Dogs and Cats Image Classification using Teachable Machine

Karunpat Promvisut
Chanatip Ampia
Tawan Muadmuenwai
Wongsakorn Saengsurasak

September 23, 2023

1 Introduction

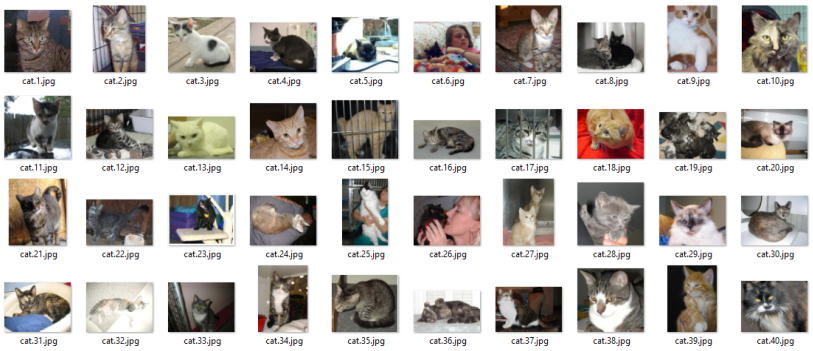
The objectives and goals of an assignment on "Dogs and Cats Image Classification" typically revolve around using machine learning and computer vision techniques to create a model that can accurately classify images of dogs and cats.

In this assignment, we utilized a dataset sourced from Kaggle, specifically from the following(`www.kaggle.com/datasets/chetankv/dogs-cats-images`). This dataset is abundant in images and is widely recognized as the "Dogs vs. Cats" dataset. The training dataset originally comprises 4,000 images of dogs and 4,000 images of cats, while the testing dataset consists of 2,000 images of dogs and cats.

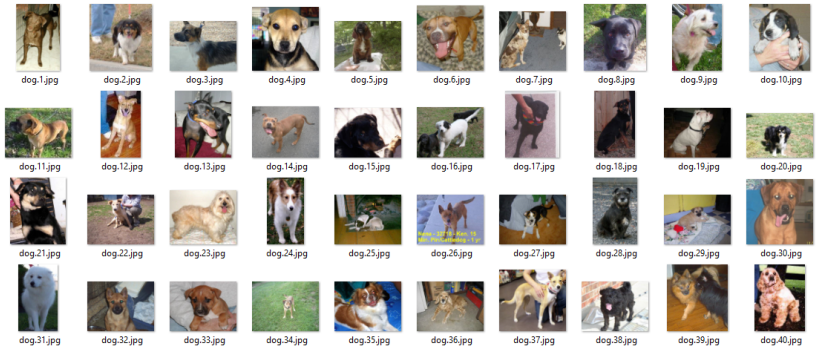
However, for this experiment, we use of a reduced subset of this dataset. We selected 1,000 images of dogs and 1,000 images of cats for training the model, and we use 200 images of dogs and cats for testing the model.

The examples of images in each class are shown as follows :

Class Cat:



Class Dog:



2 Model Building

We create and train machine learning models using Teachable Machine. It uses transfer learning, an ML technique, to find patterns and trends within the images samples, and create a simple and easy classification model within seconds as following step:

Step1: Gather and group your examples into classes, or categories, that you want the computer to learn.

Step2: Train your model, then instantly test it out to see whether it can correctly classify new examples.

Step3: Export your model for your projects: sites, apps, and more. You can download your model or host it online.

For model building, We have 3 models with varying parameters (epochs, batch size, and learning rate) to test with the images we prepared.

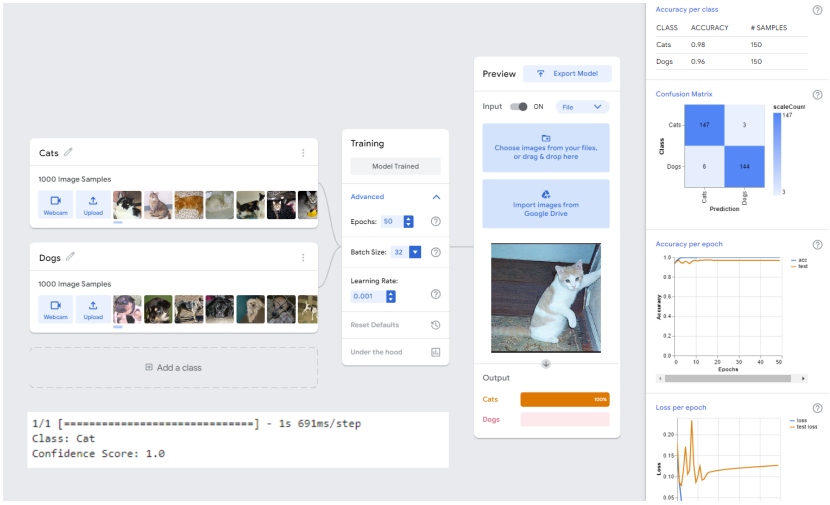
Model	Epochs	Batch Size	Learning Rate
Model 1	50	32	0.001
Model 2	100	64	0.01
Model 3	75	128	0.005

