Dog Breed Classifier

Project Report

Will Muehlhausen 6/12/23

Introduction

This project focuses on developing a Convolutional Neural Network (CNN) pipeline to classify images of dogs and humans, identifying the dog breed in the case of dogs and suggesting resembling dog breeds in the case of humans. By utilizing CNNs for image classification, the project aims to provide accurate predictions and practical solutions for dog breed identification and resemblance suggestions. The project involves data collection, preprocessing, building the CNN architecture, training the model, and evaluating its performance.

Project Definition

Project Overview

The project aims to develop a Convolutional Neural Network (CNN) pipeline that can process images of dogs and humans and identify the corresponding dog breed or resembling dog breed, respectively. The project focuses on utilizing CNNs for image classification tasks.

Problem Domain

The problem domain of this project is image classification, specifically for dog breeds. The algorithm needs to be able to analyze input images and accurately identify the breed of the dog present in the image. Additionally, if a human image is provided, the algorithm should suggest a dog breed that the human resembles.

Project Origin

The project is likely an educational or learning project that focuses on implementing CNNs for image classification. It provides hands-on experience with building a complete pipeline for processing real-world images, training a CNN model, and making predictions based on the trained model.

Related Data Sets or Input Data

The project uses labeled data sets of dog images and human images. The dog image data set consists of images of various dog breeds, each labeled with the corresponding breed. The human image data set contains images of human faces. These data sets serve as the training and testing data for the CNN model, allowing it to learn patterns and features to make accurate predictions. They are provided by the Udacity course.

By building this pipeline and training a CNN model, the project aims to provide a practical solution for identifying dog breeds from images and suggesting resembling dog breeds for human images.

Problem Statement

Problem

The problem is to classify images of dogs and humans, identifying the dog breed in the case of dogs and suggesting the resembling dog breed in the case of humans.

Strategy for Solving the Problem

1. **Data Collection:** Gather a labeled dataset of dog images, including various breeds, and a dataset of human images. The datasets will serve as training and testing data for the CNN model.
2. **Preprocessing:** Preprocess the images to ensure they are in a suitable format for training the CNN model. This includes resizing the images, normalizing pixel values, and potentially applying other transformations like data augmentation.
3. **Build CNN Architecture:** Design and construct a CNN architecture suitable for image classification tasks. This typically involves stacking convolutional layers, pooling layers, and fully connected layers. The architecture should be able to learn relevant features from the input images.
4. **Train the Model:** Feed the preprocessed images into the CNN model and train it using appropriate optimization algorithms (e.g., stochastic gradient descent) and loss functions (e.g., categorical cross-entropy). The model will learn to classify dog breeds based on the labeled training data.
5. **Evaluate and Tune the Model:** Assess the performance of the trained model using the labeled test dataset. Measure the accuracy. If the model's performance is not satisfactory, fine-tune the architecture, adjust hyperparameters, or apply regularization techniques to improve performance.
6. **Prediction:** Once the model is trained and evaluated, use it to make predictions on new, unseen images. Given an image, the model should

identify the dog breed if it's a dog image or suggest the resembling dog breed if it's a human image.

Expected Solution

The expected solution is a pipeline deployed on a flask website that takes an input image as an input and applies the following steps:

1. Feed the preprocessed image into the trained CNN model.
2. Obtain the most likely dog breed.
3. If the image contains a human, suggest the resembling dog breed based on the predicted probabilities.
4. Return the identified dog breed or resembling dog breed as the output.
5. The solution is expected to provide accurate predictions for a wide range of dog breeds and offer meaningful suggestions when faced with human images resembling specific dog breeds.

Metrics

For this classification problem I will be using accuracy, recall, F1 score and a classification report that shows: precision as well. Here's a justification for using these metrics:

Accuracy:
Accuracy provides an overall measure of how well the model predicts the correct class labels. It calculates the ratio of correctly predicted samples to the total number of samples.

Recall:
Recall is especially important when the problem has a class imbalance or when correctly identifying positive samples is crucial. In our case the classes are not well balanced and recall helps evaluate the model's ability to correctly identify specific dog breeds.

F1 Score:
The F1 score captures the trade-off between precision and recall, making it a suitable metric when both are important.

By using accuracy, recall, and F1 score together, we can gain a comprehensive understanding of the model's performance from different perspectives. These metrics provide insights into the overall correctness, the ability to identify positive samples, and the balance between precision and recall. It is important to choose metrics that align with the problem characteristics and goals to ensure an accurate evaluation of the model's performance.

