Image Classifier for Jaguars and Leopards

Brian Gerkens, Nathan Wood, Quondarious Woody

University of North Florida
Intro to Python Programming
1 U N F Dr, Jacksonville, FL 32224

Abstract

This application is designed to assist individuals in distinguishing between jaguars and leopards using machine learning techniques. It is common for these two big cats to be confused due to their similar physical characteristics. The application takes two pre-loaded images of both a jaguar and leopard and utilizes deep learning algorithms to identify distinguishing features, such as the shape and size of their spots, head shape, and other characteristics. The application provides an accurate identification of the two different endangered species and can be used by conservationists, wildlife researchers, and general animal enthusiasts to improve their understanding of these magnificent animals.

Introduction

Jaguars and leopards are two of the most iconic big cats in the world. These species share many similarities in their physical characteristics, which can make it difficult to differentiate between them. However, accurate identification of jaguars and leopards is critical for conservation efforts, as both species face significant threats to their survival. To address this challenge, an application has been developed that uses machine learning to distinguish between jaguars and leopards based on their physical features. This application can benefit conservationists, wildlife researchers, and animal enthusiasts by providing quick and accurate identification of the species.

The motivation behind creating an application that uses machine learning to distinguish between jaguars and leopards is to both educate those unfamiliar with the physical differences between the two endangered species, and also aid in their conservation and preservation. Due to the fact that jaguars and leopards share similar physical characteristics, it can lead to confusion in their identification. This can be problematic for conservationists and wildlife researchers who need to accurately identify the species in order to understand their behaviors, habitats, and populations.

Related Work

While researching the topic of convolutional neural networks, we stumbled across a research paper called "CLASSIFICATION OF ANIMALS DATASET FROM THE KAGGLE ANIMALS (CHEETAH, HYENA, JAGUAR ANDTIGER) DATABASE USING NEURAL NETWORK-BASED TECHNIQUES AND DEEP LEARNING". This paper is about an image classifier application for 4 different types of animals that uses a deep learning algorithm to create a convolutional neural network (CNN) which trains a machine learning model for image recognition for these animals. The datasets used for this research were pulled from www.kaggle.com. The datasets are divided into two main classes:

1. Training
2. Validating and Testing

For training, there are 200 images for each type of animal, so 400 images total. For validating and testing, there are 100 images for each type of animal, so 200 images total. All images are the same size and the dataset contains no empty values. The CNN layers for this research experiment can be represented by the figure below:

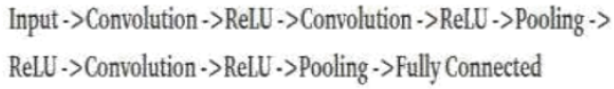

Figure 1: *Example breaking down layers of a convolutional neural netork.*

This research paper helped us gain some insight and an understanding of how to implement deep learning algorithms in order to created a CNN to help classify images between 2 different types of animals:

- Leopards
- Jaguars

Methodologies

ResNet50V2 Model

This project involves constructing a Convolutional Neural Network (CNN) using TensorFlow and Keras, but more specifically, a ResNet50V2 model. ResNet50V2 is a deep CNN architecture that is an extension of the original ResNet50 architecture. It is designed to enable training of very deep networks and has shown to outperform other CNN architectures in computer vision tasks. It uses

residual blocks, which enable the network to learn residual functions and handle the vanishing gradient problem in very deep networks. The architecture includes improvements over the original ResNet50 such as bottleneck blocks, improved weight initialization, and batch normalization before activation function in each residual block. ResNet50V2 is pre-trained on the ImageNet dataset and is often used for transfer learning in computer vision tasks such as image classification, object detection, and segmentation. We used the following imports to help construct our application:

- import pygame
- import cv2
- import os
- import random
- import tensorflow as tf
- import numpy as np
- from keras.models import load_model

Below is the implementation of our ResNet50V2 model and its corresponding layers:

```
res = tf.keras.applications.ResNet50V2(
    input_shape=(224,224,3),
    include_top=False,
)
res.trainable = False

res_model = Sequential()
res_model.add(res)
res_model.add(Dropout(0.25))
res_model.add(GlobalAveragePooling2D())
res_model.add(Flatten())
res_model.add(Dense(256, activation='relu'))
res_model.add(BatchNormalization())
res_model.add(Dropout(0.5))
res_model.add(Dense(10, activation='softmax'))
res_model.summary()
```

Figure 2: *Block of code representing the ResNet50V2 Convolutional Neural Network (CNN).*

For the first block of code (res), this creates an instance of the ResNet50V2 model provided by Keras, with the input shape of (224,224,3) and without the top layer. ResNet50V2 is a convolutional neural network architecture that contains residual blocks to address the vanishing gradient problem in deep neural networks. After that block, we have our neural network with different layers. The 'Sequential()' aspect of the network creates a sequential model object that will be used to build the complete model. The next line adds the ResNet50V2 model to the sequential model object. Then, adds a dropout layer to the model, which randomly drops out 25% of the inputs during training. This helps to prevent overfitting.

