



Hyperparameter Tuning Improves Computer Vision Tools for Wildlife Conservation

Achyut Venkatesh

Abstract

Camera traps are an excellent way to collect a large amount and variety of wildlife data; however, the volume of images poses a serious challenge for manual analysis. Through the use of machine learning and computer vision, this process can be automated, enabling rapid and accurate classification and identification, thus aiding conservation efforts. This study explores the effectiveness of EfficientNet models in classifying eight different classes from camera trap images, focusing on how hyperparameters such as learning rate and batch size affect model performance. Using a dataset of 16,487 images from Tai National Park, we experimented with different hyperparameter values and found that a lower learning rate and a moderate batch size yielded the highest accuracy. The `efficientnet_b1` model was the most effective model, achieving 86.27% accuracy with 20 epochs. It identified leopards, hogs, birds, and genet civets with over 90% accuracy but struggled with the blank class. Training was completed in under eight hours on a single laptop, showcasing the efficiency of lightweight models. Our findings underscore the potential of computer vision in conservation, enabling rapid and accurate analysis of large datasets that would take large amounts of time and workers to go through manually. This work highlights the importance of hyperparameter tuning in enhancing model performance, paving the way for more effective automated wildlife monitoring tools.

Introduction

Camera traps are invaluable tools for conservation, providing non-invasive methods to monitor wildlife populations and behaviors over large geographic areas and extended periods. These devices can capture images of animals in their natural habitats without human interference, offering crucial data for ecological research and conservation planning. However, the sheer volume of images generated by camera traps presents a significant challenge for manual analysis.

Machine learning (ML) and computer vision present a solution to this challenge. These technologies allow for the data analysis to be automated, significantly reducing the time and effort required to comb through the data. They allow for tasks such as classification and identification to be completed at astonishing speeds. By using ML models, conservationists can process large datasets quickly and accurately, gaining insights that inform conservation strategies and policy decisions.





Several studies have already demonstrated the effectiveness of ML in conservation efforts. One study utilized deep learning to classify 48 species from camera trap images, achieving high accuracy and demonstrating the potential for large-scale automated wildlife monitoring (Norouzzadeh et al., 2020). Another study developed a deep learning model to identify animal species in camera trap images across diverse ecosystems, highlighting the generalizability of these methods (Willi et al., 2018).

In this paper, we aim to evaluate how different hyperparameters affect the performance of a computer vision model trained to classify camera trap images. Hyperparameters are predefined settings in machine learning models, like learning rate, batch size, and the number of epochs,

that control the training process and significantly influence model performance. We focus on the EfficientNet computer vision model architecture and assess the impact of varying the batch size , learning rate, and epoch hyperparameters on the model's training efficiency and accuracy. By tuning these three hyperparameters and identifying the optimal values, we aim to maximize the model's performance in identifying animal species from a camera trap dataset. This work contributes to the broader goal of enhancing automated wildlife monitoring tools, making them more effective and accessible for conservation efforts worldwide.

Data

The data used for this project were camera trap images taken from various locations across the Tai National Park in Côte d'Ivoire. These images were collected by the Wild Chimpanzee Foundation and the Max Planck Institute for Evolutionary Anthropology from 2016 to 2018. The dataset used had a total of 16,487 images split into 8 unique classes, each containing a different species type. The 8 classes were the Duiker Antelope, Genet Civet, Hog, Leopard, Prosimian Monkey, Rodent, Bird, and a blank class. Table 1 below shows a sample image from each class.

<p style="text-align: center;">Duiker Antelope</p>  <p style="text-align: right; font-size: small;">02-03-2017 15:10:46</p>	<p style="text-align: center;">Genet Civet</p>  <p style="text-align: right; font-size: small;">12-20-2016 17:53:55</p>
<p style="text-align: center;">Hog</p>  <p style="text-align: right; font-size: small;">12-21-2016 21:44:47</p>	<p style="text-align: center;">Leopard</p>  <p style="text-align: right; font-size: small;">03-28-2017 11:42:03</p>
<p style="text-align: center;">Prosimian Monkey</p>	<p style="text-align: center;">Rodent</p>

