CNN: Write an Algorithm for a Dog Identification App

Jianming Han

1 Definition

1.1 Project Overview

Image classification is the task of taking an input image and outputting a class (dog, car, etc.) or the probability of a class that best describes the image. It faces multiple challenges, including scale variation, viewpoint variation, intra-class variation, image deformation, image occlusion, illumination conditions and background clutter [1].

The dog breed identification problem is to build a model to infer the dog breed by supplying an image, it's a classic image classification problem which can be found on Kaggle [2], totally 1282 teams have competed on this problem and shared their solutions. In Literature, some researchers also studied the problem with different solutions. Jiongxin Liu [3] proposed a part localization approach to make dog breed classification and a recognition rate of 67% was achieved. Kaitlyn Mulligan [4] studied this problem by applying a Xception CNN architecture.

The dataset used can be downloaded from human dataset and dog dataset. In this project, the dataset is provided by Udacity within the project workspace.

1.2 Problem Statement

The goal of this project is to build a pipeline that can be used for dog breed identification. The model will take any real-world, user-supplied images as input and make predictions accordingly. Specifically, two main functionalities come with the model:

- Given an image of a dog, the model will identify an estimate of the canine's breed;
- Given an image of a human, the model will identify the resembling dog breed.

To achieve the goal, two key questions need to be solved: 1) Whether a human or a dog is on the picture, so a human detector and a dog detector are needed to distinguish between human and dog; 2) Which dog breed is on the image, so a multi-classification model is needed to infer dog breeds. Basically, the pipeline consists of three parts: human face detection, dog detection and dog breed inference. The image below displays potential sample output of the project.

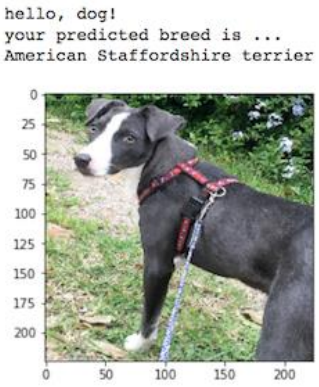

Fig.1 Sample output of the project

1.3 Metrics

Since the problem stated here is a multi-classification problem, the evaluation metrics I used are accuracy evaluation metric and categorical_crossentropy cost function.

- Accuracy is used to evaluate how many images are correctly inferred by the model;
- Categorical_crossentropy cost function punishes the classifier if the prediction is different with the actual label and leads to a higher accuracy.

The metrics will help measure the performance of the model, the ideal classifier has zero loss and 100% accuracy.

2 Analysis

2.1 Data Exploration

Totally we get 13233 and 8351 images for human and dogs respectively. The human images are sorted by names. All human images are in a standard size of 250*250, most images contain only one human face, but some contain multiple human faces.

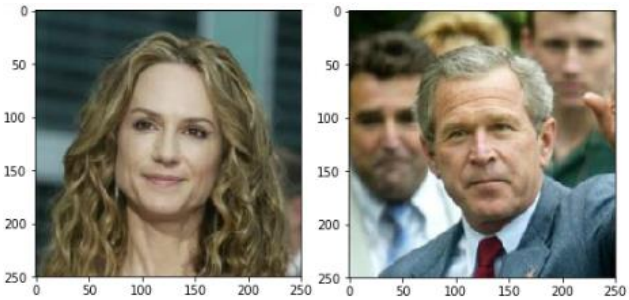

Fig.2 Examples of single human face and multiple human faces

The dog images are classified into 33 dog categories and split into train (6680 images), validation (835 images) and test (836 images) subsets. The average image quantity for each breed is about 63 (≈ 8351/133). And some breeds contain more and some contain less, it's slightly imbalanced among all the breeds. Dog images are with different image sizes, resolutions and lighting conditions, some contain complete dog body while some just contain a dog head. Further, some images contain multiple dogs and even human.

