Automatic Species Identification in Camera-Trap Images

Yifan Yu
yifanyu@stanford.edu

Yancheng Li
lycheng@stanford.edu

Tianpei Qian
tianpei@stanford.edu

Abstract

At Jasper Ridge Biological Preserve, heat-change triggered cameras have been deployed for wildlife detection for over 10 years. These cameras generated more than 191,000 photos that were subsequently labeled by volunteers. To relieve the intensive human labor required in the labeling process, we developed an automatic photo classification system using convolutional neural networks. In this process, we experimented with two data preprocessing techniques and four CNN-based pipelines, with our best single model achieving an accuracy of 93.05% and an average F-1 score of 84.39% on the test set. Furthermore, we experimented with different ensemble strategies and model combinations. The best-performing ensemble achieves an accuracy of 93.73% and an average F-1 score of 85.73% on the test set. The system is also equipped two additional features. Firstly, for users that aim for higher prediction accuracy, the system has the option to only settle predictions with confidence higher than a user-specified threshold, and separate out unsettled photos for further human verification. Secondly, the system provides suggested labels and model visualization for unsettled photos to ease the human verification process. With this automatic photo classification system, we hope to largely reduce the human labor required to label camera trap images at Jasper Ridge Biological Preserve.

1. Introduction

Since 2009, Jasper Ridge Biological Preserve (JRBP) has been utilizing heat-change triggered cameras, also known as "camera traps", to capture images of wildlife in the Santa Cruz Mountains[1]. These images, once properly labeled, could help biologists conduct a variety of ecological researches to better study native plants and animals there[1]. However, manually labeling these images remains a labor-intensive and time-consuming task. Within the past 10 years, around $200,000$ camera trap photos have been taken, which took 10 volunteers 650 hours to hand-label. To further validate the labeling results, all labeled photos were verified in a second-round check. In other words, it took volunteers 1300 hours in total to finalize the labeling process. Hence, with more and more new camera trap photos coming in each day, it would be beneficial to have an automatic classifier perform this labeling task to reduce both labor and time cost.

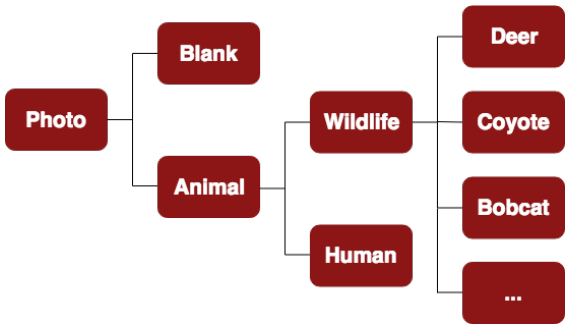

Figure 1: Overview of the classification system.

Provided with all the camera traps photos that have been labeled by JRBP volunteers, we worked on delivering an automatic image classification system that could accomplish the following tasks (see Figure 1): 1) eliminate blank images; 2) Determine whether the animals captured in images are humans or wildlife; 3) Identify the species of the wildlife captured in images.

Our main contributions in this project are as follows. First, we propose two data preprocessing techniques that were designed based on the characteristics of the camera trap photos at JRBP. Secondly, we compare the performance of automatic classifiers constructed using different pipelines, CNN architectures and loss criteria. Then, we perform error analysis and illustrate how to integrate a Confidence-Threshold approach and a Class Activation Map (CAM) with our classifier to improve its practical implementation. Finally, an automatic classification system that is executable by biologists is delivered

to JRBP to label all the future incoming camera trap images.

2. Related Work

Early attempts on automating animal identification in camera trap images highly rely on hand-engineering features. Yu et al.[2] applied sparse coding spatial pyramid matching on cropped images to extract features and built a classifier using support vector machines (SVM). Similarly, S, Matuska et al. [3] developed an animal classification system that requires extensive feature engineering work. Although these systems are able to achieve decent performance, they require problem-specific feature engineering that is not only labor-intensive but also difficult to transfer to other similar tasks.

