Machine Learning Engineer Nanodegree

Capstone Project

Kiki Huang
09/18/2020

1. Definition

Project Overview

This project is one of Udacity's specific projects for completing Machine Learning Engineer nanodegree. In this project, a pipeline to process supplied images to identify the breed of the dog in the image using Convolutional Neural Network is built. The code will also identify the resembling dog breed if a human is detected.

Problem Statement

Overall, the project is to develop an algorithm for a dog breed identification app. To be specific, the following tasks need to be finished:

1. Write a human detector function using OpenCV's implementation of Haar feature-based cascade classifier.
2. Write a dog detector function using a pre-trained VGG16 model.
3. Create a CNN to classify dog breeds from scratch and attain a test accuracy of at least 10%.
4. Create a CNN to classify dog breeds using transfer learning and attain a test accuracy of at least 60%.
5. Write an algorithm that accepts a file path to an image and first determine whether the image contains a human, dog or neither. Then use the model develop in step4 to predict the dog breed.

Metrics

For both task 3 and task 4. Accuracy is used to measure the performance of the model. Test accuracy is defined as the number of dogs for which the breed is correctly predicted devided by the total test population.

For a CNN developed from scratch, a test accuracy of at least 10% is requited.

For a CNN utilizing transfer learning, a test accuracy of at least 60% is required.

2. Analysis

Data Exploration and Exploratory Visualization

The dataset is provided. It includes human images and dog images. The dog dataset is used to train the CNN model. Both human and dog images are used to test the performance of the dog detector and the human detector. In addition, the train dataset, valid dataset and test dataset is split in advance. There are in total 13233 human images and 8351 dog images. Train dataset contains about 80% of the total population and both valid and test dataset contain about 10% of the total population. All three datasets contain images for all 133 dog breeds. There are about 40 - 100 images for each breed, meaning some breeds have fewer samples compared to others. In addition, some breeds tend to have large image sizes. Based on the first 100 images of human images and dog images, 98% of the human images have a detected human face 0% of the human images have a detected dog image, 17% of the dog images have a detected human face and 100% of the dog images have a detected dog image.

A screenshot of the summary of the images count, image size is provided below:

	Height								Width							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
Breed																
001.Affenpinscher	64.0	445.765625	260.754936	225.0	319.0	375.0	506.50	2170.0	64.0	491.703125	232.621273	205.0	317.50	438.5	500.0	1500.0
002.Afghan_hound	58.0	585.689655	419.313190	223.0	360.5	500.0	595.75	3264.0	58.0	524.741379	345.586383	200.0	334.75	438.0	633.0	2448.0
003.Airedale_terrier	62.0	508.480769	404.034433	176.0	332.5	447.5	600.00	3192.0	62.0	468.384615	311.120439	190.0	341.00	411.0	500.0	2452.0
004.Akita	63.0	522.555556	299.638755	230.0	371.0	450.0	600.00	1762.0	63.0	541.888889	277.123130	169.0	374.00	494.0	640.0	1760.0
005.Alaskan_malamute	77.0	525.506494	408.754731	113.0	312.0	400.0	532.00	2560.0	77.0	581.116883	457.761113	113.0	336.00	405.0	640.0	3014.0

Algorithms and Techniques

The input image will be transformed in order to feed into the CNN. Details will be discussed in the section of data preprocessing.

The CNN developed from scratch has an architecture listed below:
Conv2d -> MaxPool -> Conv2d ->Maxpool -> Conv2d -> Maxpool -> Conv2d -> MaxPool -> Conv2d -> MaxPool -> Flattened -> FC -> FC
In addition, a ReLu function is chosen as the activation function and a dropout of 0.3 is applied before forwarding to a fully connected layer to avoid overfitting.

To utilize transfer learning, a pre-trained ResNet 50 model is used and a fully connected layer is added as the final classifier.

Benchmark

The CNN model developed from scratch is chosen as a benchmark model. And according to the project itself, an test accuracy of at least 60% is required.

3. Methodology

Data Preprocessing

For train_data, RandomResizedCrop, RandomHorizontalFlip and RandomRotation are applied to randomly resize the image to 224x224 and image augmentations is achieved as well. 224x224 is chosen for the input tensor as it's the minimum size required for PyTorch pretrained models.

For valid_data and test_data, only Resize to 224x224 is applied.

Data augmentation is achieved by applying RandomResizedCrop, RandomHorizontalFlip and RandomRotation to train_data. For every epoch during training, the dataloader will apply a fresh set of random operations “on the fly” so that the model can be generalized.

Implementation

For the CNN developed from scratch, the steps are listed below:

- 1. A Conv2d with kernal size of 3, stride of 1 and padding of 1 are used to transform an input size of (H, W, Channel) = (224, 224, 3) to (224, 224, 16) to achieve 'same' padding and increasing of channels. A ReLu activation function is applied. Then a MaxPool2d with kernal size of 2, stride of 2 is used to transform the tensors from (224, 224,16) to (112, 112, 16).