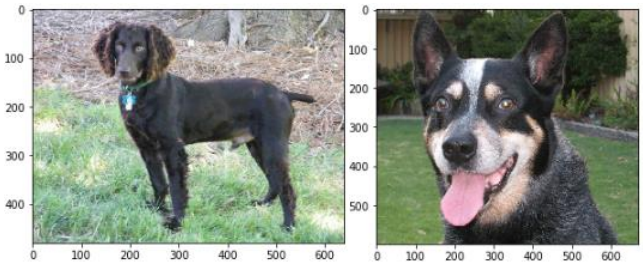

Fig.3 Examples of complete dog body and dog head

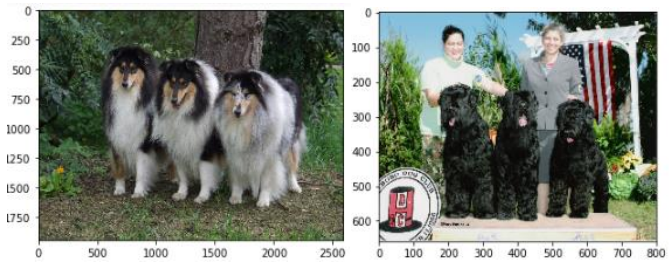

Fig.4 Examples of multiple dogs and dog with human

2.2 Algorithms and Techniques

The pipeline built contains three major parts: human face detection, dog detection and dog breed inference. The algorithms and techniques used in each part is discussed as following:

- OpenCV model will be used to detect human face in an image because OpenCV provides many pre-trained face detectors.
- To detect dog in an image, a pre-trained VGG16 model will be applied. Given an image, this VGG16 model will return a prediction for the object that is contained in the image.
- Lastly, to efficiently and accurately conduct the dog breed classification, convolutional neural network (CNN) will be applied to construct the model. In deep learning, CNN is a class of deep neural networks, most commonly applied to analyzing visual imagery [5]. It's one of the most popular techniques used in improving the accuracy of image classification.

Below is a brief summary of the project design:

1. Import all the images of dog and human, do some statistics and check some samples;

2. Detect Humans by OpenCV face detector;
3. Detect Dogs by applying a pre-trained VGG16 model;
4. Create a CNN to Classify Dog Breeds (from Scratch);
5. Create a CNN to Classify Dog Breeds (using Transfer Learning);
6. Write an algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither;
7. Test the algorithm on sample images and my own images.

2.3 Benchmark

I searched several models for dog breed identification, and the model from this article [6] shows that the created model from scratch is with an accuracy of 12%, the Xception model that utilizes transfer learning can reach an accuracy of 86% and a loss of 0.4723 on test data. With this benchmark model I can have something fairly enough to compare and crosscheck.

3 Methodology

3.1 Data Preprocessing

Data preprocessing is an important step in machine learning algorithms, it includes cleaning, instance selection, normalization, transformation, feature extraction and selection, etc. In CNN architecture, it's a standard step to transform the image to a fixed size, alongside normalization and conversion to a tensor datatype. Also, we add augmentation to training data to avoid overfitting. To summarize, the data preprocessing for the dog breed algorithm is done as following:

- All the images are resized to 224*224 as input because it's the size that the pre-trained VGG16 model expects;
- Normalization is conducted by using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225];
- Augmentation is applied to the training dataset to avoid overfitting. The images were augmented by randomly flipping and randomly rotation with 10 degrees;
- Data is transformed to tensor which is suitable for supplying to a CNN architecture.

3.2 Implementation

3.2.1 Human Face Detection

OpenCV provides many pre-trained face detectors, stored as XML files on github. I chose to use the Haar feature-based cascade classifiers to detect human faces in images. The cascade function can return the

human faces detected in an image so this procedure can be used as the human face detector that returns True if a human face is detected in an image and False otherwise.

Before using the face detectors, it is standard procedure to convert the images to grayscale. The detectMultiScale function executes the classifier stored in face_cascade and takes the grayscale image as a parameter. The structure of human face detector function is shown as below:

1. Obtain face detector;
2. Load color (RGB) image;
3. Convert image to grayscale;
4. Use pre-trained face detector to detect human face;
5. Return True if a human face is detected and False otherwise.

