Classification of Tourist Photos with Gorillas and Other Animals to Detect Distancing Violations

Jingyu Wang[1], Jiarui Fan[1], Ritesh Kasamsetty[1], Krittika Shahani[1#] and Prasenjit Mitra[1#]

[1]The Pennsylvania State University
[#]Advisor

ABSTRACT

Many people insist on getting close to wildlife without realizing how harmful the consequences of their actions can be such as the death of the animals or even themselves. To decrease the number of such events, we aim to classify tourist photos of mountain gorillas, a critically endangered species, and other animals to detect violations derived from the distance maintained between the animals and the humans. We plan to implement object detection technology to locate the human and animal targets in the image and generate features. Then, image processing techniques are applied. We propose innovative models following multi-input logic which takes in image data and non-image features with ensemble model logic to perform the classification. We show that this model performs more accurately and has a relatively shorter run-time compared to other models when detecting violations in tourist photos with animals.

INTRODUCTION

According to Gorilla Fund International, “There are also very important rules regarding the distance to be maintained between gorillas and visitors, the number of visitors per group, and a strict one-hour limit to the visit”. Many national parks have a distance limit that visitors should maintain and some provide tourists with the opportunity to watch the wild animals closely. However, some people violate this limit since they believe closer interactions give them a better experience. Many of these places can’t maintain the minimum distance to keep both parties safe which can intervene in animals’ routine behaviors, their daily life, and sometimes even their health conditions such as panic attacks. Because of this, our focus is on gauging the distance between people and animals, especially gorillas, to detect any violation based on photos taken by tourists. All the images used contain both animals and people; in the case of multiple people, the closest person to the animal is used to determine distance. The project can be potentially implemented to help manage some safari parks such as issuing warnings to tourists that get too close. Furthermore, this project can be utilized to protect endangered species by catching and preventing close encounters with tourists.

RELATED WORK

The project is mainly composed of object detection, image processing, and classification models. To figure out the best possible approach for each step in the project, we have done literature reviews about pre-trained models for object detection and image processing methods. Although there have been a few projects that used both, we didn't encounter any that used them in the wildlife setting, a more environmental and social topic.

Pre-trained Models for Object Detection

The goal of object detection in the project is to first check whether the animal and human can be detected and then locate both in the collected images. We are not looking for extreme precision from object detection algorithms, since we just need to ensure the location and categories of interesting targets are accurate. To perform this object detection, we had to research different types of algorithms and came across many popular methods to perform object detection on images, such as EfficientDet, MobileNet, and Yolo.

EfficientDet -- A pre-trained model, EfficientDet-Lite2, achieved mAP (mean average precision, a commonly used metric to measure the performance of object detector) of 41.5 indicated in Tensorflow Hub on the COCO 2017 eval set. Based on the research, this model can be combined with OpenCV to make detections, recognize the category of the objects, provide scores for the certainty of detected objects, and draw boxes around them as well.

MobileNet -- This model was first proposed as an implementation method designed for mobile and embedded vision applications. After some research, we found out it can also be implemented with OpenCV to detect objects in images. However, the gap between the use cases and our project still exists, with no label categories corresponding to our animals of interest.

YOLOv3 -- YOLO (You only look once) is mostly known for real-time object detection and is common in streaming videos. YOLOv3 is a quite fast and accurate detector, achieving an mAP of 57.9% on the COCO test-dev.

There are also some innovative and newly published state-of-the-art models for object detection. For example, Soft Teacher + Swin-L, published in 2021, can achieve a high average precision (AP) value of 61.3 on the COCO test-dev (Xu, M., 2021). This model is a semi-supervised object detection algorithm, meaning that the training data contains labeled images and unlabeled images. During the training process, the teacher model and student model are both trained. The teacher model is trained on unlabeled images and generates pseudo boxes while the student model is trained on the labeled images and unlabeled images with those pseudo boxes. The student model keeps providing information for the teacher model to help improve the pseudo-label quality for the unlabeled images.

YOLOR (You Only Learn One Representation), also published in 2021, can also achieve a pretty high AP value of 57.3 on the COCO test-dev (Wang, Yeh & Liao, 2021). This is a method to serve different tasks all at once by combining explicit knowledge (learning based on given data and input) and implicit knowledge (learned subconsciously). The implicit knowledge is introduced to the network model which has been trained by explicit knowledge. Compared with other solutions, it is more similar to what humans can do like answering different questions based on the same input image.

A setback however is that the newly published methods, although proven to perform much better than the traditional methods, have few use cases similar to ours, making it harder to be a good reference for us.

Pre-trained Models for Image Processing

Nowadays, there are many state-of-art solutions for image processing, each widely used by researchers in different fields of study. Listed are the methods we used.

VGG-16 -- VGG stands for Visual Geometry Group, which is the group of researchers at the University of Oxford who designed this model. VGG-16 is the model with very deep convolution networks for large-scale image recognition, and 16 implies there are 16 layers in this model. VGG family also has other different models such as VGG-19 which contains 19 layers. VGG-16 is extensively used in many deep-learning image classification tasks. The strength of VGG is its ease of implementation and its good performance. However, the 138 billion parameters it utilizes made it a larger and slower model compared to other CNN models.

ResNet-50 -- ResNet stands for Residual Neural Network, and ResNet-50 is also a convolutional neural network with 50 layers. The ResNet model was the winner of the ImageNet challenge in 2015. The model aimed to tackle the issue similar to vanishing gradients that appear in gradient descent and is designed to avoid poor performance when the model becomes deeper. The core of ResNet is its “identity shortcut connection” which bypasses/skips one or more layers. The authors argue that it is easier to let the stacked layers fit a residual mapping and that the training error should not be higher in the deeper models. By making use of the shortcut connections, ResNet can achieve very high accuracy with about 25 million parameters, much less than VGG models. However, there are also some drawbacks to the model. One issue is the increased complexity of architecture and the complicated process of adding skip-level connections which are not easy to determine (Arjun, 2020).