Implementation

In the project, metrics, algorithms, and techniques were implemented to evaluate the performance of the model. The following steps were taken:

1. **Dataset Loading:**

The project started with loading the train, test, and validation datasets using the load_dataset() function. This function used the load_files() function from the sklearn.datasets module to load the data and extract the file paths and target labels. The target labels were then converted to categorical using np_utils.to_categorical().

2. **Model Training and Testing:**

The ResNet-50 model was used for training and testing. The model was trained on the training dataset using the fit() function, and the validation dataset was used for validation during training. After training, the model was tested on the test dataset using the predict() function to obtain predictions.

3. **Performance Metrics Calculation:**

To evaluate the model's performance, various metrics were calculated. The accuracy score was computed using the formula accuracy_score(test_true_labels, test_predictions). Additionally, the recall score was calculated using

recall_score(test_true_labels, test_predictions, average='weighted'), and the F1 score was obtained using f1_score(test_true_labels, test_predictions, average='weighted').

Complications:

- Making the model hyperparameters fit the matrix of the model.
- Model Hyper parameters: I needed to select the best parameters to improve the model accuracy.
- Handling class imbalance: There was class imbalance as shown in the data visualization section of the notebook.

Refinement

Model Fit:

```
# Train the model
model.fit(train_Resnet50, train_targets,
          validation_data=(valid_Resnet50, valid_targets),
          epochs=10, batch_size=20, callbacks=[checkpoint], verbose=1)
```

```
Test Accuracy: 0.8182
Test Recall: 0.8182
Test F1 Score: 0.8178
```

Increasing epochs to 20 added a half a percentage point to the accuracy as shown:

```
# Train the model
model.fit(train_Resnet50, train_targets,
          validation_data=(valid_Resnet50, valid_targets),
          epochs=20, batch_size=20, callbacks=[checkpoint], verbose=1)
```

```
Test Accuracy: 0.8230
Test Recall: 0.8230
Test F1 Score: 0.8198
```

Hyper parameter:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_8 ( (None, 2048)              0
_________________________________________________________________
batch_normalization_1 (Batch (None, 2048)              8192
_________________________________________________________________
dense_6 (Dense)              (None, 133)               272517
=================================================================
Total params: 280,709
Trainable params: 276,613
Non-trainable params: 4,096
_________________________________________________________________
```

```
Test Accuracy: 0.7847
Test Recall: 0.7847
Test F1 Score: 0.7799
```

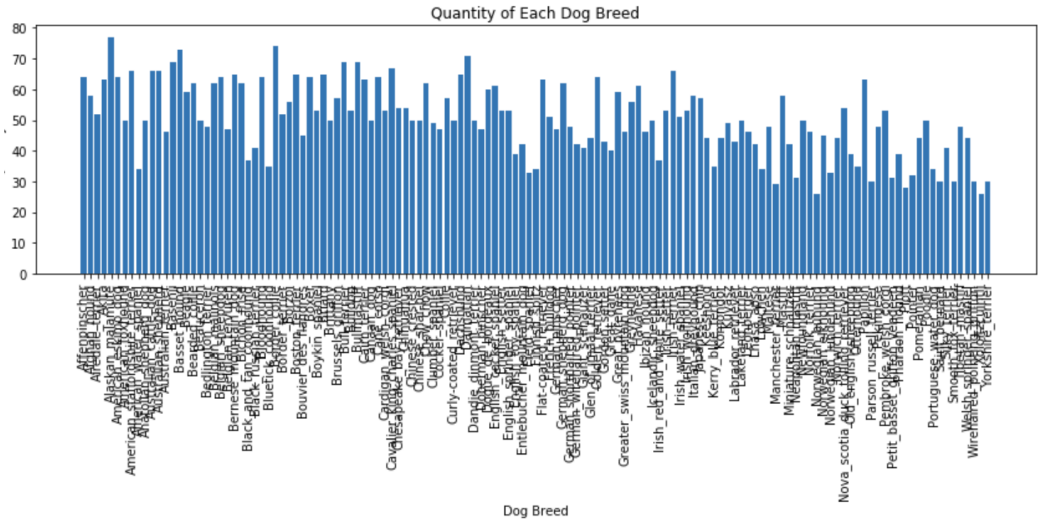
Remove batch normalization

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_5 ( (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 133)               272517
=================================================================
Total params: 272,517
Trainable params: 272,517
Non-trainable params: 0
_________________________________________________________________
```

```
Test Accuracy: 0.8230
Test Recall: 0.8230
Test F1 Score: 0.8198
```