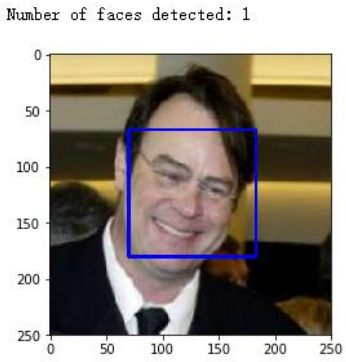

Fig.5 Sample output of face detector

To test the performance of the face detector function, I run it on the first 100 sample images for both human and dog dataset. The detector performs well on human images with an accuracy of 98%. But 17% of the dog images are mistaken for human.

3.2.2 Dog Detection

I used a pre-trained VGG-16 model to detect dogs in images. The VGG-16 model takes an image as input and returns the index corresponding to the ImageNet class. The output should always be an integer between 0 and 999 (inclusive). And in order to check if an image is predicted to contain a dog by the pre-trained VGG-16 model, we need only check if the pre-trained model predicts an index between 151 and 268 (inclusive).

To use this model, it's necessary to pre-process the data by applying resizing, normalization and converting to tensor. The structure of dog detector function is shown as below:

1. Obtain VGG16 model;
2. Load and pre-process the image;
3. Send an image to the VGG16 model;
4. Return True if predicted value is between 151 and 268 and False otherwise.

The dog detector performs well in the test with 100% accuracy on dog images and 0~1% error rate for human images.

3.2.3 Dog Breed Classification

I constructed a 3-layer CNN architecture with ReLu activation function from scratch in PyTorch to classify dog breeds. The description of the architecture is as following:

1. Construct 3 convolutional layers to detect regional patterns in images with depth 16, 32 and 64 respectively, using filters 3x3 and stride of 1, and add padding equals 1 to make sure the convolutional layer have the same width and height as its input from previous layer;
2. A ReLu activation function is applied to the output of the filters to standardize the output values;
3. Add Max pooling layers with filters 2x2 and stride of 2 after convolutional layers to reduce the dimensionality of input arrays half of what they were from previous layer and discard some spatial information;
4. Flatten the eventual output of convolutional and pooling layers so that all parameters can be seen as a vector by a linear classification layer;
5. Add two fully connected layers after the flatten output to determine the dog breed contained in the image;
6. Use dropout rate of 25% to prevent overfitting.

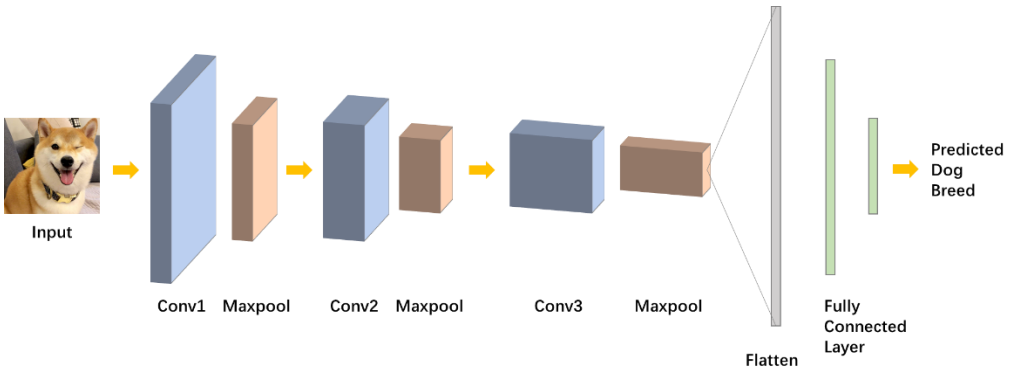

Fig.6 Dog breed classification CNN architecture from scratch

To train the model, the batch size is defined to 20 and the optimizer chosen is the SGD with learning rate 0.03. The model is trained with 50 epochs.

3.3 Refinement

Training a CNN classifier made from scratch on a small data like this will lead to underfitting with so many layers, and parameter tuning often causes overfitting. So, it is time to consider better solutions to create a CNN classifier.