To address this shortcoming of hand-crafted features, many recent works applied deep learning techniques, which enable automatic feature generation, to classify camera trap images. Cao et al.[4] developed a marina animal classification system using a combination of Convolutional Neural Network (CNN) and hand-crafted image features. Gomez et al. [5] completely abandoned feature engineering and experimented with various CNN architectures to perform animal identification on Snapshot Serengeti dataset. It is worth noticing that the dataset is composed of 48 highly unbalanced animal classes. Gomez et al. [5] combated this imbalanced data issue and achieved a relatively balanced data by removing all the classes with fewer than 1000 images. The class imbalance issue is also present in our dataset but we tackle it with oversampling and a novel pipeline design. In a more recent work, Norouzzadeh et al. [6] proposed a two-stage CNN-based pipeline to perform animal classification on the same dataset. The pipeline eliminates empty photo before making classifications and is reported to have better performance than end-to-end CNN models. This approach is also tested in our work.

3. Datasets

The dataset we work on is provide by Jasper Ridge Biological Preserve and consists of over 191,000 camera trap photos manually labeled by volunteers [7]. These photos were taken by 18 fixed-angle cameras that are triggered by heat changes in the surrounding environments for over 10 years.

3.1. Photo Data

As shown in Figure 2, the photo data contains 24 classes in total and is highly unbalanced. The three major classes are "Human", "Black-Tailed Deer", and "None", which together constitute over 70% of the entire dataset. "Unidentified Birds", "Unidentified Mammals", and "Other" are three classes that require special attention. Images are labeled as "Other" when the identified wildlife is on a short list prepared by biologists at JRBP that consist of rarely-seen species. Volunteers assign "Unidentified Birds" when they are certain about the presence of birds, but fail to confidently identify their species. The same rule applies to the "Unidentified Mammals" class. The fact that these three classes are composite, that is, each contains images of more than one species makes the classification on these three classes especially difficult.

For this project, We randomly split the dataset into three sets: training ($164,861$ photos), validation ($11,487$ photos), and test sets ($11,491$ photos).

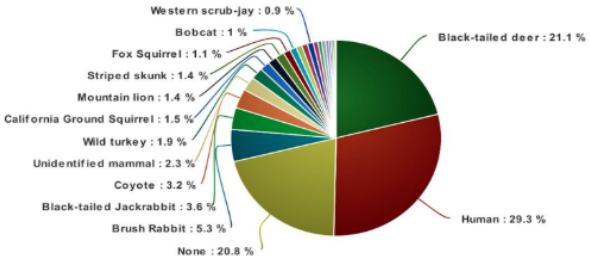

Figure 2: Pie-chart illustrating the distribution of photo data across 24 classes.

3.2. Meta-Data

For each photo, three types of meta-data are provided: 1) camera information, including the model of the camera used and its geo-location; 2) environmental information, including the light level and the temperature when the photo was taken; 3) time information, including the exact date and time when the camera trap was triggered.

4. Methods

4.1. Data preprocessing

Considering the properties of the dataset, we experiment with two data preprocessing steps.

4.1.1 Background Subtraction

As described in the previous section, the photos in our dataset are taken by fixed-angle cameras. Therefore, the background for images coming from the same camera

is more or less unvaried. For this reason, we construct two background images for each of the 18 cameras, one for day and one for night. We then subtract those backgrounds from each photo to make the species in the photo, if any, more visible. The effects of background subtraction are illustrated in Figure 3.

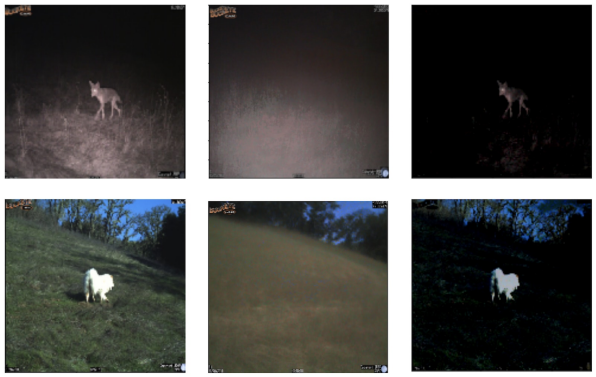

Figure 3: Examples of background subtracted images. The left column contain the original images. The middle column contain the background images. The right column contain the images with backgrounds subtracted.