- 2. Same function as Step 1. A Conv2d with kernal size of 3, stride of 1 and padding of 1 are used to transform an input size of (H, W, Channel) = (112, 112, 16) to (112, 112, 32) to achieve 'same' padding and increasing of channels. A ReLu activation function is applied. Then a MaxPool2d with kernal size of 2, stride of 2 is used to transform the tensors from (112, 112, 32) to (56, 56, 32).

- 3. Same function as Step 1. A Conv2d with kernal size of 3, stride of 1 and padding of 1 are used to transform an input size of (H, W, Channel) = (56, 56, 32) to (56, 56, 64) to achieve 'same' padding and increasing of channels. A ReLu activation function is applied. Then a MaxPool2d with kernal size of 2, stride of 2 is used to transform the tensors from (56, 56, 64) to (28, 28, 64).

- 4. Same function as Step 1. A Conv2d with kernal size of 3, stride of 1 and padding of 1 are used to transform an input size of (H, W, Channel) = (28, 28, 64) to (28, 28, 128) to achieve 'same' padding and increasing of channels. A ReLu activation function is applied. Then a MaxPool2d with kernal size of 2, stride of 2 is used to transform the tensors from (28, 28, 128) to (14, 14, 128).

- 5. Same function as Step 1. A Conv2d with kernal size of 3, stride of 1 and padding of 1 are used to transform an input size of (H, W, Channel) = (14, 14, 128) to (14, 14, 256) to achieve 'same' padding and increasing of channels. A ReLu activation function is applied. Then a MaxPool2d with kernal size of 2, stride of 2 is used to transform the tensors from (14, 14, 256) to (7, 7, 256).

- Step1 - step5 follow a classic architecture for a CNN task to increase the dimension of channels and reduce the dimensions of both height and width.

- 6. The tensor is flattened.

- 7. A dropout is applied to avoid overfitting.

- 8. The tensor is passed through a fully connected layer with a output dimension of 1024.

- 9. A dropout is applied again to avoid overfitting.

- 8. The tensor is passed through a final fully connected layer with a output dimension of 113, as there are 113 classes of dog breeds.

For the CNN utilizing transfer learning, the steps are listed below:

- ResNet50 is picked.

- A new fully connected layer which has an input dimension of 2048 and output dimension of 133 (the number of classes for dog breed) is added as a final classifier.

- All the weights from the pre-trained network are freezed.

- The network is trained to update the weights for the new added fully connected layer only. We don't have a large dataset for training, the weights of the original ResNet 50 is hold constant to avoid overfitting.

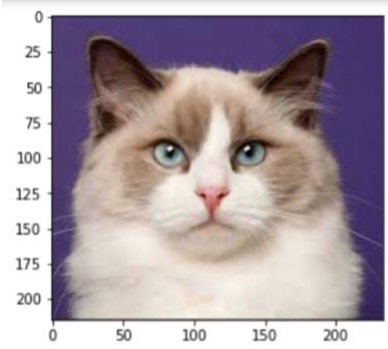
Refinement
Initially, an epoch of 20 is chosen, as the validation loss continues to decrease after 20 epochs is done, I increase the epoch to 30.

4. Results

Model Evaluation and Validation
For the CNN developed from scratch, a test accuracy of 16% is achieved, which meets the requirement.
For the CNN developed by combining ResNet50 and a fully connected layer, a test accuracy of 75% is achieved, which is also higher than the expected 60%.

In addition to the test dataset provided, 7 images(including 2 human images, 2 cat images, 2 dog images and 1 human and dog image) are provided to test the model. Among all 7 images, only one dog image, which is a pug is incorrectly predicted as a French bulldog. In particular,

the image with both human face and dog is predicted to be the dog breed in the image. And cat images are not detected as either a human or a dog.



```
Neither a dog or a human is detected, error!
```

```
Image:./test_images/liuyuxin and never.JPG, the predicted breed for the dog is Bichon frise.
```

Justification

A ResNet50 is chosen as it utilizes a 'identity block'/'convolutional block' to solve the gradient vanishing problem during training so that we can train a neural network with deep layers and achieve good performance. As ResNet50 was trained on ImageNet database, I think the pre-trained neural network layers already contain relevant high level feature information about the dog data set.

5. Conclusion

For CNN developed from scratch, we achieved
Test Loss: 3.438718
Test Accuracy: 18% (152/836)

For CNN combining ResNet50 and a fully connected layer, we achieved
Test Loss: 1.468853
Test Accuracy: 75% (631/836)

Transfer learning is really helpful as we can take advantage of both a good model architecture and pre-trained model weights so that we can save a lot of computing power and time.

Reflection and Improvement

As the training dataset is still not large enough, the model still has difficulty to predicting some similar dog breeds. If a larger dataset for different dog breeds can be provideds, maybe we can retrain the whole network from scratch with randomly initialized weights to better capture features of dogs.

More epochs can be tried.

A different architecture can be tested, or more layers can be added to achieve a higher accuracy.