InceptionV3 -- InceptionV3 was originally from GoogleNet/InceptionV1. The model contains a hierarchy of complex inception blocks which can reduce the parameter space. The authors (Szegedy et al., 2015) proposed the idea of “scaling up networks in ways that aim at utilizing the added computation as efficiently as possible by suitably factorized convolutions and aggressive regularization”. This means that despite being relatively deeper and more complex, InceptionV3 uses fewer parameters and lower computational complexity than other models.

However, according to research (Klemen, 2017), InceptionV3 has a poor performance on noise and image blur. DenseNet 201 -- DenseNet is based on the idea that convolutional networks can be substantially deeper, more accurate, and more efficient to train if they contain shorter connections between layers close to the input and close to the output (Gopani, 2020). Fewer parameters and high accuracy are achieved using the short and dense connections between layers. Concatenation is applied in the Dense block. When each layer receives “collective knowledge” or feature maps from all preceding layers, the network becomes thinner and more compact (Tsang, 2019). DenseNet can reduce the effect of the vanishing gradient problem and strengthen the feature propagation with a strong gradient flow that propagates error signals to earlier layers more directly.

Xception -- Introduced by Google, Xception, which stands for the Extreme version of Inception, introduces a modified depthwise separable convolution (SeparableConv) and is proven to be better than Inception v3 (on ImageNet datasets). The SeparableConvs are treated as Inception modules and placed throughout the Xception architecture in addition to residual connections from ResNet that are also added to the architecture which improves the performance significantly.

Distance between Objects in Photos

During the COVID-19 pandemic when social distancing was required, technology was implemented to detect the distance between people. Based on the paper Social Distancing Analyzer Using Computer Vision and Deep Learning, a starting point is to estimate the distance between people by performing pixel calculation between the centroids of the boxes which are used to indicate where each person is in photos or videos (G V Shalini et al, 2021).

METHODOLOGIES

Image Augmentation -- In any classification problem, the distribution of the training dataset is quite important. In other words, if we have two groups, neither one should completely outweigh the other. An even distribution for the dataset is required. In our case, the number of violation images far outweighed that of the non-violation images, because we require all the photos to contain both people and animals. Intuitively, people only take pictures of the animals when they are close to them, which leads to a biased distribution of the training dataset. Because of this, image augmentation is applied to generate more negative examples based on the non-violation images we have collected by rotating, vertical or horizontal moves, zooming in, or zooming out. This technique is commonly used in the image classification problem when there is a lack of data. We used a function called ImageDataGenerator which took parameters such as rotation range, height shift range, and width shift range to generate new images. We simply used this to help us generate sixty more images that we used to enhance our training dataset.

Object Detector using EfficientDet-Lite -- EfficientDet-Lite can detect most images correctly with the right category and moderately high certainty in our scenario. This detector can help us generate boxes that denote where the target objects including people and animals are. Also, labels can be produced to denote the categories of each object. For example, when detecting a human object, the label "person" can be produced. Similarly, when detecting animal objects, the label "animal" can be created. For the gap mentioned earlier, even though we cannot get an accurate label to denote mountain gorillas, the more general term "animal" can be used to serve the purpose of the project without obvious influence on this particular detector. A "checker" method is added to the object detection process to make sure the animal and human both can be detected in each image, especially for the augmented images. Because the augmentation can generate many images based on the original ones, it is not efficient to check one by one whether some changes like horizontal or vertical moves cut important targets from the original images. This method is similar to a filter: only the images which can satisfy the requirement, that both an animal and a person are detected, can be outputs of the detection process.

Individual Image Classification Models -- Our first approach to classifying images into two classes, violation and non-violation, that were used to pre-train individual image classification models. We utilized many individual models such as ResNet, VGG, Xception, Inception, and EfficientNet. The images fed to the models contained the boxes created by the object detector.

Feature Creation -- This idea is inspired because there can be some images containing more than one animal or one person. We focus on the pair of people and animals with the closest proximity to measure whether there is a distance violation or not in the image. We first think about the pixel distance between the centroids of the boxes that denote the closest pair of person and animal. Then, we consider adding additional non-image features to add more information to the model, such as the area of rectangles that surround the objects detected to indicate the object size and the area ratio of the boxes, which were saved in a tabular data format for model use. Following the multi-input model logic, those features can be trained simultaneously with the images in future work to generate more stable results.

Multi-input Model -- For the multi-input model, we use the features created during object detection and the images to train. The inputs of the model contain non-image features, images as well as labels where 1 denotes violation and 0 represents non-violation. Then, non-image features, first, are selected by a feature selection algorithm. Only the significant features which are most relevant to labels are saved to train. Then, a model is trained only with non- image features while another model is trained on just images. After that, the two models are concatenated to make predictions.

The model logic is shown by the framework below. Within the multi-input model, there are three main methods implemented:

1) Feature Selection (RFE). Instead of picking significant features randomly, we perform the selection by a more reasonable process. In Recursive Feature Elimination (RFE), there are mainly two important parameters we need to decide when implementing the algorithm: the number of features we would like to have and the algorithm used to choose the features. In the implementation, we choose a decision tree regressor to pick the features which are the most relevant to the response variable. For the number of features selected, we try different numbers to pursue the best performance. For example, in the gorilla case, we found after several trials that using 5 features was optimal. These 5 features were 'Closest Distance' (the distance between centroids of human and animal), 'Human Length' (the maximum human y-coordinate subtracted by the minimum), 'Human Area' (area of the generated human rectangle), 'Animal Area' (area of the generated animal rectangle), and 'midy_animals' (the midpoint of the minimum and maximum y- coordinates for the animal).
2) Multilayer perceptron (MLP) -- MLP is used to train on non-image features in the multi-input model we build. It is a fully connected neural network. The model, similar to other neural networks, contains an input layer, output layer, and hidden layer.
3) CNN with pre-trained InceptionV3 -- InceptionV3 is an advanced image recognition pre-trained model. It has achieved more than 78% accuracy on the ImageNet dataset which is commonly used in image processing and recognition projects. In our multi-input model, we incorporate this pre-trained model, starting from the pre-trained weights, into the CNN model to train on the image data.