To get the background, we first group all empty images by camera and by period (day or night). We then experiment with two approaches. The first approach is to average pixels values across all images in the same group. The second approach takes photo illumination into consideration and constructs the background image pixel by pixel. Concretely, we first convert the RGB channel values of the pixels of each image to luma[8], which essentially measures the luminance level of a pixel. We achieve this with the following transformation.

$$Luma = 16 + \frac{65.5}{255}R + \frac{128.6}{255}G + \frac{25.0}{255}B \text{ [8]}$$

Basically, the three multiplicative coefficients measure the luminance perception of different color channels to human eyes. In particular, human eyes are most sensitive to green but least sensitive to blue.

After the transformation, we choose for each pixel a desirable luminance level and then locate the specific image that achieves the luminance level for that pixel.

Finally, we retrieve the RGB tuple for the specific pixel in the selected image and fill it in the corresponding position of the background image.

4.1.2 Oversampling

Class imbalance is another notable property of our dataset, which can have detrimental effects in the training of classifiers [9]. To tackle this problem, we oversample the rare classes during training. Specifically, we replicate samples from classes that represent fewer than 6% of the entire dataset for 5, 10, up to 30 times.

4.2. Classification pipelines

After the data preprocessing steps, we experiment with 4 different classification pipelines.

Pipeline A: The first pipeline is a simple end-to-end classification model.

Figure 4: Pipeline A

Pipeline B: The second pipeline is mainly designed to further battle class imbalance issue besides oversampling. The first model of this pipeline is built to classify images into common species and rare species. Then, it builds a separate classifier for the rare species. In this way, the photos used to train each intermediate classifier have relatively even distribution among the species classes they are responsible for.

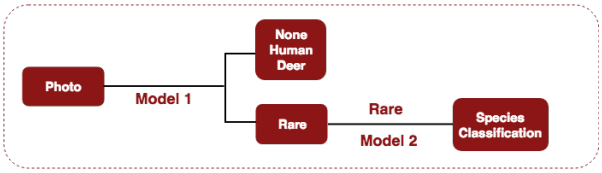

Figure 5: Pipeline B

Pipeline C: The third pipeline treats day and night photos differently due to their very distinct appearance. Another reason to train separate models for day and night photos is that the day photos cover all 24 classes while the night photos only cover 23 classes. We distinguish day and night photos by looking at the light level of each photo. A positive light level indicates that the photo is taken during daytime.

Pipeline D: The last pipeline is built upon Pipeline C and has an additional model to eliminate empty photos before generating species prediction. This additional step can potentially improve the model performance because there exists an intrinsic difference between empty photos and photos that capture an animal. Concretely, empty photos from the same camera should look very

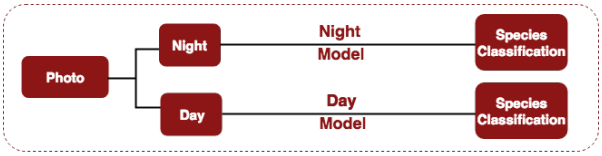

Figure 6: Pipeline C

similar to each other. However, photos with animals have larger variance in their appearance. Even for photos that contain animals of the same species type, they usually look very different due to the various locations, postures and angles of the animals, as well as the intra-class variation for that species.