Pre-trained models like ResNet have been trained on the very large ImageNet database which has 1000 object classes. The training data is so large and the models have so many parameters that models like these often takes weeks even on multiple GPUs. Transfer learning is a way that we could use what these models have already learned, and apply that knowledge to a new task of our own. In this project, instead of constructing a CNN from scratch, we can take the knowledge from a trained CNN like ResNet and use it to help classify dog breeds.

I chose to use the pre-trained ResNet152 model as fixed feature extractors that identify general features in the images, and add one fully-connected layer at the end to act as a final classifier for dog breed. The model is trained with 30 epochs and is fine-tuned by minimizing the cross-entropy loss function using SGD optimizer with learning rate of 0.03.

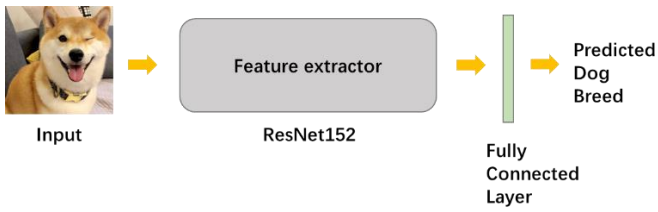

Fig.7 Dog breed classification CNN architecture using transfer learning

4 Results

4.1 CNN Models

CNN from scratch: The model constructed from scratch is not performing well with an accuracy of 18% (156/836) on the test dataset. Comparing to the benchmark model with 12% accuracy on a CNN model from scratch, it's an acceptable result. I think the augmentation step in data preprocessing plays an important role in this process that helps improve the accuracy.

CNN using transfer learning: The model using transfer learning improve the accuracy significantly from 18% to 86% (727/836), which is the same result as the benchmark Xception model that utilizes transfer learning. We can see that the transfer learning technique is very powerful in improving the model accuracy even the dataset is quite small.

4.2 My Own Algorithm

After the dog breed classification model is trained and ready for use, I write an algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither. Then,

- If a dog is detected in the image, return the predicted breed.
- If a human is detected in the image, return the resembling dog breed.

- If neither is detected in the image, provide output that indicates an error.

Then I test my algorithm with three different image groups: (1) 6 images chosen from the initial input dataset; (2) 10 example images provided in the images folder; (3) 15 images that I selected myself. Overall, the model performs well to give correct predictions for the three cases mentioned above, yet there is some interesting output. Some details are discussed for each image group.

Group 1: All 6 images are correctly detected as human or dogs, human are predicted with resembling dog breed and dogs are matched with the correct breeds.

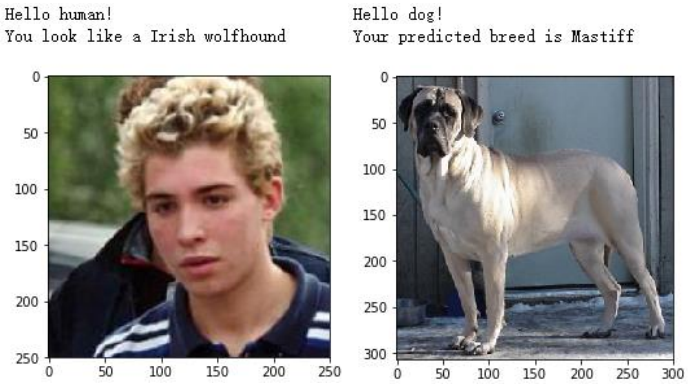

Fig.8 Sample output from Group 1

Group 2: The prediction result is also very well in this group. We can see that the algorithm can make correct predictions between resembling breeds, like Brittany and Irish red and white setter, Boykin spaniel and Curly-coated retriever. It's sometimes difficult for human to distinguish between them.