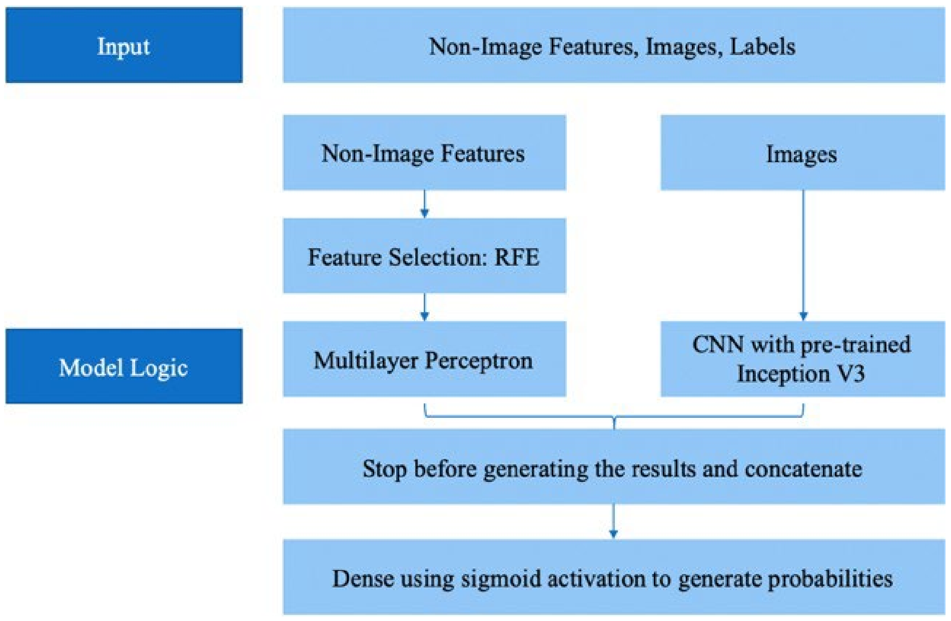

Input
Non-Image Features, Images, Labels
Non-Image Features
Images
Feature Selection: RFE
Model Logic
Multilayer Perceptron
CNN with pre-trained Inception V3
Stop before generating the results and concatenate
Dense using sigmoid activation to generate probabilities

Figure 1. Visualization of Multi-input Model Proposed

Ensemble Deep Learning Model -- Different convolutional neural network (CNN) models can provide different accuracy for image classification. An ensemble of multiple models can theoretically provide higher accuracy than individual models. The type of ensemble method we implement is the heterogeneous ensemble. This method is the combination of different types of classifiers and all the classifiers were built on the same data, and this method is suitable for small datasets like ours. The overall result of this ensemble classifier is carried out using all the results of each combined model. This ensemble classifier combined DenseNet 161, ResNet 50, ResNet 152, VGG 16, and

VGG 19_bn. The data is first trained in every single model separately. These five pre-trained models here are used as a starting point for our model, so we utilize the pre-trained weights to get a better model performance. After training the data in the five models, we save the best model weights for future use such as model reproduction. Then we define the ensemble classifier, train our data in this ensemble classifier, and save the model weights. The ensemble classifier takes in five inputs which are also the outputs of the five models, utilizes a linear layer to assign weights to the five models, and comes up with the final prediction. Theoretically, the ensemble model is supposed to have a better model performance than the individual models in this combination and be more stable as well. The results prove this hypothesis.

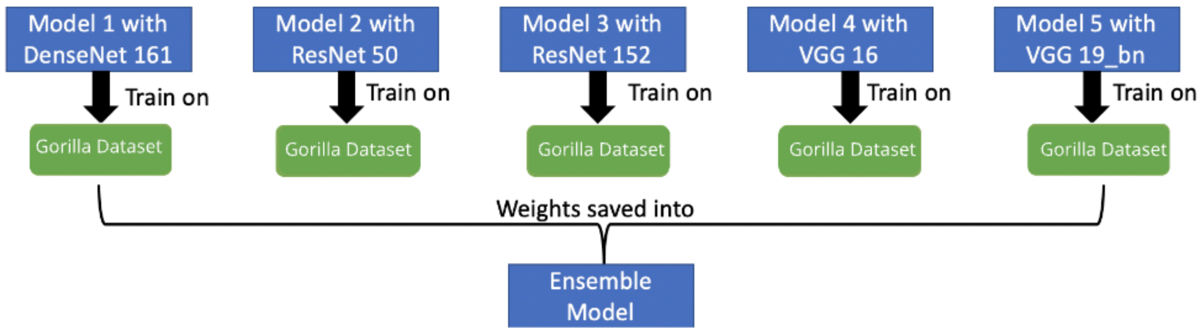


Figure 2. Visualization of Ensemble Model Proposed

EVALUATION

During the model implementation, model evaluation is always an important topic. There are several tools we use to quantify the model performance for our reference.

Confusion Matrix -- We generated a confusion matrix each time after testing a model. The confusion matrix contains four important values: true positive, true negative, false negative, and false positive. In our scenario, true positives represent images that are supposed to be in the violation group and are classified correctly. True negative is similar but for non-violation group classification. False-negative means when the images which should be regarded as violations, are classified to be in the non-violation group wrongly. False-positive is similar but for non-violation images that were categorized as a violation by the model.

Accuracy -- Accuracy is measured by the sum of true positive and true negative cases divided by the total number of testing data points. It provides a big picture of the model's performance. However, sometimes when the model performs well when detecting non-violations (true negatives) but much more poorly on classifying violations (true positives), the value of accuracy cannot show this message clearly. This is why accuracy is not enough by itself.

Recall -- Recall value can be calculated based on the confusion matrix. The reason why we care about the recall value is that we want to detect violation behaviors as much as we can to serve the goal. The recall value is calculated by the number of true positive cases divided by the sum of true positive cases and false-negative cases, which is quite consistent with our expectation for the model function.