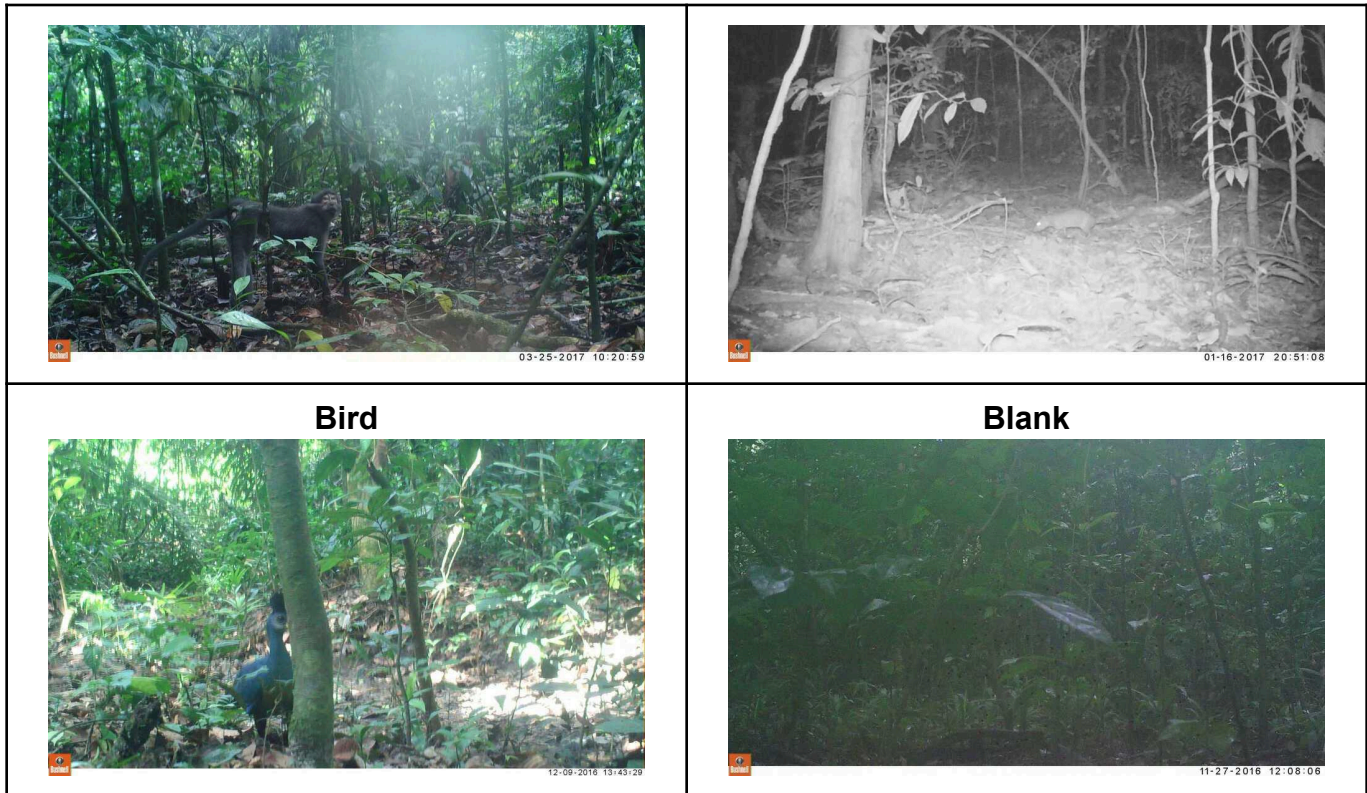


Table 1. Sample image of each class

Each class had 2,000+ images except for the hog class, which only had 979 images. This discrepancy indicates that the hog class is underrepresented compared to the others. This dataset presents several challenges while using lightweight models. First, the image isn't always centered and doesn't contain the full animal. Oftentimes it will only contain a limb such as a leg or a tail. Secondly, the animals aren't in the same spot in each image. Some images will have the animal in the center while others have it in a corner. There is also foliage and other disruptors that are also captured in the camera trap image. Finally, the lightweight models used in this study only operate with low-resolution images. Since the camera trap images are of a higher resolution, they need to be coarsened to fit the model's requirements, omitting information that the model may have been able to utilize.