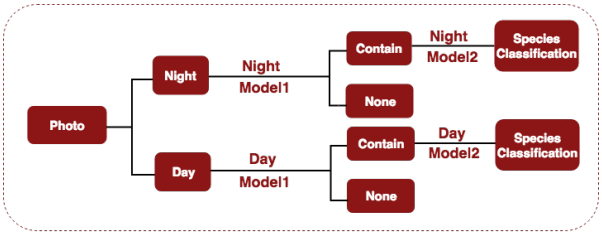

Figure 7: Pipeline D

4.3. CNN architectures

We experiment with 4 different CNN architectures as the classifiers in our pipelines. They are Inception-v3 [10], ResNet-18, ResNet-152 [11] and DenseNet-161[12]. All of these architectures

Inception-v3 [10] is an improved version of the ILSVRC 2014 winner, GoogLeNet [13]. It features the factorization of large convolutions into blocks of smaller convolutions, and auxiliary classifiers that can inject additional gradient at lower layers of the network.

ResNet [11], the winner of ILSVRC 2015, introduces identity skip connections between layers through element-wise summation. This formulation encourages small residual learning and facilitates the training of very deep convolutional networks.

DenseNet [12] takes the idea of skip connections to the extreme by adding shortcuts between every layer and every other layer. Instead of performing element-wise summation, DenseNet combines different layers through concatenation.

All of the above architectures have remarkable performance on the ImageNet challenge [14]. Considering the similarity between the ImageNet challenge and our task, it is reasonable to expect that these architectures can also perform well in this problem setting. To adapt these CNN architectures to our problem, we change the final fully connected layer of these architectures so that the output dimension matches the desired number of classes.

4.4. Loss criteria

To optimize these CNN models, we experiment with 2 loss criteria: softmax loss and SVM loss.

Suppose the output of the last fully connected layer for an image is $\mathbf{s} = (s_1, ...s_N)$, where N is the number of classes. Also, suppose the true label of the image is denoted as y.

Then, the softmax loss for this image is defined by

$$L = -\frac{e^{s_y}}{\sum_{i=1}^{N} e^{s_i}} \tag{1}$$

The SVM loss for this image is defined by

$$L = \sum_{i!=y} \max(0, s_i - s_y + 1) \tag{2}$$

The softmax loss is a variant of the cross entropy loss while the SVM loss is a generalization of the binary SVM loss.

4.5. Ensemble methods

In addition to single CNN classifiers, we have also tried different ensemble methods, which combine multiple classifiers to achieve better performance.

We experiment with two types of ensemble on all the models using the softmax criteria. The first one is the averaging approach. For each image, the softmax probabilities predicted by multiple classifiers are averaged to produce a new softmax probability. The idea is illustrated in Figure 8.

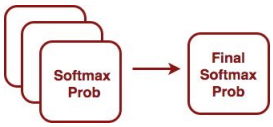

Figure 8: Ensemble with averaging

The second approach is the stacking approach (see Figure 9), where we concatenate the softmax probabilities output by multiple classifiers. The concatenated softmax output is then sent through a classification neural network to get the final softmax probability. The neural network is trained on the training set to maximize the performance of the stacking approach.

Pipeline	Intermediate Model	Intermediate Model Accuracy	Pipeline Accuracy
Pipeline A	Full Model	92.97%	92.97%
Pipeline B	Model 1 Model 2 (Rare)	94.52% 87.76%	92.21%
Pipeline C	Model (Day) Model (Night)	93.76% 89.03%	92.04%
Pipeline D	Model 1 (Day) Model 2 (Day) Model 1 (Night) Model 2 (Night)	95.65% 95.31% 94.51% 90.70%	92.09%

Table 1: Accuracy of the best classifier of each pipeline on the development set. All the best classifiers are achieved with models using softmax loss criterion and DenseNet161.

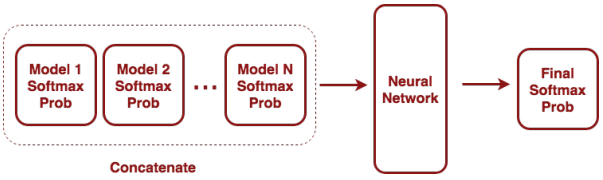

Figure 9: Ensemble with stacking

5. Results

5.1. Classifier Comparison

5.1.1 Single Classifier

With different combinations of pipelines, CNN architectures, and loss criteria, we conducted a total of 41 experiments to build the classifier. We performed transfer learning on all the models involved in the 41 experiments with pre-trained ImageNet weights. Table 1 shows the accuracy results and implementation details for the best classifier found of each pipeline.

