Wildlife Classification Through Transfer Learning and Architecture Optimization

Zhexu Li
UC San Diego, Electrical and Computer Engineering
A14532514

Juo-Yun Chen
UC San Diego, Electrical and Computer Engineering
A59010410

Abstract

The effectiveness of wildlife preservation relies heavily on in-depth understandings of the diversity and behavior of animals, which requires intensive manual labor of experts to correctly label the images collected. In this project, we will construct a robust deep learning model to automate this process, through transfer learning with various state-of-the-art deep learning architectures, and improving network structures to better fit our tasks. Trained on a heavily augmented dataset of animals, our improved Wide-Resnet50 model achieved an impressive test accuracy of 96.2%, which introduced over 5% improvement over the vanilla Wide-ResNet model, with a little extra computational cost.

1 Introduction

Wildlife conservation is an essential part of environment protection, a diverse wildlife provides balance and stability to the ecosystem. Modern wildlife management requires in-depth understanding about the behavior and diversity of different animals in the habitat, usually through images collected by sensors and cameras. Traditionally, such insights were produced by human experts who manually label and study those images. But with more cameras installed and more images being collected, manually labeling large amount of images generated everyday becomes essentially infeasible. An automated wildlife classification process can save tremendous amount of valuable man-hours of wildlife experts, and free the experts from the tedious classification routine so they can focus on more important tasks.

In this project, we aim to build an accurate wildlife classification model to automate the labeling process. We plan to construct a deep neural network based on current state-of-the-art models for the task, and utilize transfer learning to accelerate the training process, and compare their performance. The resulting model will be helpful for automatic wildlife classification, and it could provide valuable information about characteristics of certain species.

2 Problem Formulation

Like mentioned above, the goal of our project is to create a deep learning model for accurate wildlife classification. To be more specific, the objective is to create a model which can achieve a high classification accuracy on the testing set. Since our project is essentially a classification task, it's natural to use the Cross-Entropy loss on the training process, which is defined by:

$$Loss = -\Sigma_{c=1}^{M} y_{o,c} log(p_{o,c}) \tag{1}$$

Where M is number of classes we need to classify, p is the predicted probability that our prediction o is of class c, and y is the binary indicator for the classification which is defined as:

$$y = \begin{cases} 0, & if \quad o \neq c \\ 1, & if \quad o = c \end{cases} \tag{2}$$

To evaluate the model's performance, we calculate its' classification accuracy on the validation and testing set, which is defined by:

$$Accuracy = \frac{Number \quad of \quad Correct \quad Classifications}{Total \quad Number \quad of \quad Classifications} \tag{3}$$

3 Related Works

In *Insights and Approaches using Deep Learning to Classify Wildlife* [6], Miao et al. tried several convolution neural networks to classify wildlife from a camera-trap dataset which contains 111,467 images of 20 species. They employed multiple novel modifications in their convolution layer and their best accuracy was 87.5%. One important reason behind their high accuracy is that their dataset was very clean and uniform, and every specie has over 5000 training images available, which is almost the size of the entire raw dataset we used in our project. Such dataset is very unpractical in real world scenarios since they are often too expensive to collect.

In *Assessing Convolutional Neural Network Animal Classification Models for Practical Applications in Wildlife Conservation* [7], Larson applied ten different CNNs models on seven datasets to simulate 13 scenarios about wildlife conservation. Her experiments revealed some important aspects people should consider in order to achieve optimal performance when selecting or training a CNN for use in a wildlife camera-trap project.

In *Automatically Identifying, Counting, and Describing Wild Animals in Camera-Trap Images with Deep Learning* [8], Norouzzade et al. utilized CNN to identify and count different species of animals from camera-trap data. They worked with a 3.2 million-image Snapshot Serengeti dataset and achieves an accuracy of 93.8%, but instead of focusing on classifying wild animals, they focused more on simplifying the background environment of the images, and understanding the behavior of the wildlifes in various environments.

Overall, the above works are inspirational for our project. Despite their focus and dataset are often quite different compared to ours, they still provided some valuable ideas which were helpful during the development of our own models.

4 Raw Dataset and Image Augmentation

The raw dataset we used for our project is an animal image dataset collected by Sourav Banerjee [1], which consists of 5400 animal images in 90 different classes (60 each). The dataset turned out to be tricky to work with because of its relatively small size (only 60 images for every specie) and very irregular image resolutions (ranges from 70 x 70 to 1920 x 1080). Originally, we tried a naive approach to simply resize the images and feed them to the network, but this approach turned out to hurt the performance a lot. Therefore, we experimented with various image augmentation techniques to overcome this issue.

We have applied different transformations that makes sense for our object, including adding color jitters, random rotations, horizontal and vertical flips and converting the image into a gray scale images. Some other common augmentation techniques, like noising, were not used because many images were already quite noisy. After applying data augmentation, we have a total of 10505 images of 88 species (we found 2 duplicates in the raw dataset) in the dataset. The augmentation process almost doubled the amount of images in our dataset, and it will be helpful for us to create a more robust classifier. We split the dataset into 70% training set, 20% validation set and 10% test set.

Figure 1: Sample images from the raw dataset (left) and augmented dataset (right).

5 Approach

5.1 Optimizer

To minimize the Cross-Entropy loss mentioned above, we chose to use the Adam optimizer, which for every step it calculates:

$$m_t = \beta_1 m_t + (1 - \beta_1)(\frac{\partial L}{\partial w_t}) \tag{4}$$

$$V_t = \beta_2 V_t + (1 - \beta_2)(\frac{\partial L}{\partial w_t})^2 \tag{5}$$

Which it basically combines momentum and RMSProp (Root Mean Square Propagation) together, where $\beta 1$ and $\beta 2$ are the parameters for momentum and RMSProp, respectively. After that, we have:

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{6}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{7}$$

And finally we can update our weight w at step t as:

$$w_t = w_{t-1} - \gamma \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon} \tag{8}$$

Where γ is the learning rate defined for this algorithm. Adam optimizer is a very popular and powerful optimizer because the combination use of momentum and RMSProp which effectively stabilizes training and accelerates convergence. We will use it as our optimizer in all of our experiments.

5.2 Baseline Selection

In order to build an effective classification network, we first have to select a proper baseline model which performs well enough so it's worthwhile for improvements and transfer learning. We will try a number of state-of-the-art deep learning architectures, including VGG 16, Resnet 50, Wide Resnet50, and DenseNet 121. We will also learn the intuitions behind every architecture, so we can improve them later. The details about these networks will be discussed below.

5.2.1 VGG 16

The VGG architecture, proposed by Karen Simonyan and Andrew Zisserman in 2013 [2], serves as a significant milestone in the realm of computer vision. The model used a small 3x3 receptive filters throughout the entire network, and every convolutaional layer was followed by a non-linear activation layer. The number of parameters was reduced significantly compared to previous models, which helped with reducing overfitting. The VGG 16 model we used in our project is consisted of 16 convolutional and fully connected layers.

5.2.2 Resnet 50

The Resnet architecture, introduced by He et al. [3], tackled a problem that, as the neural network grows deeper and deeper, the accuracy saturates and even degrades at a certain point due to the issue of vanishing gradient. To reduce the gradient saturation, residual learning was introduced. The residual block allows the acitvation from the previous layer to skip connection and be added to the deeper layer's activation as seen in figure 2. The emergence of residual learning made the previously unpractical very deep neural networks feasible again. The Resnet 50 architecture we tried in our experiments contains 50 convolutional layers, with residual blocks consists of 2 - 3 convolutional layers (filter size 1 x 1) skip connecting every 3 layers.

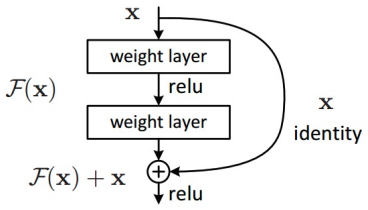

Figure 2: Residual Block for Skip Connection.

5.2.3 Wide Resnet 50

Wide Resnet, proposed by Sergey Zagoruyko et al. in 2017 [4], solved the problem of diminishing features reuse which is commonly seem in very deep residual networks, by introducing a modified architecture of Resnet: the Wide Resnet (WRN). The intuition behind the modification was simple, basically they decreased the width and increased the depth of the residual blocks, which allowed faster training and higher performance [4]. The structure of Wide Resnet is shown in figure 3. The WRN50 architecture we used in our experiments contains 50 layers with the layout similar to Resnet50, but residual blocks replaced by the WRN version.

group name	output size	block type = $B(3,3)$
conv1	32×32	[3×3, 16]
conv2	32×32	$\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix} \times N$
conv3	16×16	$\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix} \times N$
conv4	8×8	$\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix} \times N$
avg-pool	1×1	$[8 \times 8]$

Figure 3: WRN Architecture [4]

5.2.4 DenseNet 121

DenseNet was introduced by Gao et al. in 2016 [5], it was a novel architecture which directly connects each layer to every other layer in a feed-forward fashion. The network brought significant improvements over other existing architectures and was famous for its impressive efficiency. The DenseNet 121 model we tried in our experiments includes 121 densely connected convolutional layers.

We will compare their training error curves and validation accuracy, and selecte the best performing one to be the baseline model. To ensure fairness of comparison, all models will be trained and evaluated on the same setting, using the Cross-Entropy loss and Adam optimizer, with weights pretrained on ImageNet, batch size (64), number of epochs (10), and the best learning rates for each model selected from trials. The results of our experiments will be discussed in the Experiments and Results section. And we will build our own improved model based on the baseline model selected, by resolving problems we observed during the training.

5.3 Improving Architecture

After we experimented with the models mentioned above, we will select the best performing network to be our baseline model to be transfer learned from. We will experiment with various techniques in efforts to optimize the architecture. Since overfitting is a very common problem in classification tasks, especially for the very deep architectures we use in our experiments, we will mainly focus on adding extra regularization terms. Among the popular regularization techniques, Batch Normalization and Dropout usually works well. The batch normalization is defined by:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{VAR[x^{(k)}]}} \tag{9}$$

Where x is a batch of features or weights calculated by the network. While batch normalization acts like a regularization term, it also stabilizes and accelerates the training process by normalizing the activation vectors from hidden layers.

On the other hand, the intuition behind Dropout is even simpler, it basically force the network to forget a certain portion of information it learned, by randomly dropping a percentage of weights learned in the previous layers. Dropout is a quite powerful regularization technique, very helpful for reducing overfitting.

We will try different combinations of regularization techniques. Again, to ensure fairness, all experiments will be ran on the same setting, which is Cross-Entropy loss, Adam Optimizer, batch size = 64, number of epochs = 10, and the best learning rates. The performance of our best resulting modification will be discussed in the Experiments and Results section later. The optimal modification we create should provide considerable improvement in validation accuracy over the vanilla baseline model, while maintaining computationally efficient.

5.4 Visualize Network

After we come up with an improved model, we will try to visualize its decisions to better understand its behavior and the intuition behind its decision process. We will extract the last convolutional layer weights from the model, and visualize it using Class Activation Map (CAM). We will visualize that using a selected set of image, each representing a different scenario, and try to explain them. The CAM plots generated will be discussed at the end of the Experiments and Results section.

6 Experiments and Results

6.1 Baseline Selection Experiments

To select an appropriate baseline model for our task, we ran a number of experiments using the VGG 16, Resnet 50, Wide Resnet 50, and DenseNet 121 on the settings mentioned above. The dataset was splitted into three parts: 70% training, 20% validation, and 10% testing. We only evaluated the models' performance on the validation set in the baseline selection experiments, and the test dataset was reserved for comparison between the selected baseline model and our improved model.

The pretrained VGG 16 model achieved a best validation accuracy of 65% after being trained for 10 epochs. The Resnet 50 architecture got a 86% validation accuracy at the end of training, using learning rate = 0.0001. The DenseNet 121 model received a validation accuracy of 81% using the same 0.0001 learning rate. The WideResnet 50 model achieved the highest validation accuracy among all models we experimented with: 89%, also using the 0.0001 learning rate.

The experiment results, including training error curves and validation accuracy curves over the 10 training epochs, are displayed in the figure 4. Based on the plots we can see the VGG 16 was heavily overfitting to the training set, its validation accuracy was decreasing near the end of the training. The Resnet 50 and DenseNet 121 also behaved similarly, with more obvious overfitting further into the training. On the other hand, the Wide Resnet 50 model, while also overfitted, converged faster than other models, and had higher validation accuracy than other models throughout the entire 10 training epochs. Because the WRN 50 model was the best performing network in our baseline selection experiments, we chose it to be our baseline model for transfer learning, and in the next section we will discuss how we improved it to better fit our tasks.

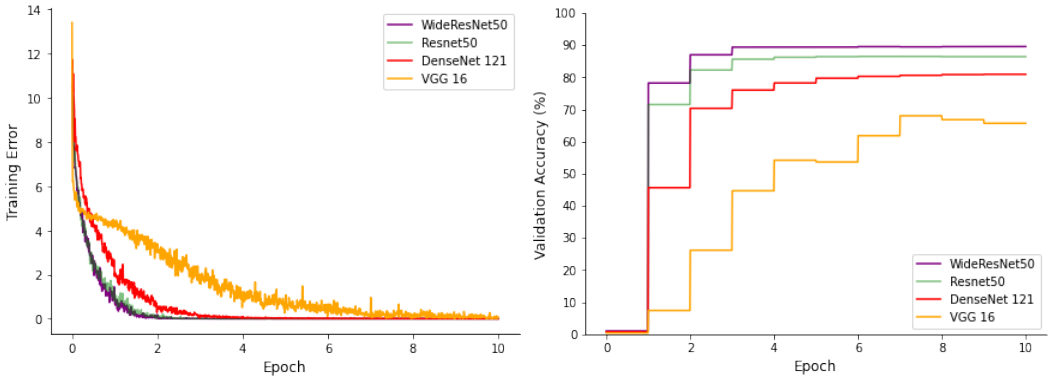

Figure 4: Training Error (left) and Validation Accuracy (right) for Baseline Selection Experiments.

6.2 Improved Architecture

As mentioned in section 5.3, we experimented with various techniques in efforts to improve the performance of the Wide Resnet 50 model. We mainly focused on trying different combinations of regularization techniques because we observed overfitting during the baseline selection experiments. After a lot of trials and errors, we came up with an extra "Regularization Block" which replaces the last fully connected layer in WRN 50 with a linear layer, a Batch Normalization layer, a Dropout layer with 20% dropout rates, and a linear layer. Basically, after the model outputs the average pooled results from the last convolutional layer, we feed the outputs into a fully connected layer which "wraps up" the pretrained weights and shrink them from 2048 channels to 256, then go through a batch normalization to regularize the outputs, then dropout 20% of the learned weights for further regularization, and feed the regularized results to a fully connected layer to produce the 88 classes classification. The structure of our "Regularization Block" and the improved architecture is shown in figure 5.

We ran our improved model using the same settings we used for training the WRN 50 network (lr = 0.0001, batch size = 64, epochs = 10). The resulting training and validation curves are shown in figure 6. In the plots we can see our improved architecture brought considerable improvements over the original WRN 50 network, with 4.5% higher validation accuracy (93.7% vs. 89.2%). It's also clear that our model converged way faster than other models during training, and was constantly outperforming all other models by a considerable amount. We then evaluated our model and the vanilla WRN 50 model on the testing set, our improved model achieved 96.2% test accuracy, which is over 5% higher than the 91.2% test accuracy of the vanilla WRN 50. The high testing accuracy suggests our improved model generalizes pretty well to unseem images. Overall, our model achieved impressive improvement over the vanilla WRN 50 network, while involving a little extra computational cost (only one extra fully connected layer, and one replacement).

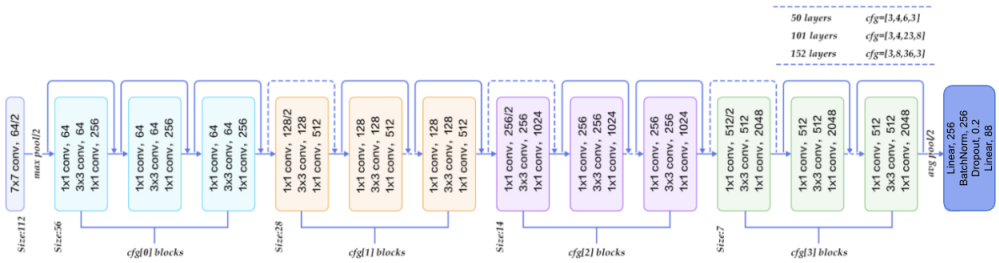

Figure 5: Our Improved Architecture with Regularization Block.

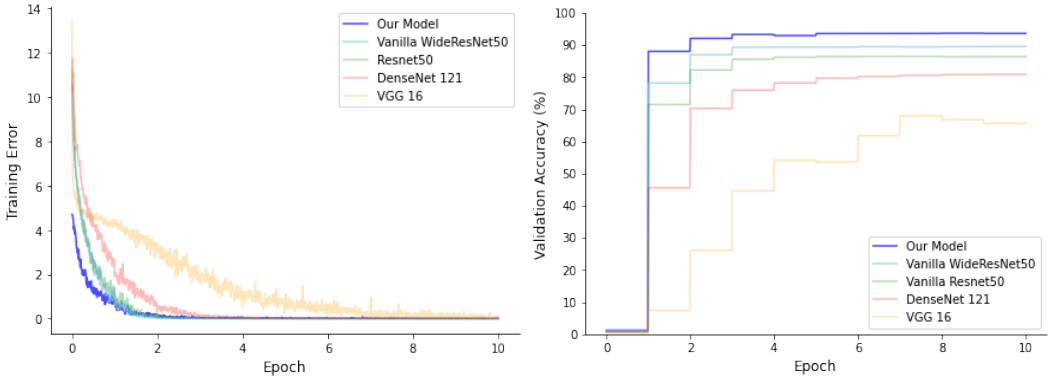

Figure 6: Training Error (left) and Validation Accuracy (right) for All Models.

6.3 Model Visualization

After we created our improved model, we also visualized its last convolutional layer features using the Class Activation Map mentioned above to better understand its decisions. Here, we will discuss about several interesting cases.

In figure 7 we can see our model worked well on both raw images and processed (rotated, color jittered, and downsampled) images, producing correct classifications with extremely high confidence. This is partially thanks to the heavy image augmentation process which helped building a robust classifier.

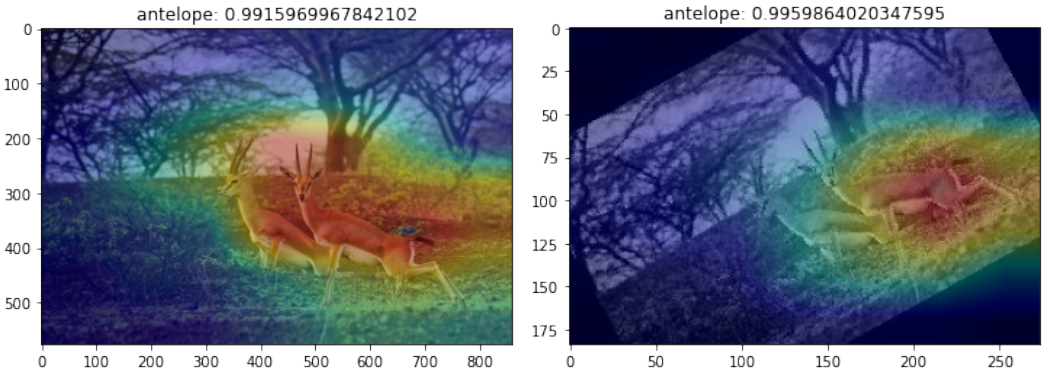

Figure 7: CAM of Raw Image (left) and Processed Image (Right).

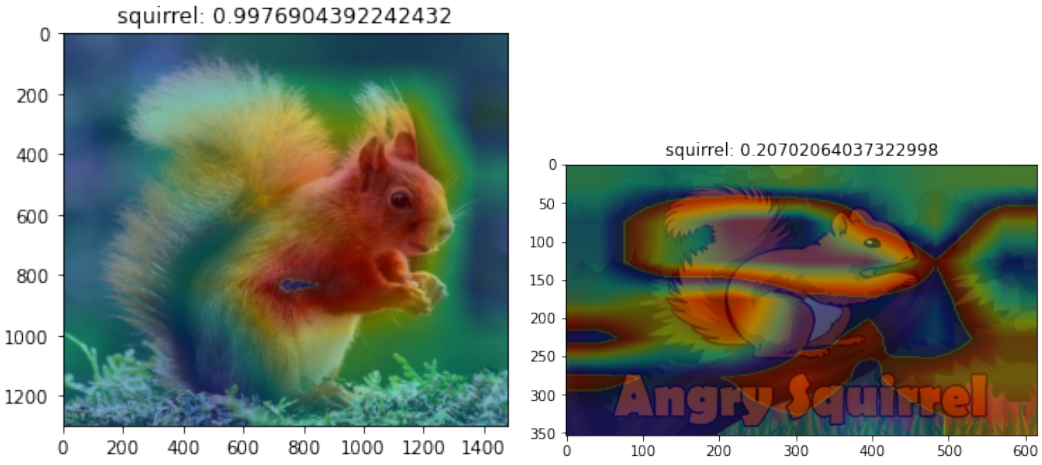

Figure 8: CAM of Photo (left) and Cartoon (Right).

In figure 8, we can see while our model had high confidence in classifying animal photos, it had a hard time at classifying cartoon images. Though the classification was correct, the confidence was way lower, and the activation map was messy. This is likely because of the lack of cartoon images in our dataset. And the capability of the classifier is heavily determined by the quality of the dataset it is trained on. Also, the purpose of our model is to classify photos of wildlifes, so it doesn't matter if it's not working properly on cartoon images.

In figure 9, there are 3 somewhat misclassified images which are interesting to look at. The first one is a human dressing as a cat, the CAM shows that our model made its decision based on cat-like features (cat ear, cat outfit), and considering our training dataset doesn't contain human, it's impossible to predict the correct label, so we can say this case is somehow both correctly classified and misclassified. The second image is a reindeer misclassified to be an antelope with fairly high confidence. Deers and antelopes do have similar shapes and horns, so it's reasonable for our model to be confused by the two. The image on the top right is a deer misclassified as a kangaroo with <50% confidence, and one may argue the highlighted portion indeed looks like a kangaroo, especially the ears.

Overall, the CAM show that the model we built was quite robust at classifying wildlife images. Though there were several edge cases it misclassified, we believe it can become more powerful if it's trained on a bigger and higher quality dataset.

7 Conclusion and Future Works

In this project, we have experimented with a number of state-of-the-art deep learning models on an animal image dataset which was heavily augmented using various image augmentation techniques. Using the Wide Resnet 50 architecture as the baseline model, we went through a lot of trials and errors and came up with a custom "Regularization Block" which turned out to improve the performance of the network significantly. Our improved model achieved a test accuracy of 96.2%, which was over 5% higher than the vanilla WRN 50 network, while still maintaining computationally efficient. Our analysis on visualizations of the network also proved that our model is quite robust at classifying animal images even heavily processed. In the future, we can train our model on a bigger and higher quality dataset, if it's available. We hope that our project would contribute to the important task of wildlife preservation.

References

[1] Animal Image Dataset (90 Different Animals). (2022, February 21). Sourav Banerjee, Kaggle. https://www.kaggle.com/datasets/iamsouravbanerjee/animal-image-dataset-90-different-animals

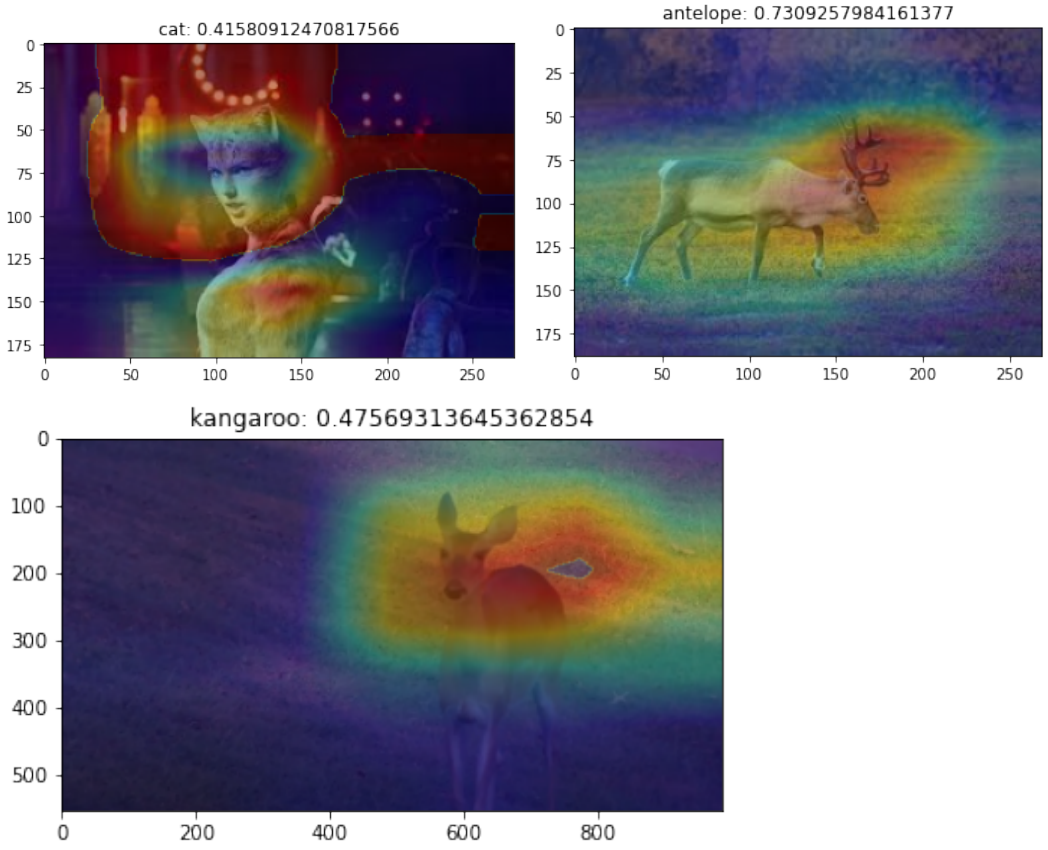

Figure 9: CAM of the Misclassified Images.

[2] Very deep convolutional networks for large-scale image recognition. Karen Simonyan and Andrew Zisserman. In ICLR, 2014.

[3] Deep Residual Learning for Image Recognition.Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. In arXiv:1512.03385.

[4] Zagoruyko, S. et al. (2016, May 23). Wide Residual Networks. arXiv.Org. https://arxiv.org/abs/1605.07146

[5] Densely Connected Convolutional Networks.Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger. In arXiv:1608.06993.

[6] Miao, Z., Gaynor, K.M., Wang, J. et al. Insights and approaches using deep learning to classify wildlife. Sci Rep 9, 8137 (2019). https://doi.org/10.1038/s41598-019-44565-w

[7] Larson, Julia, "Assessing Convolutional Neural Network Animal Classification Models for Practical Applications in Wildlife Conservation" (2021). Master's Theses. 5184. DOI: https://doi.org/10.31979/etd.ysr5-th9v https://scholarworks.sjsu.edu/etd_theses/5184

[8] Norouzzadeh, M.,Nguyen A., Kosmala, M. et al. "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning". DOI: https://doi.org/10.1073/pnas.1719367115