Results

Dataset



The dataset used for training the model exhibited class imbalance, with some dog breeds having more instances than others. Class imbalance can impact the model's

performance, as it may be biased towards predicting the majority classes while struggling with minority classes. However, despite the class imbalance in the dataset, the model performed well and achieved high recall. This indicates that the model was able to handle the class imbalance effectively, demonstrating its robustness and ability to generalize across different dog breeds. The high accuracy achieved across the breeds suggests that the model learned meaningful patterns and features that are not heavily influenced by the class distribution. Hence, it can be concluded that the model's performance was not significantly affected by the class imbalance in the dataset.

Model Results

```
Test Accuracy: 0.8146
Test Recall: 0.8146
Test F1 Score: 0.8108
```

Classification Report:

	Precision	Recall	f1-score
Avg / total	0.84	0.81	0.81

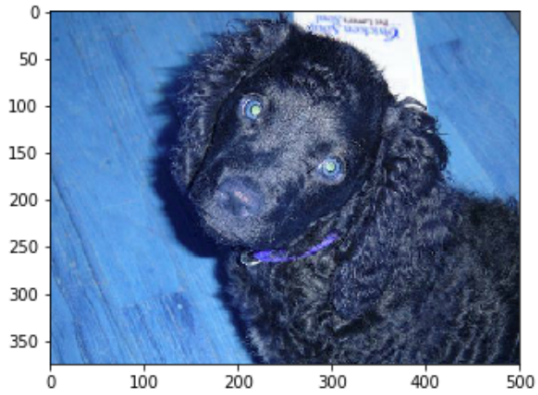
*Please refer to the notebook for a more detailed classification report.

*The results obtained from the model evaluation on the test dataset will vary per each run.

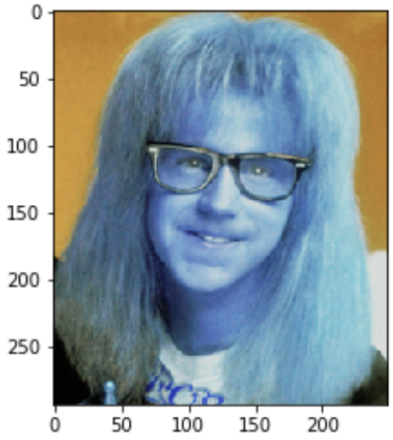
Overall, these results suggest that the model performed well in classifying the dog breeds, with relatively high accuracy, recall, and F1 score. It indicates that the model has learned meaningful patterns and features to discriminate between different dog breeds, leading to reliable predictions. However, the classification report did show that there were some underrepresented dog breeds that had a lower accuracy. This is due to the nature of the appearance of the dog and class imbalance.

Below are some examples of the model being tested.

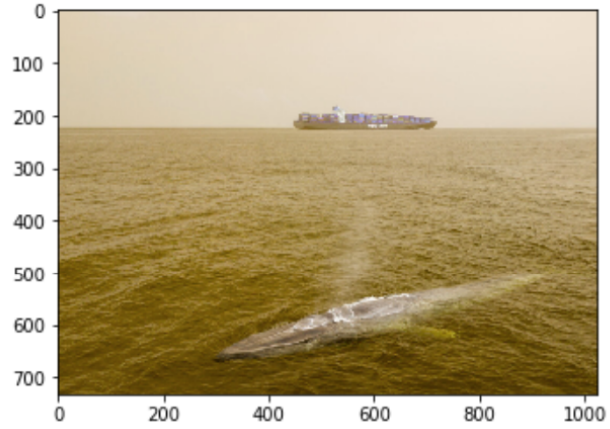
```
This dog is an Boykin_spaniel
```




```
This is a human, but if they were a dog they would be a Pekingese
```