All of the 4 pipelines achieved promising results, with the accuracy rates varying between 92.04% and 92.97%. In both Pipeline C and D, models handling day-time photos outperformed their counterparts, which handled night-time photos. This fact suggests that it is easier to classify day-time photos, which conforms to our expectation, as wildlife captured in day-time photos are typically more visible than those captured in night-time photos, where often only animals' glaring eyes could be seen.

Best Single Classifier As shown in Table 1, the best classifier constructed using Pipeline A reached the highest performance on the development set, which is selected as our best single classifier. During the training of this classifier, we used a mini-batch size of 16 photos, and implemented Stochastic Gradient Descent (SGD) with Nesterov momentum. The initial learning rate is 1e-3 and learning rate decay is scheduled for every 5 epochs with scale of 0.5. One explanation for the fact that this classifier achieved the most satisfying result is that its pipeline structure involves no intermediate steps, and therefore the trained model saw more data than the models involved in other pipelines. The additional photos it saw helped it learn species information more accurately.

We evaluated the best single classifier's performance on the test set, and achieved an accuracy of 93.05%. We also examined its by-class recall, precision, and F1-score, and the result is presented in Table 2. This detailed result shows that our classifier achieves great performance on most of the categories, with F1-scores varying between 0.73 to 0.99. Yet there are three classes with unusually low recall, precision and F1-scores, which are "Other", "Unidentified Mammals," and "Unidentified Birds." The poor model performance for these three classes was also observed in the development set and led to the discovery of an important data limitation in our project, which will be discussed in detail in section 5.2.

Class Name	Recall/Precision/F1-Score
Human	0.99/1.00/0.99
Wild turkey	0.97/0.98/0.98
Black-tailed deer	0.97/0.98/0.98
Mountain lion	0.95/0.99/0.97
Striped skunk	0.93/0.95/0.94
Gray Fox	0.93/0.93/0.93
Coyote	0.94/0.92/0.93
None	0.95/0.89/0.92
Raccoon	0.93/0.90/0.91
Black-tailed Jackrabbit	0.90/0.91/0.90
Brush Rabbit	0.91/0.89/0.90
California Ground Squirrel	0.92/0.87/0.90
Bobcat	0.88/0.89/0.88
Great blue heron	0.84/0.91/0.87
Western scrub-jay	0.84/0.90/0.87
Virginia Opossum	0.87/0.85/0.86
Fox Squirrel	0.81/0.86/0.83
Stellers jay	0.76/0.87/0.81
California quail	0.89/0.68/0.77
Western Gray Squirrel	0.76/0.79/0.77
Dusky-footed Woodrat	0.69/0.76/0.73
Unidentified Bird	0.45/0.68/0.55
Unidentified mammal	0.47/0.61/0.53
Other	0.44/0.62/0.52

Table 2: Performance of the best single classifier on the test set in terms of recall, precision, and F1-score. The average F1-score is 0.8439. The table is sorted by descending F1-score.

5.1.2 Ensemble Classifier

We performed both averaging ensemble and stacking ensemble using the classifiers constructed during the 41 experiments. The best ensemble classifier we found utilized the stacking technique with 6 single classifiers, whose detailed pipeline and CNN architectures are presented in Table 3. It achieves a total accuracy of 93.73% on the test set, which is higher than that of the best single classifier.

5.2. Data Limitation

As mentioned in section 5.1.1, compared with the other classes, our best single classifier achieved unusually low recall, precision and F1-scores on three categories, which are “Other”, “Unidentified Mammals”, and “Unidentified Birds”. Further investigation was conducted on these three classes through manual error analysis. For some of the misclassified cases annotated as one of these three categories by volunteers, our predicted labels might be more appropriate than the original labels. Figure 10 shows a typical case. The original label is “Other”, yet our classifier labels it as ”Wild Turkey”, which is more appropriate for this photo. Biologists helped to relabel all the misclassified cases in these three categories in the development set (a total of 298 photos), and found that the predicted labels for 153 of them were more appropriate than the corresponding original labels. Due to this labeling quality issue in the original data set, we believe the accuracy metrics reported in this paper are conservative measures of our classifier’s true performance.