Difference between training loss and validation loss -- Because the small size of our dataset puts some limitations on the model performance, we are concerned about the stability of the model. Hence, the difference between the training loss and validation loss is taken into account as a criterion to measure whether the model is stable and fits well in addition to the accuracy values of the two sets.

Evaluation of Individual Image Classification Models

The performance of many of these models was mediocre, but three stood out -- Xception, InceptionV3, and VGG. The models were trained with 200X200X3 images of gorillas and lions. We had to do quite a bit of hyperparameter tuning and juggling between optimizers to see which ones were the best fit. After we trained our model on training and validation datasets, we tested it on new data. We got the output in the form of a confusion matrix that showed us the accuracy, precision, recall, and F1 score.

Evaluation of Multi-input Model

The multi-input model performed quite well on gorilla images, mostly achieving 100% recall and accuracy. The potential reason might be that the overall data size is small, and the size of testing data is not large correspondingly: there are 11 non-violation images and 22 violation images in the testing data set. There might be some coincidences to get good results. However, based on the stress testing we have performed which is discussed in the following session, the model can perform quite well under different scenarios. There is some evidence to prove the outstanding performance of the model. Also, the difference between training and validation loss is reduced compared to the individual CNN with Inception V3 incorporated illustrating the effectiveness of the model.

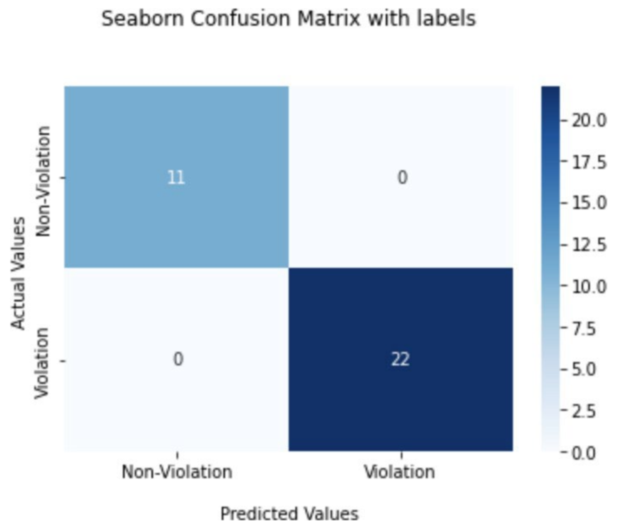


Figure 3. Confusion matrix of testing results for multi-input model on gorilla images

Evaluation of Ensemble Model

For the ensemble deep learning model, the performance on the gorilla dataset is also outstanding. Here, we have a total of 48 images as a testing dataset, including 24 violation pictures and 24 non-violation pictures. After the successful ensemble of five models (DenseNet 161, ResNet 50, ResNet 152, VGG 16, VGG 19_bn), this method has achieved 100% recall and 98% accuracy. Of the 48 pictures, only 1 image is predicted as a false positive. In addition, in the training process of the ensemble model, the training results also indicate that the ensemble model is more stable than the individual models. As the model trains, we see that the validation accuracy is consistently in the range of 0.94 to 0.99 with the validation loss also decreasing.

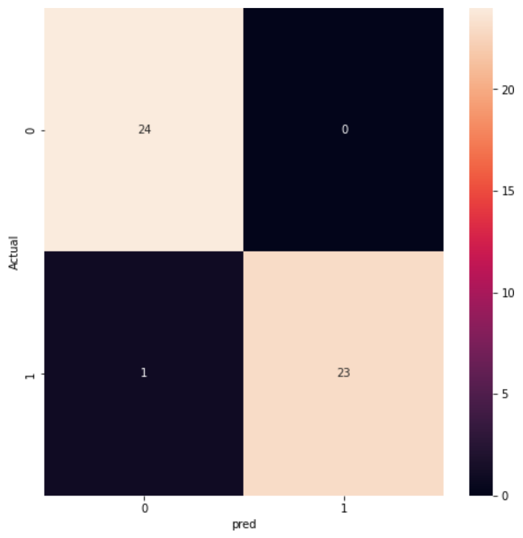


Figure 4. Confusion matrix of testing results for ensemble deep learning model on gorilla images.

DISCUSSION

As mentioned earlier, the model performs quite well on gorilla images. However, there are two main concerns. The first is the generalization of the model about whether the model can be implemented in different cases. The second is to check whether good performance is achieved by chance or not.

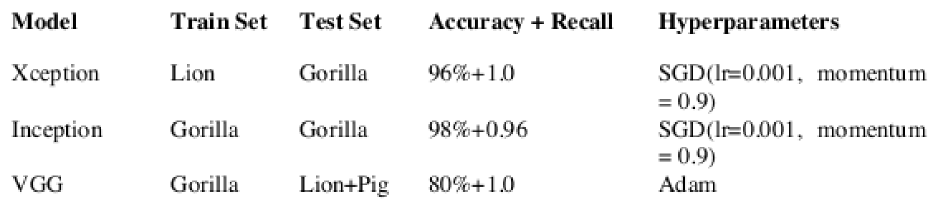
We could do stress testing in two ways: transfer learning and direct learning. In transfer learning, we still train on pure gorilla images, but several other images are added to testing data to see whether the model works well. Direct training, on the other hand, is when we provide the model with data on different animals to train with and then test the remaining images of gorillas and other animals to figure out the model's performance. If the model works well in one or more of the stress testing cases, we have evidence to believe the model can be generalized to some extent. If the model cannot work in some particular cases, we will have to further explore the reason.

The images used in the stress testing are in three categories. The first one is the photos we took around the Penn State campus: we use the lion and pig statues around the campus to set up a scenario by ourselves. In those photos, team members can be far away from statues and get close to the statues to produce non-violation and violation images. The second one is real animal photos: we scoured the internet for images of animals other than gorillas to do stress testing with. The third one is human distancing photos: because human distancing is a popular topic, we wondered whether our model can be implemented in this case or not. When the distance between two people is greater than the social distance, the image is regarded as non-violation and vice versa.