3 Results

3.1 Testing with Individual Images

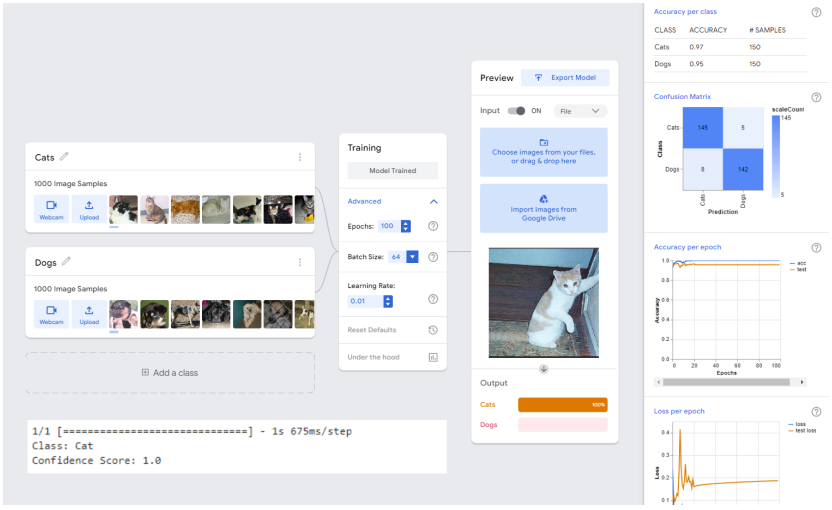
We use individual images from test set to test each of the 3 models. Then we upload images to Teachable Machine and get predictions from each model. For document the results, including the predicted class and confidence scores.

Model 1:



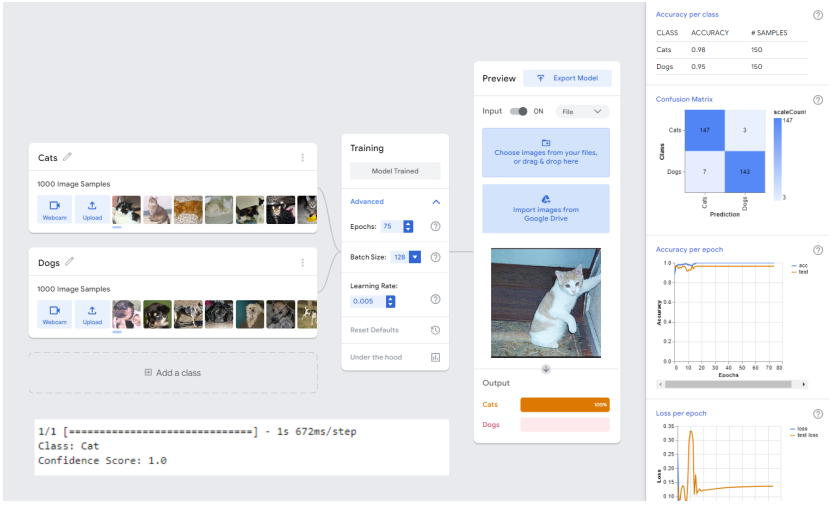
Class: Cat / Confidence Score: 1.0

Model 2:



Class: Cat / Confidence Score: 1.0

Model 3:

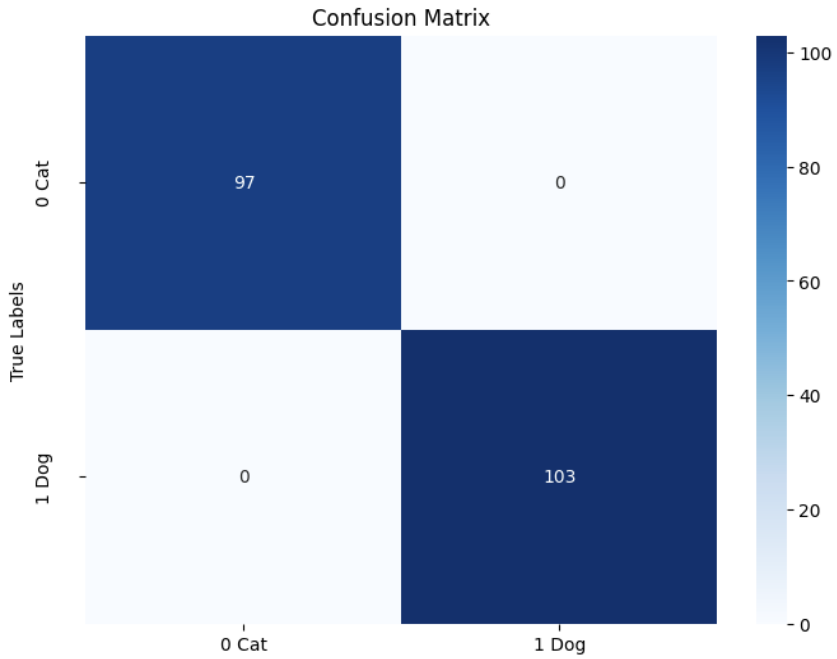


Class: Cat / Confidence Score: 1.0

3.2 Testing on Test Images and Evaluation

We use the entire test dataset (images not used during training) to evaluate each of the three models. Calculate and report the following metrics for each model. including the confusion matrix, accuracy, precision, recall, and F1-score.

Model 1:

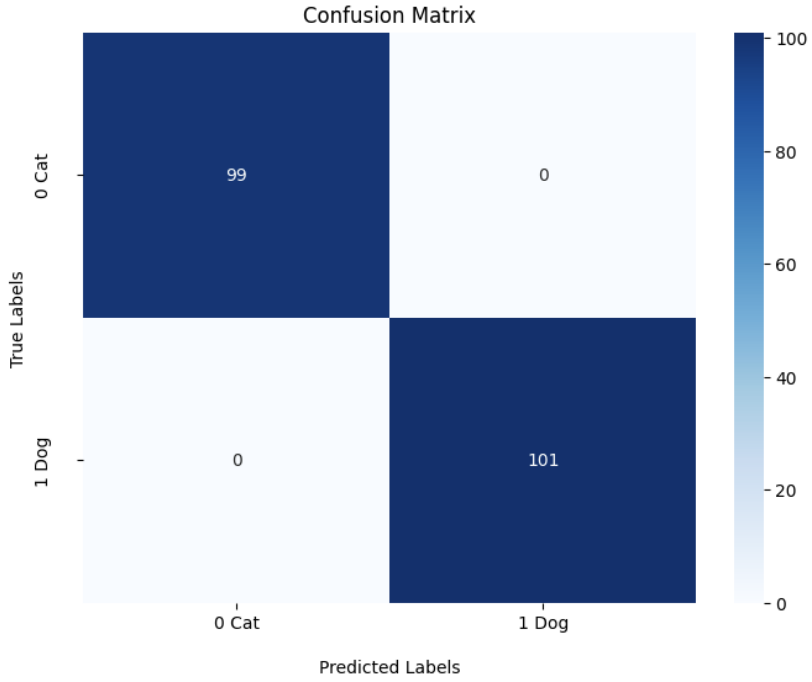



```
Classification Report:
              precision    recall  f1-score   support

      0 Cat
       1.00      1.00      1.00        97
      1 Dog
       1.00      1.00      1.00       103

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```

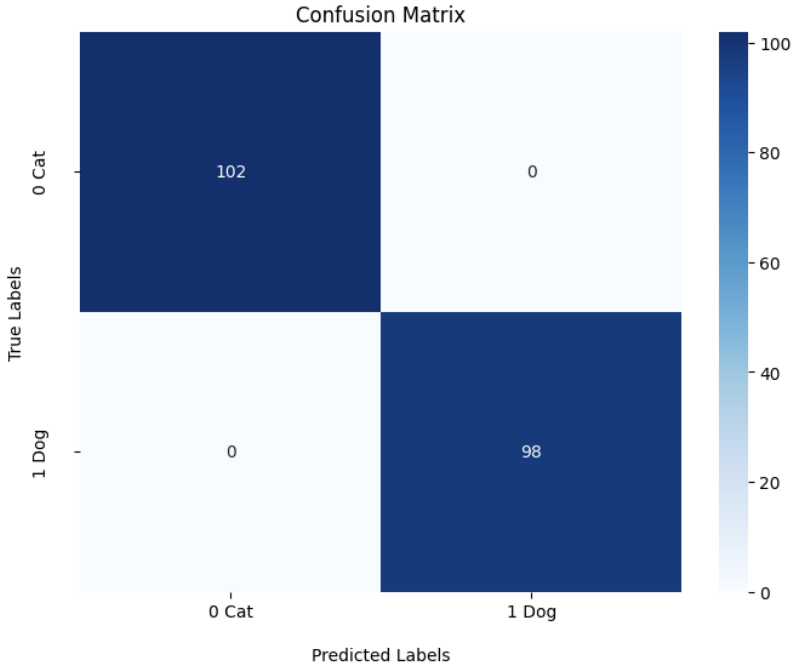
Model 2:



Classification Report:

	precision	recall	f1-score	support
0 Cat				
	1.00	1.00	1.00	99
1 Dog				
	1.00	1.00	1.00	101
accuracy			1.00	200
macro avg	1.00	1.00	1.00	200
weighted avg	1.00	1.00	1.00	200

Model 3:




```
Classification Report:
              precision    recall  f1-score   support

      0 Cat
       1.00      1.00      1.00       102
      1 Dog
       1.00      1.00      1.00        98

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```

4 Conclusion

The main finding from this dataset assignment is that the model achieved a remarkable 100% accuracy when classifying images as either a cat or a dog. This means that the model correctly identified every image in the test dataset, showing no misclassifications.

It's important to note that achieving 100% accuracy in image classification is an exceptional result and indicates a highly effective model. However, it's also important to consider potential factors that may have contributed to this result, such as the quality and diversity of the dataset, the architecture of the model used, and any preprocessing or augmentation techniques applied to the data.

In conclusion, the assignment of the "Dogs vs. Cats" dataset from Kaggle resulted in a model that demonstrated perfect accuracy in classifying images of cats and dogs, which is a notable achievement in the field of image classification.

5 Appendix: Code Samples

Include code snippets or code samples relevant to the assignment.

```
from keras.models import load_model
from PIL import Image, ImageOps
import numpy as np
import os
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix,
                                classification_report
import seaborn as sns

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model = load_model("keras_Model.h5", compile=False)

# Load the labels
class_names = open("labels.txt", "r").readlines()

# Define the folder path
folder_path = "dog_vs_cat/dataset/test_set_1/"

# List all image files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith
                                (".jpg")]

true_labels = []
predicted_labels = []
```

```
    for image_file in image_files:
        # Create the array of the right shape to feed into the
                                        keras model
        data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

        # Load the image
        image_path = os.path.join(folder_path, image_file)
        image = Image.open(image_path).convert("RGB")

        # Resize the image to be at least 224x224 and then crop
                                        from the center
        size = (224, 224)
        image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

        # Turn the image into a numpy array
        image_array = np.asarray(image)

        # Normalize the image
        normalized_image_array = (image_array.astype(np.float32) /
                                        127.5) - 1

        # Load the image into the array
        data[0] = normalized_image_array

        # Predict the model
        prediction = model.predict(data)
        index = np.argmax(prediction)
        class_name = class_names[index]
        confidence_score = prediction[0][index]

        # Print prediction and confidence score
        print("Image:", image_file)
        print("Class:", class_name[2:])
        print("Confidence Score:", confidence_score)
        print("----------------------------------------")

        true_labels.append(class_name[2:])
        predicted_labels.append(class_names[index][2:])
    # Create a confusion matrix
    confusion_mat = confusion_matrix(true_labels, predicted_labels)

    # Plot the confusion matrix
    plt.figure(figsize=(8, 6))
    sns.heatmap(confusion_mat, annot=True, fmt="d", cmap="Blues",
                                    xticklabels=class_names,
                                    yticklabels=class_names)
    plt.xlabel("Predicted Labels")
    plt.ylabel("True Labels")
    plt.title("Confusion Matrix")
    plt.show()

    # Generate a classification report
    classification_rep = classification_report(true_labels,
                                    predicted_labels,
                                    target_names=class_names)
    print("Classification Report:\n", classification_rep)
```