Pipeline	Models	CNN Architecture
Pipeline A	N/A	ResNet-152
Pipeline A	N/A	Inception-v3
Pipeline A	N/A	DenseNet-161
Pipeline A	N/A	DenseNet-161 (Oversample 15x)
Pipeline C	Model (Day)	DenseNet-161
	Model (Night)	DenseNet-161
Pipeline D	Model 1 (Day)	DenseNet-161
	Model 2 (Day)	DenseNet-161
	Model 1 (Night)	DenseNet-161
	Model 2 (Night)	DenseNet-161

Table 3: Implementation details of the 6 single classifiers used in the best ensemble classifier, including pipeline structure, CNN architecture, and preprocessing techniques if any

Figure 10: An example of the misclassified photos in the “Other” category

5.3. Confidence-Threshold Approach

According to Table 2, our classifier performs better in identifying classes such as "None" and "Human". Examining the predictions in these classes led to the finding that our classifier is typically more confident about predicting these classes, which is indicated by higher top 1 softmax probabilities. Inspired by this discrepancy in prediction confidence, we decided to provide our clients with the possibility to split photos based on prediction confidence level. Photos with confidence level below a certain threshold will go into a separate folder with unsettled classification decisions.

For example, if a biologist were to choose a threshold of 0.85, our system would classify all the images but only settle classification decisions for images whose top 1 softmax probabilities were higher than 0.85. Specifically, we would only settle predictions for 88.1% of the images and in return achieve a 97.1% accuracy. For the rest 11.9% of the images, the system would suggest human confirmation. That is, we would provide visualized results for biologists to confirm the predictions in a second-round check. Biologists have the flexibility in deciding their favorable balance between prediction accuracy and amount of data predicted by adjusting the confidence threshold. The detailed trade-off is shown in Figure 11.

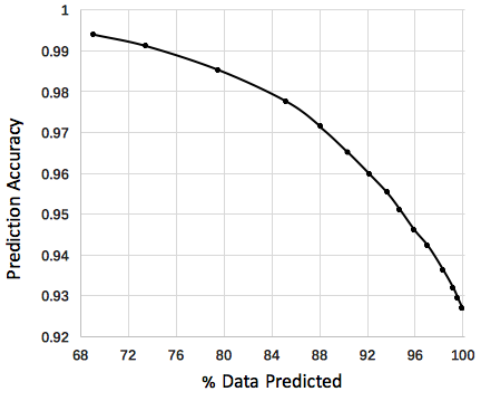

Figure 11: Confidence-Threshold Trade-off Curve. Illustrates the trade-off between portion of data predicted and prediction accuracy

5.4. Model Visualization: Class Activation Map

Photos that require further confirmation can be addressed easily with our provided visualization. The visualized results are produced by class activation maps (CAM) [15]. CAM computes a weighted-sum of the feature map with learned weights from the final convolutional layer. The results are then up-sampled to match the size of the input image. It allows us to easily visualize the predicted class scores on an input photo through highlighting the areas that most influenced the classifier's decision. Color encodes the importance of different parts in the photo detected by our model. Specifically, the red color represents the part of the image that our prediction highly relies on.

With this tool, experts can easily locate where our predictions come from, even for hard cases like in Figure 12. In this example, it would be really hard for volunteers to tell immediately if there is any animal and where it is in the photo. With our visualized prediction (right photo in Figure 12), we can easily locate the squirrel and confirm the prediction. Aided by the visualization tool, specialists can verify labels for hard cases in a more efficient manner.