Stress Testing of Individual Image Classification Model

We stress-tested our model using mixed datasets that incorporated images of giraffes, elephants, and humans. However, our models didn't do that well on those images because we believe the features of these other animals were vastly different. However, some of the models that stood out are as follows in Table 1.

Table 1. Summary of applying transfer learning to test on other animal categories for individual models.

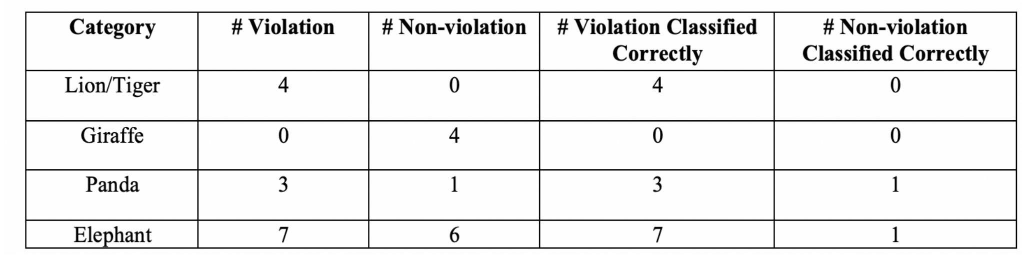


Model	Train Set	Test Set	Accuracy + Recall	Hyperparameters
Xception	Lion	Gorilla	96%+1.0	SGD(lr=0.001, momentum = 0.9)
Inception	Gorilla	Gorilla	98%+0.96	SGD(lr=0.001, momentum = 0.9)
VGG	Gorilla	Lion+Pig	80%+1.0	Adam

Stress Testing of Multi-input Model

Animal images through transfer learning -- For the multi-input model, animals like lions or tigers, giraffes, pandas, and elephants are involved in the stress testing. During the process, we discovered that the model detected violation or non-violation pretty accurately for animals such as lions, tigers, and pandas compared to animals like giraffes and elephants. We deduced that this was the reason because giraffes and elephants are much more different in size and body structure than the other animals. This discovery is quite reasonable because it is consistent with already known knowledge: when the implementation cases are too different, then transfer learning is likely less reliable.

Table 2. Summary of applying transfer learning to test on other animal categories for multi-input model.



Category	# Violation	# Non-violation	# Violation Classified Correctly	# Non-violation Classified Correctly
Lion/Tiger	4	0	4	0
Giraffe	0	4	0	0
Panda	3	1	3	1
Elephant	7	6	7	1

Lion and pig statue images by direct training -- To further test the generalization of the multi-input model, we implement it in a different case. Here, the data used is the statue images taken by us. Under this scenario, we train on the statue images directly and then test on similar images. The results show that the model can perform fairly well.

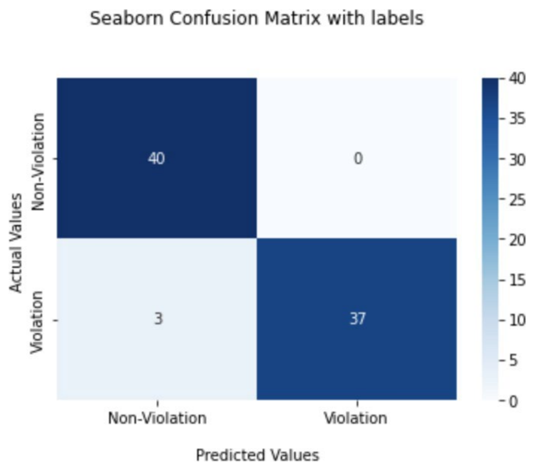


Figure 5. Confusion matrix of testing results for multi-input model on lion/pig statue images

Human distancing cases through transfer learning and direct training -- Generalization of the model seems like it could be proven by implementing lion and pig statue images. The best way to further test the model's generalization was through the human distancing case. During the experiment, the human distancing case could not be handled well by the classifier by only depending on the transfer learning. Hence, we put several human- distancing images into the training set and then predict several images. In the training dataset, there are 10 non- violation images and 10 violation images for human distancing mixed together with gorilla images. The same distribution is in the validation dataset as well. In the testing dataset, there are 8 images for each group. The result shows that the performance can be increased, not dramatically but at least noticeably, especially in the violation group, shown below: the first row contains the results for the human distancing case through transfer learning and the second row is by direct training.

Table 3. Comparison between using transfer learning and direct training.

Category	# Violation	# Non-violation	# Violation Classified Correctly	# Non-violation Classified Correctly
Human	7	6	4	5
Human	8	8	8	6

As training on human-distancing images directly increases the model performance compared to using transfer learning, we have reason to believe that if other animal images can be added to the training dataset, the model should work better even though the animal category can be quite different from the gorilla. Also, when the model is trained on even more diverse data, the model can potentially work on cases in a wide range related to distance-measure problems. We do realize, however, that to successfully come to this conclusion, we must increase the size of our dataset.

Stress Testing of Ensemble Model

Lion and pig statue images by direct training -- The Ensemble model is implemented on lion and pig statue images to test the model's generalization. With 618 photos for training, 142 photos for validation, and 60 for testing. The ensemble model has a training accuracy of around 0.9, and the validation accuracy is 0.8521. As for the testing, this model reached 0.9 on accuracy, recall, and precision. For 30 violations and 30 non-violations, it successfully detected 27 from each.

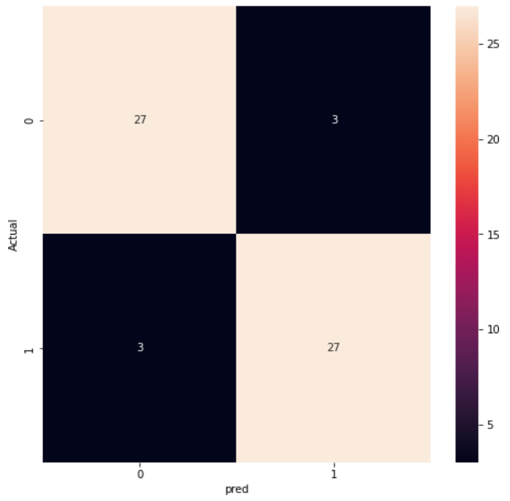


Figure 6. Confusion matrix of testing results for ensemble model on Lion and pig statue images

Other animal images -- Direct testing via transfer learning on other animal images did not perform well using the ensemble model. After training on the mixed data, a few testing data loaders were created to do the stress testing. However, the performance of the testing data on other animal images and human-human pictures did not perform well. Therefore, we thought of using another method following the framework shown below - to use our models trained previously on the mixed dataset, load its saved weights, train on the new dataset, and test to see its performance.

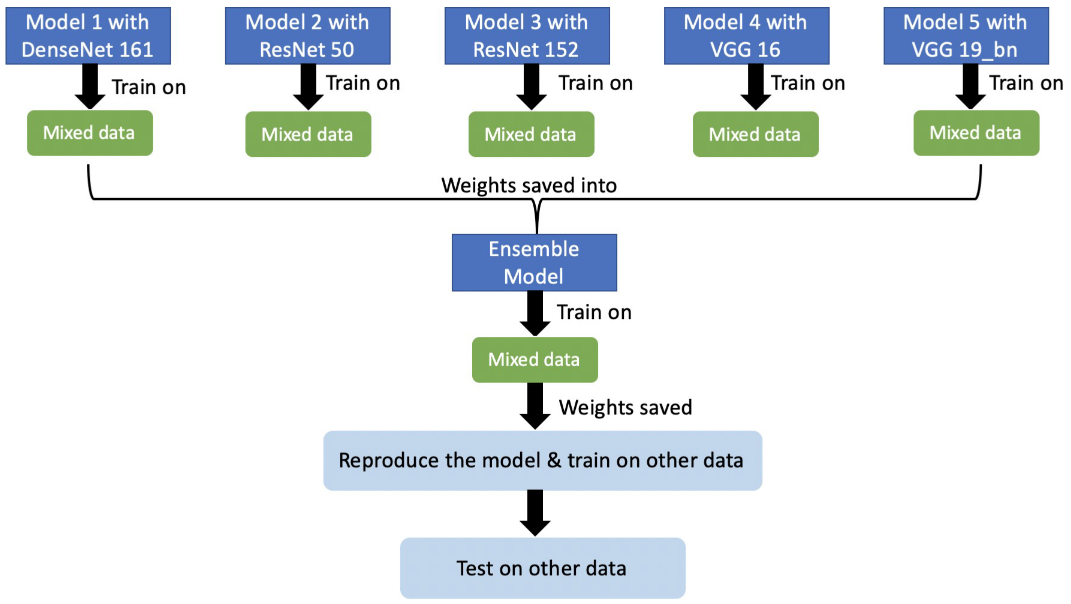


Figure 7. Ensemble model's stress-testing logic on other animals and human-distancing images

For the last two steps shown in the framework, the already built model is trained on other image data and tested on other image data. The training, validation, and testing dataset for giraffes (18, 15, 5), panthers (10, 8, 4), and human-human distancing (25, 10, 5) were manually split. As the results showed, the ensemble model does not perform extremely well on the giraffe data and the panther data, but the overall performance is not bad. They also show that the ensemble model might not perform well on animals that are very different from the gorilla, but when several images can be trained, the model performance is not too bad. The stress testing indicates the power of the model to be generalized to some extent. The model achieved an accuracy of 1 when training and testing on the human-human distancing photos and classified all 5 pictures correctly.

Table 4. Summary of stress testing of ensemble model on other image data

Category	# Violation	# Non-violation	# Violation Classified Correctly	# Non-violation Classified Correctly
Giraffe	2	3	2	2
Panther	3	1	2	1
Human Distancing	3	2	3	2

DISCUSSION

In this project, we have implemented several techniques to complete the tourist image classification task on mountain gorillas. After collecting images of gorillas, we implemented image augmentation techniques to make the training data evenly distributed. Then, object detection was used to locate the target objects in each image. After that, we built two innovative models following multi-input modeling and ensemble deep learning modeling logic based on individual models with pre-trained models nested. Both models had more stabilized performance, reduced overfitting, and enhanced predictions compared to the individual models. The models can also be generalized into other images related to other animal categories and human-distancing scenarios and not only in pure gorilla cases. Our models have been proven to function well to detect violation behaviors based on distance measures.

The multi-input classifier used the image data and the feature data we got from the images, trained separate models on two input categories, and combined the results in the last layer of the model to also result in an outstanding performance. The ensemble model takes 5 existing image processing models with different pre-trained models nested (DenseNet 161, ResNet 50, ResNet 152, VGG 16, and VGG 19_bn), combined the results from five models using a linear layer, and generated the final prediction. If we took run time into account, the multi-input classifier beats the ensemble model becoming the most efficient model overall.

FUTURE WORK RECOMMENDATION

Even though our models work well on the gorilla data and most of the other data we have collected, the performance is still concerned to be limited because of the lack of data. When we keep doing some stress testing by testing the model performance on other animals such as giraffes as well as human distancing pictures, the model can perform better on animals that are more similar to gorillas via transfer learning. With human distancing data added to the training dataset, the models can work better. Therefore, the generalization of our models needs to be further tested and improved. The things that could be improved are the size and diversity of the dataset. If more data points or more diverse data can be fed to train the model, the performance can be improved, and the results could be more reliable.

For the multi-input model, when the training data is changed, the number of features trained should be reconsidered and further decided. Feature selection is added mainly because the current dataset is small. If the dataset got larger with more images added, the feature selection part could be removed, and the workload can be saved to the neural networks to perform feature extraction directly.

For the ensemble deep learning model, choosing the best models to use in it is a way of improving the model's performance. Among the five models ensembled in this project, some of them can be given up or switched to another good-performing model. Also, choosing the best hyperparameters is crucial as this can influence the model performance a lot. In the project, only a few hyperparameters have been tuned and only two optimizers are tested, so there is still room for improvement by tuning them. The current ensemble model uses the same hyperparameters for all the single models that were ensembled, so choosing the best-suited hyperparameters for every single model can be also helpful to the future improvement of the model.

References

[1] Cassimiro, G. (2022, January 6). Object detection with Tensorflow model and OpenCV - Towards Data Science. Medium. https://towardsdatascience.com/object-detection-with-tensorflow-model-and-opencv-d839f3e42849
[2] TensorFlow Hub. (n.d.). TensorFlow Hub. https://tfhub.dev/tensorflow/efficientdet/lite4/detection/2
[3] A brief introduction to object detection: Yolov3, MobileNetv3, and EfficientDet: Imad El Hanafi — Portfolio &

Blog. (2020, May 5). A Brief Introduction to Object Detection: Yolov3, MobileNetv3, and EfficientDet. Retrieved April 4, 2022, from https://imadelhanafi.com/posts/object_detection_yolo_efficientdet_mobilenet/
[4] G. (2021). GitHub - gabrielcassimiro17/raspberry-pi-TensorFlow. GitHub. https://github.com/gabrielcassimiro17/raspberry-pi-tensorflow
[5] Brownlee, J. (2019, October 7). How to Perform Object Detection with YOLOv3 in Keras. Machine Learning Mastery. https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/
[6] Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.* 3 (Mar. 2003), 1289-1305.
[7] J. (2021b, August 26). Building an Object Detection Model from Scratch in Python. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/06/understanding-building-object-detection-model-python/
[8] Winastwan, R. (2021, December 22). How to Create a Simple Object Detection System with Python and ImageAI. Medium. https://towardsdatascience.com/how-to-create-a-simple-object-detection-system-with-python-and-imageai-ee1bcaf6b111
[9] Papers with Code - COCO test-dev Benchmark (Object Detection). (n.d.). Papers with Code - COCO Test-Dev Benchmark. https://paperswithcode.com/sota/object-detection-on-coco
[10] Xu, M. (2021, June 16). End-to-End Semi-Supervised Object Detection with Soft Teacher. arXiv.Org. https://arxiv.org/abs/2106.09018
[11] Papers with Code - You Only Learn One Representation: Unified Network for Multiple Tasks. (2021, May 10). You Only Learn One Representation: Unified Network for Multiple Tasks. https://paperswithcode.com/paper/you-only-learn-one-representation-unified
[12] Advanced Guide to Inception v3 | Cloud TPU |. (n.d.). Google Cloud. https://cloud.google.com/tpu/docs/inception-v3-advanced
[13] Shalini, G. V., Margret, M. K., Niraimathi, M. J. S., & Subashree, S. (2021). Social Distancing Analyzer Using Computer Vision and Deep Learning. Journal of Physics: Conference Series, 1916(1), 012039. https://doi.org/10.1088/1742-6596/1916/1/012039
[14] Y. (n.d.). GitHub - yasarniyazoglu/YOLOR ------Object-Detection-in-Custom-Data-with-YoloR: YoloR ile özel verilerde nesne tespiti. GitHub. https://github.com/yasarniyazoglu/YOLOR------- Object-Detection-in-Custom-Data-with-YoloR
[15] Rosebrock, A. (2021, July 9). Keras: Multiple Inputs and Mixed Data. PyImageSearch. https://pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/
[16] How to add non-image features alongside images as the input of CNNs. (2018, May 8). Data Science Stack Exchange. https://datascience.stackexchange.com/questions/31388/how-to-add-non-image-features-along-side-images-as-the-input-of-cnns
[17] Ye, A. (2021, December 16). Turning Non-Image Data into Images for Classification is Surprisingly Effective. Medium. https://towardsdatascience.com/turning-non-image-data-into-images-for-classification-is-surprisingly-effective-70ce82cfee27
[18] Pai, K. N. M. (2021, October 4). Combining Multiple Features and Multiple Outputs Using Keras Functional API. Paperspace Blog. https://blog.paperspace.com/combining-multiple-features-outputs-keras/
[19] Briggs, B. H. (2018, November 14). "Conservation successes" bring hope for mountain gorilla. BBC News. https://www.bbc.com/news/science-environment-46199100#:%7E:text=There%20were%20around%20600%20mountain,to%20gorillas%20in%20the%20wild
[20] Clark, J. (2022, March 3). Mountain Gorilla Tourism: Costs and Benefits. Dian Fossey. https://gorillafund.org/uncategorized/mountain-gorilla-tourism-costs-and-benefits/#:%7E:text=Gorilla%20tourism%20has%20since%20become,surveillance%20of%20additional%20gorilla%20groups.&text=With%20increased%20protection%20from%20poachers,habituated%20for%20research%20or%20tourism
[21] Huilgol, P., 2020. Top 4 Pre-Trained Models for Image Classification | With Python Code. [online] Analytics

Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/

[22] Cs231n.github.io. 2022. CS231n Convolutional Neural Networks for Visual Recognition. [online] Available at: https://cs231n.github.io/transfer-learning/

[23] Brownlee, J., 2019. A Gentle Introduction to Transfer Learning for Deep Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[24] Koehrsen, W., 2018. Transfer Learning with Convolutional Neural Networks in PyTorch. [online] Medium. Available at: https://towardsdatascience.com/transfer-learning-with-convolutional-neural-networks-in-pytorch-dd09190245ce#:~:text=The%20basic%20premise%20of%20transfer,layers%20which%20make%20a%20prediction.

[25] Majumdar, A., 2020. What are the disadvantages of using residual neural network? [online] Quora. Available at: https://www.quora.com/What-are-the-disadvantages-of-using-residual-neural-network

[26] Grm, K., Štruc, V., Artiges, A., Caron, M., & Ekenel, H.K. (2018). Strengths and weaknesses of deep learning models for face recognition against image degradations. IET Biom., 7, 81-89.

[27] Alex Krizhevsky, Ilya Sutskever, Geoffery E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NeurIPS, 2012.

[28] Karen Simonyan and Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv (2014), https://arxiv.org/abs/1409.1556.

[29] He et. al., Deep residual learning for image recognition, CVPR 2015.

[30] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2015. Rethinking the Inception Architecture for Computer Vision. [online] arXiv.org. Available at: https://arxiv.org/abs/1512.00567

[31] Tan, M., 2019. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling. [online] Google AI Blog. Available at: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

[32] Huang et. al., Densely Connected Convolutional Networks. CVPR 2017.

[33] A. Gopani, S. Krishna, S. Mukherjee, and S. K. Nair, “Benchmark analysis of popular image classification models,” Analytics India Magazine, 26-Apr-2020. [Online]. Available: https://analyticsindiamag.com/benchmark-analysis-of-popular-image-classification-models/

[34] D. Nair, “NASNet - A brief overview,” OpenGenus IQ: Computing Expertise & Legacy, 26-Dec-2020. [Online]. Available: https://iq.opengenus.org/nasnet/

[35] S.-H. Tsang, “Review: DenseNet - dense convolutional network (image classification),” Medium, 20-Mar-2019. [Online]. Available: https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

[36] S.-H. Tsang, “Review: Xception - with depthwise separable convolution, better than inception-V3 (image...,” Medium, 20-Mar-2019. [Online]. Available: https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568

[37] Brownlee, J. (2019, August 6). How to use Learning Curves to Diagnose Machine Learning Model Performance. Machine Learning Mastery. https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

[38] Brownlee, J. (2020, August 15). What is a Confusion Matrix in Machine Learning? Machine Learning Mastery. https://machinelearningmastery.com/confusion-matrix-machine-learning/#:%7E:text=A%20confusion%20matrix%20is%20a%20summary%20of%20prediction%20results%20on,key%20to%20the%20confusion%20matrix

[39] Fotache, C. (2018, December 11). How to Train an Image Classifier in PyTorch and use it to Perform Basic Inference on Single Images. Medium. https://towardsdatascience.com/how-to-train-an-image-classifier-in-pytorch-and-use-it-to-perform-basic-inference-on-single-images-99465a1e9bf5

[40] Fusia, C., Quiroa, L., Silva, S., Wolukanis, F. V., Boulabiar, I., Babu, S., Alexandrou, F., & Chedella, P. (2021, May 4). PyTorch for Beginners: Image Classification using Pre-trained models. LearnOpenCV – OpenCV, PyTorch, Keras, Tensorflow Examples and Tutorials. https://learnopencv.com/pytorch-for-beginners-image-

classification-using-pre-trained-models/
[41] G. (2018a, November 22). ResNet50 with PyTorch. Kaggle. https://www.kaggle.com/gxkok21/resnet50-with-pytorch
[42] G. (2020, March 5). Plotting accuracy scores and losses in ResNet. PyTorch Forums. https://discuss.pytorch.org/t/plotting-accuracy-scores-and-losses-in-resnet/72085
[43] Kharwal, A. (2021, July 7). Classification Report in Machine Learning. Data Science | Machine Learning | Python | C++ | Coding | Programming | JavaScript. https://thecleverprogrammer.com/2021/07/07/classification-report-in-machine-learning/
[44] Kumar, A. (2020, September 3). PyTorch - How to Load & Predict using Resnet Model. Data Analytics. https://vitalflux.com/pytorch-load-predict-pretrained-resnet-model/
[45] P. (2018b, October 3). Transfer learning with ResNet-50 in PyTorch. Kaggle. https://www.kaggle.com/pmigdal/transfer-learning-with-resnet-50-in-pytorch
[46] Singh, R. (2021, December 15). Dish Classification using ResNet50 Model with PyTorch. Medium. https://medium.com/@wularitz/image-classifier-for-dish-classification-using-resnet50-with-pytorch-5d11c02067b5
[47] Singhal, G. (2020, May 5). Transfer Learning with ResNet in PyTorch. Pluralsight. https://www.pluralsight.com/guides/introduction-to-resnet
[48] torch.stack — PyTorch 1.11.0 documentation. (2019). Pytorch. https://pytorch.org/docs/stable/generated/torch.stack.html
[49] Transfer Learning for Computer Vision Tutorial — PyTorch Tutorials 1.11.0+cu102 documentation. (2022). Pytorch. https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
[50] Rahman, A., & Tasnim, S. (2014). Ensemble Classifiers and Their Applications: A Review. ArXiv, abs/1404.4088.
[51] Chandrababu, A. (2022, January 4). Mixed Input Data in PyTorch: CNN + MLP - Arjun Chandrababu. Medium. https://medium.com/@iamarjunchandra/mixed-input-data-in-pytorch-cnn-mlp-8aeff336e8a3
[52] G. (2018a, October 30). Combining Trained Models in PyTorch. PyTorch Forums. https://discuss.pytorch.org/t/combining-trained-models-in-pytorch/28383
[53] I. (2019a, August 6). Ensemble Model Pytorch. Kaggle. https://www.kaggle.com/code/iamsdt/ensemble-model-pytorch/notebook
[54] J. (2018b, June 15). Multiple input model architecture. PyTorch Forums. https://discuss.pytorch.org/t/multiple-input-model-architecture/19754/2
[55] P, A. (2022, February 12). How to Train an Ensemble of Convolutional Neural Networks for Image Classification. Medium. https://medium.com/@alexppppp/how-to-train-an-ensemble-of-convolutional-neural-networks-for-image-classification-8fc69b087d3
[56] Pedamkar, P. (2021, April 30). Ensemble Methods in Machine Learning. EDUCBA. https://www.educba.com/ensemble-methods-in-machine-learning/
[57] R. (2018c, June 29). Concatenate layer output with additional input data. PyTorch Forums. https://discuss.pytorch.org/t/concatenate-layer-output-with-additional-input-data/20462
[58] R. (2019b, July 31). Custom Ensemble approach. PyTorch Forums. https://discuss.pytorch.org/t/custom-ensemble-approach/52024/8
[59] Rosenfelder, M. (2020, July 19). Multi-Input Deep Neural Networks with PyTorch-Lightning - Combine Image and Tabular Data. Python Tutorials for Machine Learning, Deep Learning and Data Visualization. https://rosenfelder.ai/multi-input-neural-network-pytorch/