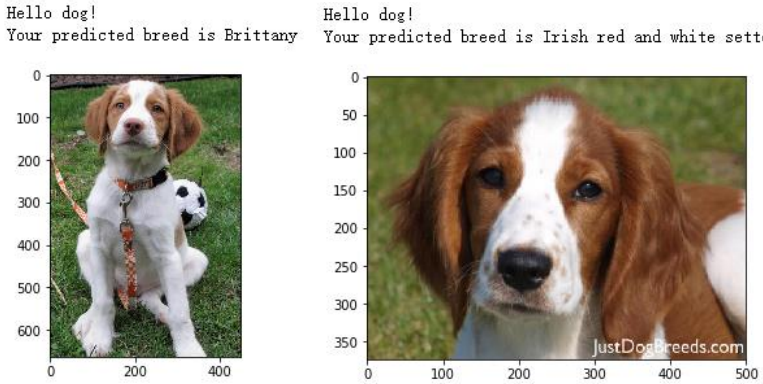

Fig.9 Predictions of Brittany and Irish red and white setter

Group 3: For the images I selected myself, most images are correctly predicted, meanwhile something interesting happens that some non-human images are detected as human faces, which means our human face detector performs not well sometimes and needs to be improved.

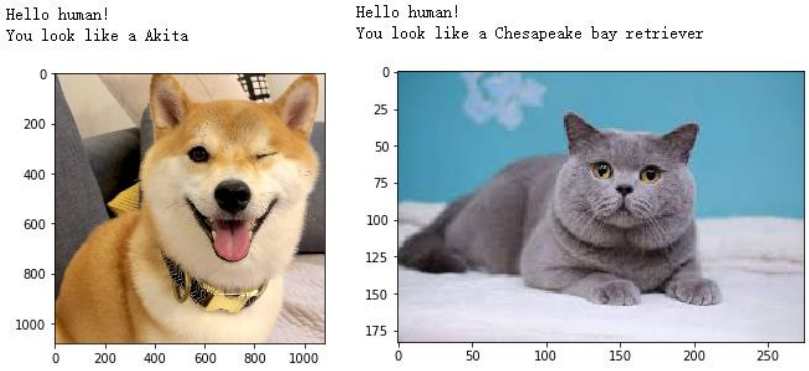

Fig.10 Examples of non-human images detected as human faces

5 Conclusions

In this project, I built a pipeline, which contains a human face detector, a dog detector and a CNN architecture to infer dog breed, for the dog breed identification problem. The techniques applied are summarized as following:

- Human face detector: Open CV's haar cascade classifier;
- Dog detector: Pre-trained VGG-16 model;
- Dog breed classifier: CNN architecture from scratch and CNN architecture using transfer learning with pre-training ResNet152 model.

The classifier using transfer learning results in a significant improvement with an accuracy of 86%, which is equivalent to the performance of the benchmark model. It's an acceptable result considering that the dataset is small.

Finally, an algorithm was written to take an image as input and make predictions accordingly. Three different image groups are tested on the algorithm, the result shows that the algorithm has quite good performance, but there are a few cases that non-human images are detected as human faces.

Still, the model is not perfect enough, there are several ways I can try in the future to improve the performance of the model:

1. Collect more training data for each dog breed;
2. Hyperparameter tuning such as learning rate, epochs, etc.;
3. Choose a better human face detector (17% of first 100 dog images have a detected human face).

It's an interesting and challenging project which helps me a lot in understanding image classification problem, CNN architecture and transfer learning. I will keep doing the research in this domain.

Reference

[1] Convolutional Neural Networks for Image Classification https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-networks-image-classification/.

[2] Kaggle Dog Breed Identification https://www.kaggle.com/c/dog-breed-identification/overview/description.

[3] Liu J, Kanazawa A. Dog Breed Classification Using Part Localization. European Conference on Computer Vision 2012; 172-185.

[4] Mulligan K, Rivas P. Dog Breed Identification with a Neural Network over Learned Representations from the Xception CNN Architecture. 21st International Conference on Artificial Intelligence 2019.

[5] Valueva M.V, Nagornov N.N, Lyakhov P.A, Valuev G.V, Chervyakov N.I. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation 2020; 177: 232–243.

[6] Deep Learning: How to build a dog detector and breed classifier using CNN?! https://towardsdatascience.com/deep-learning-build-a-dog-detector-and-breed-classifier-using-cnn-f6ea2e5d954a