Methods

The computer vision models used were the EfficientNet machine learning models. The EfficientNet architecture was first introduced by researchers at Google in 2019 and now consists of numerous iterations. The EfficientNet model is a computer vision model based on convolutional neural networks that allows for rapid training and quick results. A limitation of the model, however, is that it can only handle input images that are small, under 250 pixels by 250 pixels. For this project, only EfficientNet models with parameters under 250 pixels by 250 pixels were used and compared to ensure that the models had equal standing.

The primary model experiments we explored involved altering the EfficientNet hyperparameters. Hyperparameters are predefined settings in machine learning models, like learning rate, batch

size, and the number of epochs, that control the training process and significantly influence model performance. Tweaking hyperparameters is crucial in machine learning to optimize model performance and ensure it generalizes well to new data. Additionally, proper parameter tuning enhances training efficiency and stability, adapting the model to the specific characteristics of the dataset. By finding the optimal parameters, the model can function at full capacity and produce the best results.

The three hyperparameters that were experimented with in this project were the learning rate, the batch size, and the number of epochs. The learning rate is a hyperparameter that determines the size of the steps the model takes during optimization when updating the weights based on the gradient of the loss function. A properly set learning rate ensures that the model converges efficiently to a minimum of the loss function; if it's too high, the model might overshoot the minimum, and if it's too low, the training process can become excessively slow or get stuck in local minima. A higher learning rate would be expected to result in faster training time but less accurate identification. In contrast, a lower learning rate would be expected to result in a slower training time but more accurate identification.

The next main parameter experimented with in this project is batch size. Batch size is a hyperparameter in machine learning that determines the number of training images processed together before the model's internal parameters are updated. A smaller batch size provides more frequent updates and can lead to more precise adjustments, while a larger batch size results in faster computation per epoch and smoother gradient estimates but requires more memory.

The final parameter in the EfficientNet architecture manipulated here is the number of epochs. The number of epochs is a hyperparameter in machine learning that defines the number of complete passes through the entire training dataset during the training process. Each epoch allows the model to learn and update its parameters based on the entire dataset. A higher number of epochs would result in a much longer training time but, typically, a more accurate model. A lower number of epochs would result in a smaller training time but a less accurate model.

Model Validation

First, the model was run with a batch size of 4 and increasing learning rates while keeping the number of epochs constant. After the ideal learning rate was determined, that rate was kept constant while the batch size was multiplied by four in each trial. Using this process, the most effective learning rate and batch size were determined for the base model. Since the other EfficientNet models were iterations of this base model, the same optimal parameters were applied when conducting trials to determine the most effective model. Then, once the highest performing model was identified, the epoch number was drastically increased until the accuracy and loss displayed significantly diminished returns. This allowed for the best possible accuracy and the lowest possible loss from the identified model configuration.

To determine the most accurate model with the most optimal parameters, a base model is necessary. In this case, the `efficientnet_b0` model was used as the base model to determine the ideal parameter values. These values were then used in other experiments here with slight

model architecture variations, like the efficientnet_b1, efficientnetv2-b0, and efficientnetv2-b1 model architectures.

The model's performance was judged by determining the percentage of images identified correctly for each class and then averaging those values together. This effectively weights the model performance on each class evenly. Another way that the models' effectiveness was quantified was through log loss.

To evaluate the accuracy and loss from the model, the trained model was applied to a separate dataset that the model was not trained on. This dataset unseen during training, referred to as the validation set, consists of 3,297 images. leaving 13,190 images available for training the model. The model was not trained on this validation set and was only exposed to it when evaluating the final percentages, accuracy, loss, and log loss.

Results

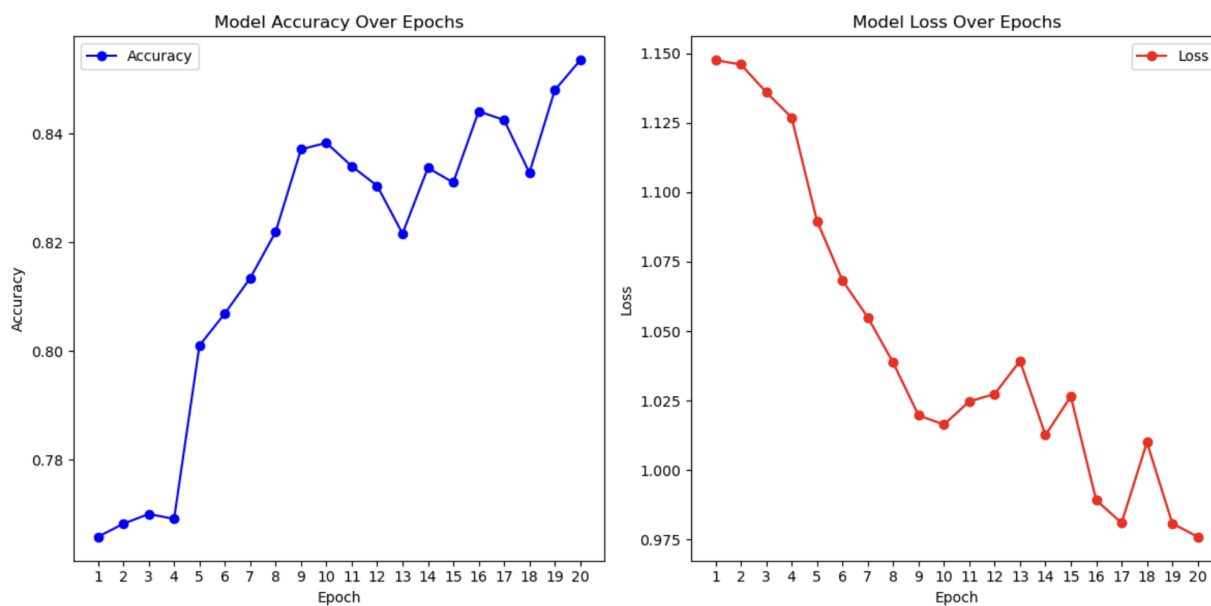


Fig. 1. These graphs represent how the accuracy and loss of the final model changed as the number of epochs increased. We can see that the accuracy is increasing and the loss is decreasing as the model trains.

The best-performing model was the efficientnet_b1 model which outperformed the efficientnet_b0, efficientnetv2-b0, and efficientnetv2-b1 models (Table 2). The hyperparameters leading to the highest performance were a learning rate of 0.005 and a batch size of 16. This means that the lowest learning rate and a moderate batch size led to the most accurate results. The models were all compared with 10 epochs, but once it was determined that efficientnet_b1 was the best model, it was run with 20 epochs to get the final percentage of 86.27%. It accepts up to images of 240 pixels by 240 pixels. Figure 1 shows that the accuracy and loss have not fully plateaued by epoch 20, suggesting that the model can be even more accurate at identifying the classes with additional training.

These results indicate that a lower learning rate is optimal for the most accurate identification. Although it takes longer to train the model, it overall leads to better results. These results also indicate that a batch size that balances both time to train and accuracy is the most effective.

Training the model with 10 epochs required 3-4 hours computation time. Training the model with 20 epochs required upwards of 8 hours. This was expected and is still a reasonable amount of time to train a model, as it can be done overnight.

Model	Learning Rate	Batch Size	Antelope	Duiker	Bird	Blank	Civet	Genet	Hog	Leopard	Monkey	Prosim	Rodent	Average	Epochs
efficientnet_b0	0.005	4	41.99%	79.88%	50.79%	90.22%	81.52%	94.76%	66.05%	75.06%	72.53%	10			
efficientnet_b0	0.05	4	2.23%	14.50%	35.73%	75.15%	35.87%	68.56%	74.74%	27.90%	41.84%	10			
efficientnet_b0	0.01	4	43.81%	70.71%	54.83%	87.17%	89.13%	86.03%	62.11%	74.32%	71.01%	10			
efficientnet_b0	0.005	16	65.11%	79.59%	60.45%	94.30%	87.50%	90.17%	72.05%	75.56%	78.09%	10			
efficientnet_b0	0.005	64	44.22%	63.61%	45.17%	88.59%	76.63%	83.19%	82.19%	61.73%	68.17%	10			
efficientnetv2-b0	0.005	16	72.41%	82.85%	48.09%	88.80%	90.76%	90.83%	67.91%	78.27%	77.49%	10			
efficientnetv2-b0	0.005	16	76.47%	71.89%	42.25%	86.56%	86.41%	84.50%	74.53%	67.90%	73.81%	10			
efficientnet_b1	0.005	16	68.97%	85.80%	49.66%	91.65%	88.59%	87.55%	74.53%	85.19%	78.99%	10			
efficientnetv2-b1	0.005	16	78.50%	78.70%	53.93%	93.28%	90.76%	94.32%	61.70%	74.07%	78.16%	10			
efficientnet_b1	0.005	16	72.41%	92.31%	66.07%	94.70%	94.57%	97.38%	85.30%	87.41%	86.27%	20			

Table 2. This table shows the experimentation process and the order in which the models and hyperparameters were tested. It also demonstrates how the final hyperparameters (learning rate of 0.005 and a batch size of 16) and model architecture (efficientnet_b1) were identified.

The model performed the best in the Leopard class with an accuracy of 97.38% on the held out validation set images. This is most likely because of its easily identifiable pattern of spots that can be seen on any part of the Leopard. This allows the model to easily identify the Leopard, regardless of how much of the Leopard is shown in the image and where in the image it is shown.

The model struggled with the Blank images the most. This could be because even though the image does not contain an animal in it, there are still various features and foliage that are in the image that the model could be overwhelmed by and assume that it is an animal.

Conclusions

We trained a series of EfficientNet models on a camera trap dataset to identify eight different species, evaluating how different hyperparameter values for learning rate and batch size influence performance. Our findings indicate that a lower learning rate with a moderate batch size is the most effective for accurate identification. The model identified leopards, hogs, genet civets, and birds well but struggled with the blank class.

We successfully trained the final model in under eight hours on a single laptop. This resulted in a model that can accurately identify over 90% of images in four species classes. This efficiency is impressive and suggests that with longer training, the performance could improve even further (Fig. 1). This emphasizes that computer vision is a powerful tool for conservationists to analyze massive datasets quickly and accurately. Our work demonstrates how tuning basic hyperparameters can lead to dramatic improvements in accuracy (Table 2). With the optimal values for learning rate and batch size, we significantly enhanced the model's performance.

Limitations and Future Work

Although the model performed well, it struggled with certain classes. The class it had the most difficulty with was the blank class. Future work could focus on improving classification accuracy for these challenging categories by incorporating more advanced data preprocessing techniques or exploring different model architectures.

Our results in Table 2 suggest that other models, besides our final one, have potential and may be worth experimenting with further. Additionally, exploring other hyperparameters such as dropout rates and weight decay could further enhance model performance.

One major limitation of the model is its potential variability in performance across different regions or camera sites. Although it works well for camera traps in the Tai National Park, its effectiveness in other environments is yet to be tested. Future studies could analyze the model's adaptability when it comes to different locations or environmental conditions.

In summary, our study demonstrates the vast potential that computer vision has in the wildlife conservation field. Its ability to process and classify large camera trap datasets efficiently and accurately makes it a significant asset to conservation work. However, poor selection of hyperparameter values during training can lead to drastically underperforming models (Table 2). Therefore by continuously refining hyperparameters and exploring new models, we can further enhance the accuracy and applicability of these tools in conservation efforts.

Acknowledgments

We thank Google for providing notebooks for training the model and augmenting the data available here:

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/hub/tutorials/tf2_image_retraining.ipynb

References

- [1] DrivenData. (n.d.). Conser-vision practice area: Image classification.
<https://www.drivendata.org/competitions/87/competition-image-classification-wildlife-conservation/page/409/>
- [2] Norouzzadeh, M. S., Morris, D., Beery, S., Joshi, N., Jojic, N., & Clune, J. (2020). A deep active learning system for species identification and counting in camera trap images. *Methods in Ecology and Evolution*, 12(1), 150–161.
<https://doi.org/10.1111/2041-210x.13504>
- [3] Willi, M., Pitman, R. T., Cardoso, A. W., Locke, C., Swanson, A., Boyer, A., Veldthuis, M., & Fortson, L. (2018). Identifying animal species in camera trap images using Deep Learning and Citizen Science. *Methods in Ecology and Evolution*, 10(1), 80–91.
<https://doi.org/10.1111/2041-210x.13099>