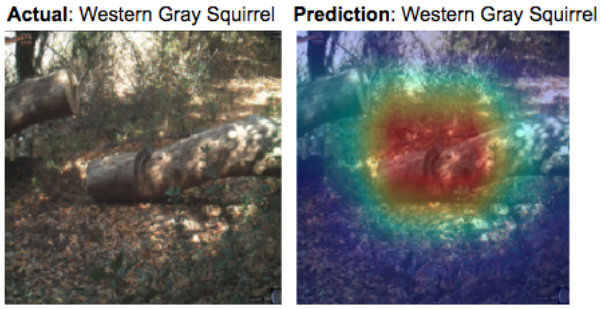

Figure 12: Examples of the CAM generated for prediction via our classifier. (At full resolution the effect is more visible.)

6. Conclusion

In conclusion, we developed a CNN-based photo classification system for Jasper Ridge Biological Preserve. We experimented with 4 different classification pipelines and conducted extensive search for hyperparameters, including data preprocessing techniques, CNN architectures, loss criteria and ensemble strategies. The best single classifier is modified from DenseNet-161 and achieves an accuracy of 93.05% and an average F-1 score of 84.39% on the test set. The best ensemble mixes six different models with the stacking approach and achieves an accuracy of 93.73% and an average F-1 score of 85.73% on the test set. Furthermore, we augmented our system with two additional features. Firstly, the system allows users that aim for higher prediction accuracy to only settle predictions with confidence higher than a given threshold, and leave unsettled photos for

further verification by human. Secondly, to facilitate the human verification process, the system also provides suggested labels and model visualization for those unsettled cases.

In the future, we plan to introduce a human-in-the-loop mechanism that can constantly improve the model performance. With the continuous human supervision for hard cases, the model can actively learn from those validated new examples. We hope in the end the system can reach an accuracy level that can match human performance and hence completely eliminate the human labor required for the labeling process.

Code

Our development code can be accessed at:
https://github.com/qiantianpei/WildAnimalDetection

References

[1] Jasper ridge biological preserve official website. `http://https://jrbp.stanford.edu`. Accessed: 2018-06-08.

[2] Xiaoyuan Yu, Jiangping Wang, Roland Kays, Patrick A Jansen, Tianjiang Wang, and Thomas Huang. Automated identification of animal species in camera trap images. *EURASIP Journal on Image and Video Processing*, 2013(1):52, 2013.

[3] Slavomir Matuska, Robert Hudec, Miroslav Benco, Patrik Kamencay, and Martina Zachariasova. A novel system for automatic detection and classification of animal. In *ELEKTRO, 2014*, pages 76–80. IEEE, 2014.

[4] Zheng Cao, Shujian Yu, Bing Ouyang, Fraser Dalgleish, Anni Vuorenkoski, Gabriel Alsenas, and Jose Principe. Marine animal classification with correntropy loss based multi-view learning. *arXiv preprint arXiv:1705.01217*, 2017.

[5] Alexander Gomez, Augusto Salazar, and Francisco Vargas. Towards automatic wild animal monitoring: identification of animal species in camera-trap images using very deep convolutional neural networks. *arXiv preprint arXiv:1603.06169*, 2016.

[6] Mohammad Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Alexandra Swanson, Meredith S Palmer, Craig Packer, and Jeff Clune. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning. *Proceedings of the National Academy of Sciences*, page 201719367, 2018.

[7] Jasper Ridge Biological Preserve Camera Trap Photos, Stanford University. 2008-2018.

[8] Charles Poynton. *Digital video and HD: Algorithms and Interfaces*. Elsevier, 2012.

[9] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *arXiv preprint arXiv:1710.05381*, 2017.

[10] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[12] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[14] Pytorch models documentation. `https://pytorch.org/docs/stable/torchvision/models.html`. Accessed: 2018-06-09.

[15] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 2921–2929. IEEE, 2016.

[16] Michael Stokes, Mattew Anderson, Srinivasan Chandrasekar, and Ricardo Motta. A standard default color space for the internetsrgb, 1996. *URL http://www. w3. org/Graphics/Color/sRGB*, 2012.

[17] Mohammed Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Ali Swanson, Meredith Palmer, Craig Packer, and Jeff Clune. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning.