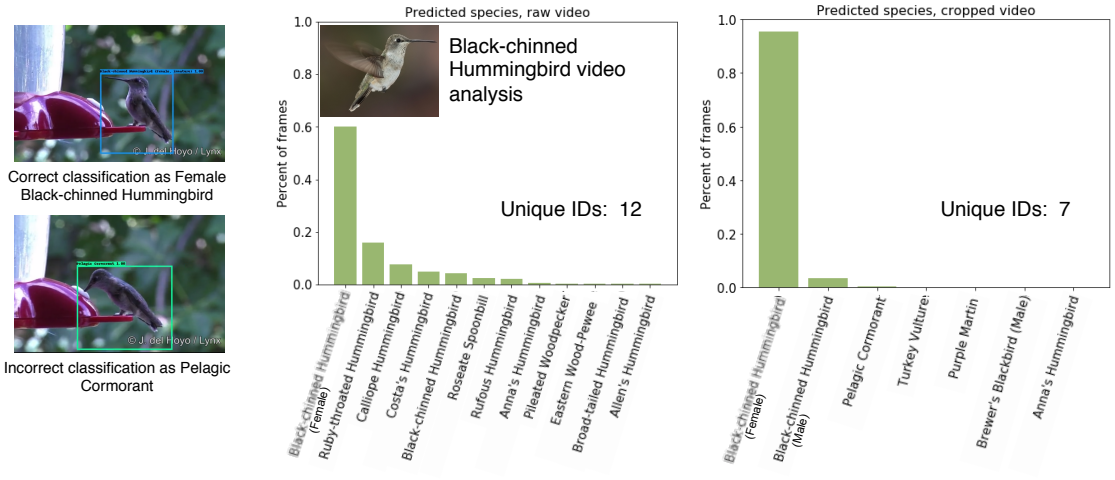

Figure 6: Comparison of the distribution and number of unique identifications made by the ResNet-50 network between inputs of raw video frames and video frames cropped by the detection network. After cropping, the frames are mostly misclassified as other birds of similar taxonomy.

6 Conclusion/Future Work

Overall, this project represents an application of deep learning to a challenging and dynamic problem in wildlife biology. I have implemented two networks, one to perform the YOLO algorithm and detect bird 'objects' from non-bird objects in single frames of video, and one to perform fine-grained image classification to extract the species of each bird detected. The two-stage approach appears to improve classification accuracy on average, which is likely due to removing potential artifacts from the background as well as increasing the detail of that bird seen by the network (if the images fed to the classification network are initially smaller, when they are resized to (310, 310) more pixels will tile input bird). However, both the YOLO detection algorithm and the ResNet fine-grained classification network suffer in certain scenarios, such as when the bird is of a non-standard shape, is in a camouflaged environment or in an irregular pose. It seems likely that this pipeline could be improved by re-training the YOLO detection network using exclusively examples of birds, to avoid mis-detection of uncommon birds not in the COCO dataset. Additionally, there is an obvious disconnect between the quality of data that the classification ResNet-50 is trained on and the more 'candid' video frames which it is asked to identify. The NAbirds dataset is full of high-resolution, professional images of centered birds looking into the camera. However, in real video data, the bird is behaving in its natural environment and many of the frames feature motion artifacts or the bird not facing the camera (figure 7). The fine-grained bird classification task so often studied in deep learning could perhaps be made more practically useful for applications by training on images that aren't all as well framed as the NAbirds dataset, so the network learns to recognize birds behaving naturally. This also suggests that perhaps the spatiotemporal information introduced by the time-series nature of the data could be better exploited to improve accuracy on challenging videos. Future work on this project could involve a custom implementation of the YOLO algorithm trained on bird species datasets to perform the localization and classification in one pass through the network. Furthermore, this project could be extended to process videos with more animals besides birds present in order to more accurately capture what wildlife cameras encounter in the field.

Figure 7: Comparison of the difference in image quality between the classification training dataset and some of the test videos supplied to the pipeline. NAbirds image is centered with subdued background; VB100 frame is a realistic portrait of a camouflaged bird with motion artifacts.

References

[1] K. V. Rosenberg, A. M. Dokter, P. J. Blancher, J. R. Sauer, A. C. Smith, P. A. Smith, J. C. Stanton, A. Panjabi, L. Helft, M. Parr, and P. P. Marra, "Decline of the North American avifauna." *Science (New York, N.Y.)*, vol. 366, no. 6461, pp. 120–124, oct 2019. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/31604313

[2] "The Art and Science of Camera Trapping · National Parks Conservation Association." [Online]. Available: https://www.npca.org/resources/3236-the-art-and-science-of-camera-trapping

[3] A. C. Burton, E. Neilson, D. Moreira, A. Ladle, R. Steenweg, J. T. Fisher, E. Bayne, and S. Boutin, "REVIEW: Wildlife camera trapping: a review and recommendations for linking surveys to ecological processes," *Journal of Applied Ecology*, vol. 52, no. 3, pp. 675–685, jun 2015. [Online]. Available: http://doi.wiley.com/10.1111/1365-2664.12432

[4] S. Young, J. Rode-Margono, and R. Amin, "Software to facilitate and streamline camera trap data management: A review," pp. 9947–9957, oct 2018.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," jun 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[6] Y. Wang and Z. Wang, "A survey of recent work on fine-grained image classification techniques," *Journal of Visual Communication and Image Representation*, vol. 59, pp. 210–214, feb 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320318303754

[7] X.-S. Wei, J. Wu, and Q. Cui, "Deep Learning for Fine-Grained Image Analysis: A Survey," jul 2019. [Online]. Available: http://arxiv.org/abs/1907.03069

[8] P. Guo and R. Farrell, "Aligned to the Object, not to the Image: A Unified Pose-aligned Representation for Fine-grained Recognition," jan 2018. [Online]. Available: http://arxiv.org/abs/1801.09057

[9] W. Ge, X. Lin, and Y. Yu, "Weakly supervised complementary parts models for fine-grained image classification from the bottom up," 2019.

[10] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy," jun 2019. [Online]. Available: http://arxiv.org/abs/1906.06423

[11] Y. Cui, Y. Song, C. Sun, A. Howard, and S. J. Belongie, "Large scale fine-grained categorization and domain-specific transfer learning," *CoRR*, vol. abs/1806.06193, 2018. [Online]. Available: http://arxiv.org/abs/1806.06193

[12] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep., 2011.

[13] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie, "Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015, pp. 595–604. [Online]. Available: http://ieeexplore.ieee.org/document/7298658/

[14] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei, "The Unreasonable Effectiveness of Noisy Data for Fine-Grained Recognition," nov 2015. [Online]. Available: http://arxiv.org/abs/1511.06789

[15] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016.

[16] Z. Ge, C. McCool, C. Sanderson, P. Wang, L. Liu, I. Reid, and P. Corke, "Exploiting temporal information for DCNN-based fine-grained object classification," in *International Conference on Digital Image Computing: Techniques and Applications*, 2016.

[17] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[18] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[19] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[20] H. N. Anh, "keras-yolo3," https://github.com/experiencor/keras-yolo3, 2018.

[21] J. Brownlee, "How to perform object detection with yolov3 in keras," https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/, 2019.

[22] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[23] Z. Li, Y. Yang, X. Liu, S. Wen, and W. Xu, "Dynamic computational time for visual attention," 03 2017.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015.

[25] B. Zhao, J. Feng, X. Wu, and S. Yan, "A survey on deep learning-based fine-grained object classification and semantic segmentation," *International Journal of Automation and Computing*, vol. 14, no. 2, pp. 119–135, apr 2017. [Online]. Available: http://link.springer.com/10.1007/s11633-017-1053-3

[26] P. Guo and R. Farrell, "Fine-grained visual categorization using PAIRS: pose and appearance integration for recognizing subcategories," *CoRR*, vol. abs/1801.09057, 2018. [Online]. Available: http://arxiv.org/abs/1801.09057

[27] A. Dubey, O. Gupta, P. Guo, R. Raskar, R. Farrell, and N. Naik, "Training with confusion for fine-grained visual classification," *CoRR*, vol. abs/1705.08016, 2017. [Online]. Available: http://arxiv.org/abs/1705.08016

[28] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei, "The Unreasonable Effectiveness of Noisy Data for Fine-Grained Recognition," nov 2015. [Online]. Available: http://arxiv.org/abs/1511.06789

[29] A. Dubey, O. Gupta, P. Guo, R. Raskar, R. Farrell, and N. Naik, "Pairwise Confusion for Fine-Grained Visual Classification," may 2017. [Online]. Available: http://arxiv.org/abs/1705.08016