Next we add a global average pooling layer to the model, which calculates the average value for each feature map in the previous layer. This reduces the dimensionality of the data and helps to prevent overfitting. The next line adds a flatten layer to the model, which flattens the output of the previous layer into a 1D array. This is necessary for connecting the output to a fully connected layer. After that, the 'Dense()' layer adds a fully connected layer to the model with 256 units and a ReLU activation function. This layer learns high-level representations of the input data. In addition, another layer, called 'BatchNormalization()' adds a batch normalization layer to the model, which normalizes the input data to have zero mean and unit variance. This helps to improve the stability and performance of the model. Following this layer is another 'Dropout()' layer and then we end the neural network with another 'Dense()' layer.

Loss and Accuracy

```
73/73 [==============================] - ETA: 0s - loss: 0.2092 - accuracy: 0.9311
Epoch 8: val_loss did not improve from 0.09537
73/73 [==============================] - 120s 2s/step - loss: 0.2092 - accuracy: 0.9311 - val_loss: 0.1724 - val_accuracy: 0.9062
Epoch 8: early stopping
```

Figure 3: *Shows the loss and accuracy when we fit the model with training data.*

```
res_model.evaluate(testdata)
                                                                    Python
2/2 [==============================] - 3s 879ms/step - loss: 0.0508 - accuracy: 0.9800

[0.05084503814578056, 0.9800000190734863]
```

Figure 4: *Shows the loss and accuracy when we evaluate the model with testing data.*

Experiments and Results

Experiment

The application is a game that allows users to choose 1 of 2 images portrayed on the graphical user interface (GUI) below:

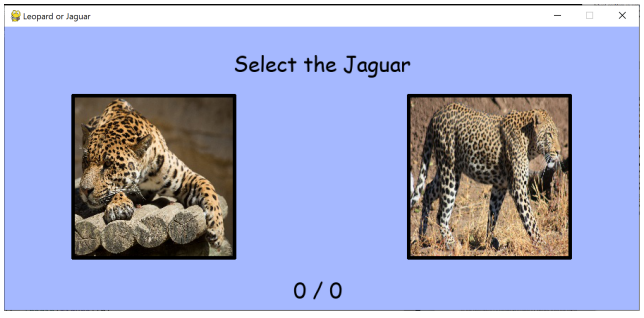

Figure 5: *Main display of the graphical user interface (GUI) for the application.*

The purpose of this game is to correctly identify a Jaguar from a Leopard, given that both have very subtle physical differences. For our experiment, each of us guessed ten times

per game for five games. Our goal is to see if we can gain a better understanding of differentiating the Jaguar from the Leopard based off of selecting the desired image. In addition to being a fun and engaging way to learn about jaguars and leopards, the game also provides a score counter that keeps track of how many correct choices the player makes. This score can be used to gauge the player's proficiency at distinguishing between the two animals, and can also provide a sense of accomplishment and motivation to continue playing and improving. Our prediction for this experiment is that when a user plays more and more games, their accuracy scores will increase in an upward trend.

Results

Our results were congruent with our original prediction that accuracy scores of playing this game will increase over time as the game is played more often. This upward trend of accuracy in scores is represented in a line graph shown in Figure 6:

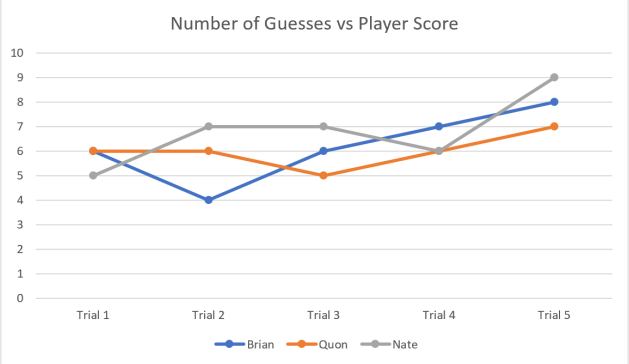

Figure 6: *Line graph data representing the number of guesses vs the player score for 5 different trials, showing the upward trend of improved accuracy scores as more games are played.*

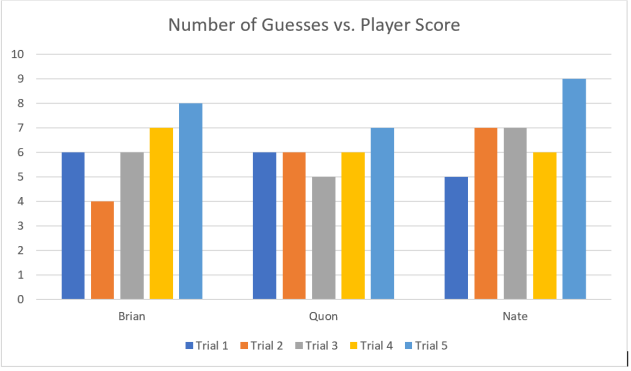

Figure 7: *Bar chart data representing the number of guesses vs the player score for 5 different trials.*

Both Figure 6 and Figure 7 represent the number of guesses per game vs the player scores for five different game attempts. In the line graph (Fig. 6), we notice every user that played the game 5 times improved on their accuracy scores over time, leading to this upward trend discussed before. Figure 7 is a clustered bar graph of the same data which also shows this increasing trend in accuracy scores. Figure 8 shows the average scores for each user, as well as the total average score among all users. The results are as follows:

AVERAGE SCORES OUT OF 10

- **Brian** - Avg. Score of 6.2
- **Quon** - Avg. Score of 6.0
- **Nate** - Avg. Score of 6.8
- **TOTAL** - Avg. Score of 6.333

We can see that Nate had the highest average score at 68% accuracy, Brian had the second highest average score at 62% accuracy, and Quon had the third highest average score at 60% accuracy. The total average score between Nate, Brian, and Quon was calculated at 63.3% accuracy, which is currently considered the average score for this application.

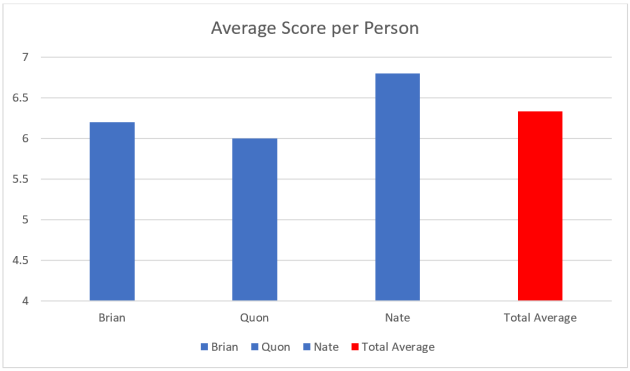

Figure 8: *Bar graph data showing the average scores of each player, as well as the total average score between all the players.*

Conclusions

In this experiment, we played a game that required us to distinguish between jaguars and leopards based on their physical characteristics. Each of us played the game ten times per session for five sessions, and our scores were recorded and analyzed to determine if we could gain a better understanding of how to differentiate between the two animals. Overall, our results showed that our ability to distinguish between jaguars and leopards improved over time, as reflected by our increasing scores. We also found that some of us were better at the task than others, suggesting that individual factors such as visual acuity and attention to detail may play a role in our ability to differentiate between the two species.

Additionally, we noticed that certain physical features of the jaguar and leopard, such as the shape of their spots and

the arrangement of their rosettes, were particularly helpful in making accurate identifications. These observations could be useful in future studies on how to teach people to distinguish between similar-looking animals. Overall, this experiment demonstrates the potential value of using games and other interactive tools to teach people about wildlife and improve their ability to identify and appreciate different species.

References

1. Ashraf, A. a. K. (2020). CLASSIFICATION OF ANIMALS DATASET USING IMAGE PROCESSING. ResearchGate. https://doi.org/10.13140/RG.2.2.35925.76009
2. 10 Big Cats of the Wild - Image Classification. (2023, February 28). Kaggle. https://www.kaggle.com/datasets/gpiosenka/cats-in-the-wild-image-classification
3. Eyadalqaysi. (2023). Big Cats Classification—— ResNet50V2 ——Acc: 100%. Kaggle. https://www.kaggle.com/code/eyadalqaysi/big-cats-classification-resnet50v2-acc-100
4. Team, K. (n.d.). Keras documentation: ResNet and ResNetV2. https://keras.io/api/applications/resnet/