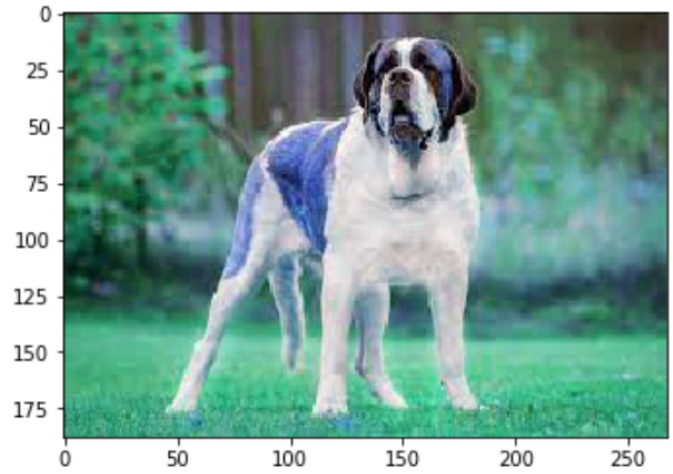



```
Human nor Dog not detected.
```

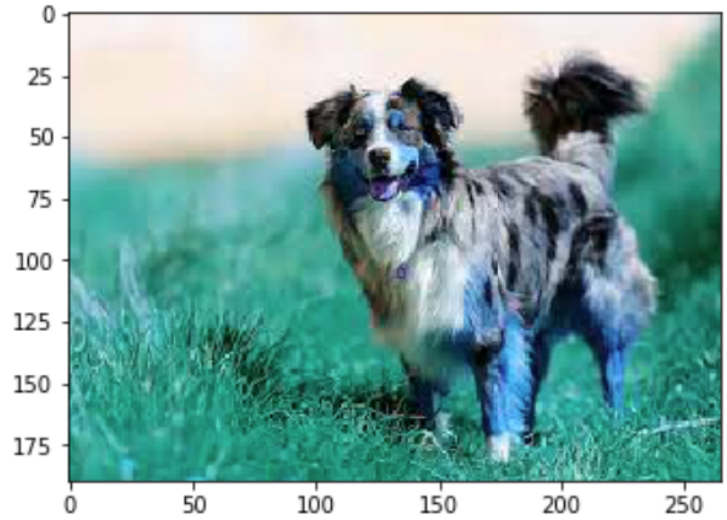


This is a human, but if they were a dog they would be a Chihuahua

This dog is an Saint_bernard



This dog is an Australian_shepherd



Overall it works very well. There are some instances where the model will predict different breeds for a human photo. This is because a human may resemble a few different dog breeds.

Reflection

The end-to-end problem solution involved developing a Convolutional Neural Network (CNN) pipeline for classifying dog breeds and suggesting resembling breeds for human photos. The solution included steps such as data collection, preprocessing, building the CNN architecture, training the model, evaluating its performance, and making predictions.

One particularly interesting aspect of the project was the substantial improvement in accuracy achieved through transfer learning. Transfer learning allowed leveraging the pre-trained ResNet-50 model, which was already trained on a large dataset. By utilizing the learned features and weights from this model as a starting point, the project achieved significant performance enhancements. Witnessing the accuracy increase from 4% to 40% and finally to 81% showcases the effectiveness of transfer learning in leveraging existing knowledge and enabling faster and more accurate training.

On the other hand, one of the challenging aspects of the project was correctly adding CNN layers to improve the model. Designing an effective CNN architecture requires a deep understanding of the underlying principles, including the choice and placement of convolutional layers, pooling layers, and fully connected layers. The difficulty lies in finding the right balance between model complexity and generalizability. Experimenting with different layer configurations, considering the input image size, and exploring various activation functions and regularization techniques can be a complex task.

Improvement

I am sure there is a lot of improvement to be made. Here are two that I think would be effective

1. Data Augmentation: This improvement focuses on enhancing the diversity and quantity of the training data through transformations. It may require additional computational resources and careful selection of augmentation techniques to ensure that the transformed samples remain representative of real-world variations.
2. Fine-tuning the CNN Architecture: This improvement involves fine-tuning the pre-trained CNN architecture specific to the dog breed classification task. Fine-tuning requires careful consideration of which layers to unfreeze and retrain, as well as optimizing the learning rate and training schedule. It may require more computational resources and extensive experimentation to find the optimal configuration.

Conclusion

In conclusion, the developed CNN pipeline for dog breed classification and resemblance suggestions has achieved promising results. The model demonstrates the ability to accurately classify dog breeds and provide meaningful suggestions for human images resembling specific dog breeds. The overall accuracy, recall, and F1 score indicate that the model has effectively learned patterns and features for discriminating between different dog breeds. Despite the class imbalance in the dataset, the model performs well, demonstrating its robustness and generalization capabilities. The project provides a comprehensive solution for image classification tasks using CNNs and showcases the potential of deep learning techniques in solving real-world problems.