Capstone Project

Yuwei Chen

September 8, 2019

I. Definition

Project Overview

The dog_app project is a visual object detection based on deep learning. Visual object detection is an important topic in computer vision, and has great practical merits in applications such as visual surveillance, autonomous driving and human-machine interaction . In recent years, significant breakthroughs of deep learning methods in image recognition research have arisen much attention of researchers and accordingly led to the rapid development of visual object detection

The problem can be divided into three steps:
1. Apply a pretrained network VGG16 to detect dog images.
2. Build a network by myself to classify the dogs;
3. Use transfer learning to predict dog breeds for higher accuracy.

My motivation for choosing this problem is my personal interest and experience for deep learning. I once carried out a research based on attention pooling, so this time, I consider the project a good chance to deepen my understanding on deep learning since CNN is one of the representative algorithm of deep learning.

Problem Statement

The purpose of this application is to achieve the following:
1. If detecting an image of a dog, return the breed.
2. If detecting an image of a person, return the resembling dog breed.
A total of 3 models, pretrained VGG_16, self_bulit CNN network and transfer learning CNN network, are asked to develop to fulfill certain requests: detect dogs or humans and classify the images to resembling dog breeds.

Metrics

The evaluation metrics are accuracy and test loss.
Test loss is updated by the formula below:
test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data - test_loss))
while since the dataset is balanced, accuracy is a good metric to evaluate the model. Accuracy is defined as following:
Accuracy = (TP+TN)/(TP+TN+FP+FN)

	Actual = Yes	Actual = No
Predicted = Yes	True Positive(TP)	False Positive(FP)
Predicted = No	False Negative(FN)	True Negative(TN)

II. Analysis

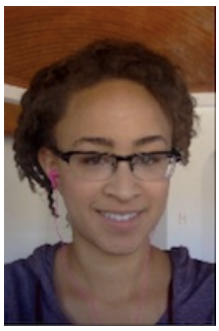
Data Exploration

The dataset for this project has a dog dataset named ‘dog_images’ and human dataset named ‘lfw’. There is no annotations or bounding boxes included for each image.
The ‘dog_images’ folder is shown below, which is divided into three parts: train, valid and test, including 133 kinds of dog breeds, a total of 8351 images.
Each breed contains 8 images, quite balanced:

['Affenpinscher', 'Afghan hound', 'Airedale terrier', 'Akita', 'Alaskan malamute', 'American eskimo dog', 'American foxhound', 'American staffordshire terrier', 'American water spaniel', 'Anatolian shepherd dog', 'Australian cattle dog', 'Australian shepherd', 'Australian terrier', 'Basenji', 'Basset hound', 'Beagle', 'Bearded collie', 'Beauceron', 'Bedlington terrier', 'Belgian malinois', 'Belgian sheepdog', 'Belgian tervuren', 'Bernese mountain dog', 'Bichon frise', 'Black and tan coonhound', 'Black russian terrier', 'Bloodhound', 'Bluetick coonhound', 'Border collie', 'Border terrier', 'Borzoi', 'Boston terrier', 'Bouvier des flandres', 'Boxer', 'Boykin spaniel', 'Briard', 'Brittany', 'Brussels griffon', 'Bull terrier', 'Bulldog', 'Bullmastiff', 'Cairn terrier', 'Canaan dog', 'Cane corso', 'Cardigan welsh corgi', 'Cavalier king charles spaniel', 'Chesapeake bay retriever', 'Chihuahua', 'Chinese crested', 'Chinese shar-pei', 'Chow chow', 'Clumber spaniel', 'Cocker spaniel', 'Collie', 'Curly-coated retriever', 'Dachshund', 'Dalmatian', 'Dandie dinmont terrier', 'Doberman pinscher', 'Dogue de bordeaux', 'English cocker spaniel', 'English setter', 'English springer spaniel', 'English toy spaniel', 'Entlebucher mountain dog', 'Field spaniel', 'Finnish spitz', 'Flat-coated retriever', 'French bulldog', 'German pinscher', 'German shepherd dog', 'German shorthaired pointer', 'German wirehaired pointer', 'Giant schnauzer', 'Glen of imaal terrier', 'Golden retriever', 'Gordon setter', 'Great dane', 'Great pyrenees', 'Greater swiss mountain dog', 'Greyhound', 'Havanese', 'Ibizan hound', 'Icelandic sheepdog', 'Irish red and white setter', 'Irish setter', 'Irish terrier', 'Irish water spaniel', 'Irish wolfhound', 'Italian greyhound', 'Japanese chin', 'Keeshond', 'Kerry blue terrier', 'Komondor', 'Kuvasz', 'Labrador retriever', 'Lakeland terrier', 'Leonberger', 'Lhasa apso', 'Lowchen', 'Maltese', 'Manchester terrier', 'Mastiff', 'Miniature schnauzer', 'Neapolitan mastiff', 'Newfoundland', 'Norfolk terrier', 'Norwegian buhund', 'Norwegian elkhound', 'Norwegian lundehund', 'Norwich terrier', 'Nova scotia duck tolling retriever', 'Old english sheepdog', 'Otterhound', 'Papillon', 'Parson russell terrier', 'Pekingese', 'Pembroke welsh corgi', 'Petit basset griffon vendeen', 'Pharaoh hound', 'Plott', 'Pointer', 'Pomeranian', 'Poodle', 'Portuguese water dog', 'Saint bernard', 'Silky terrier', 'Smooth fox terrier', 'Tibetan mastiff', 'Welsh springer spaniel', 'Wirehaired pointing griffon', 'Xoloitzcuintli', 'Yorkshire terrier']

Here are some examples of dog images:


While the lfw file contains totally 13233 images with different categories, here is a example of human images:



It is worthy mention that, each picture, no matter dogs or humans, have different sizes and should be transformed before classifying.

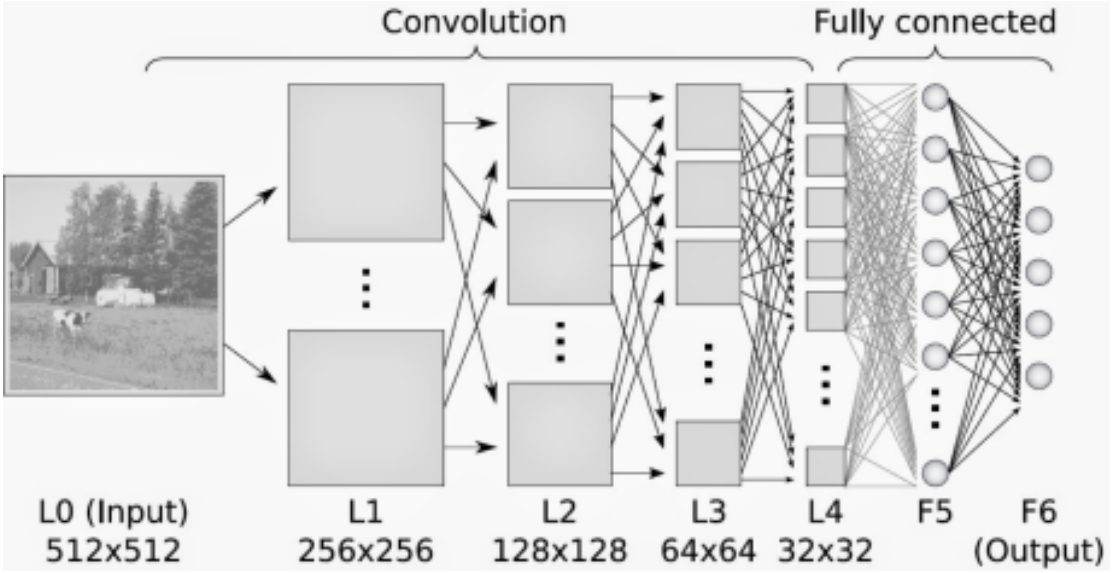
Algorithms and Technique

First, we utilize pre-trained models for human detection and dog detection. OpenCV is used to detect human faces(no need for implemention). For the dog detection, we load the VGG-16 pretrained by torchvision.models. The two models used for detection both achieve a high accuracy.

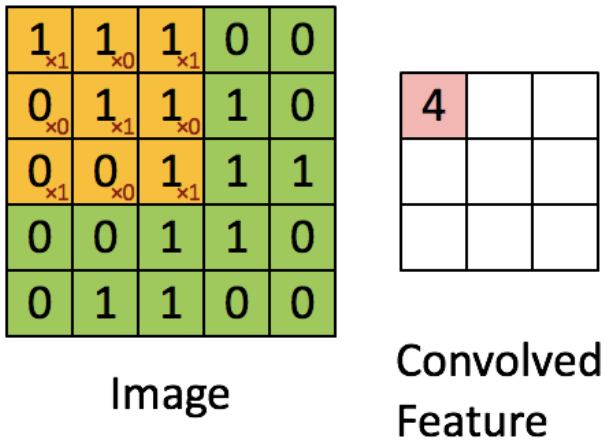
Next, I create a CNN model for dog breed classification by myself with the minimal test accuracy to pass this portion is set to 10%. I built a model similar to VGG net, including convolution layers and pooling layers with the same kernel size and orders, followed by fully connected layer leading to a 133 elements vector representing 133 categories of dogs. Also, in this process, I add the dropout layer to prevent overfitting.

Finally, with regard to transfer learning, I also utilize the pretrained Resnet_50, only change the last fully connected layer to get exactly 133 output features.

All the models are based on CNN structure. There are usually four main steps in CNN: convolution, subsampling, activation and fully connectedness.



In my own perspective, the most important step is convolution. Convolution has the nice property of being translational invariant. Intuitively, this means that each convolution filter represents a feature of interest (e.g whiskers, fur), and the CNN algorithm learns which features comprise the resulting reference (i.e. cat). The output signal strength is not dependent on where the features are located, but simply whether the features are present. Here are the formula and picture which can represent the convolution step:



The formula can be wrote as

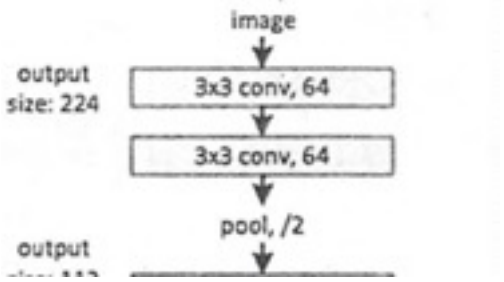
$$y(t) = \int_{-\infty}^{\infty} x(p)\, h(t-p)dp = x(t) * h(t)$$

Another important method is transfer learning. Generally speaking, transfer learning means the weight of each node in a layer-by-layer network is transferred from a trained network to a brand-new network instead of training a neural network for

each specific task from the beginning. The benefit of this approach is that when you already have a well performed pretrained model, you can load the parameter to extract the features of images and add modify a little to do some further task, like we can use a model to detect dogs to develop a model for classifying dog breeds.

Benchmark

The model I would like to talk about is VGG_19, based on basic VGG network and Resnet, which are both typical tool for extracting image features. The critical block of Resnet_50 and VGG are listed below:



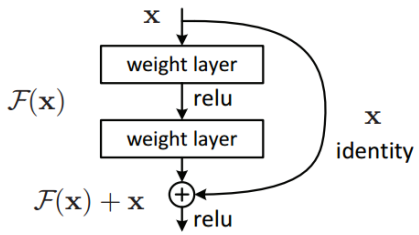
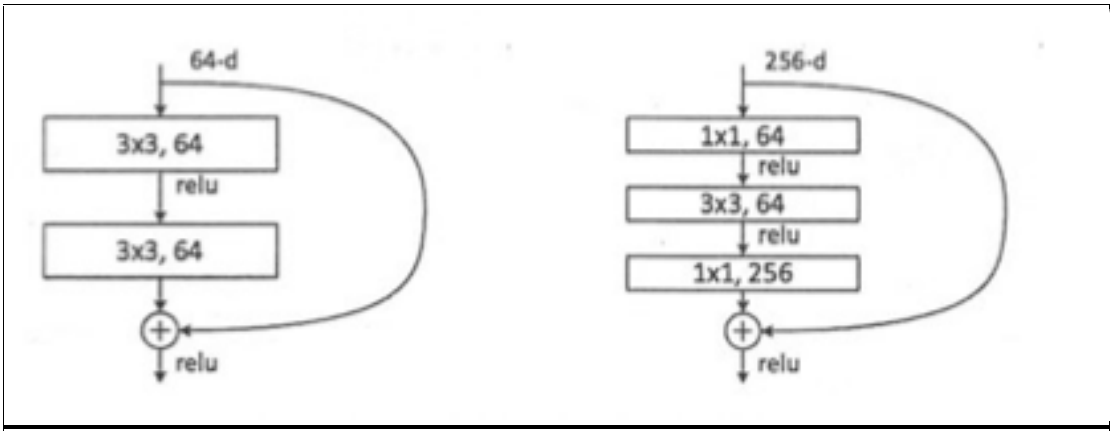


Figure 2. Residual learning: a building block.

Combine these two structures, VGG_19 has a network structure presented below:



The VGG_19 has a total of 34 layers, using conv layers to extract features and short cut to implement information loss.

III. Methodology Data Preprocessing

Before building the network, we should load the data from the files(train, valid and test). It is necessary to transform the pictures into the standard format suitable for the pretrained VGG model. The standard format is size 224*224, and a normalize parameter as $[0.485,\ 0.456,\ 0.406]$, $[0.229,\ 0.224,\ 0.225]$. Also, set the batch size to be 100. The transformation is achieved by the following code:

```
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                  std=[0.229, 0.224, 0.225])
data_trans=transforms.Compose([transforms.Resize(256),
                               transforms.CenterCrop(224),
                               transforms.ToTensor(),
                               normalize,
                              ])
```

Implementation

First, we define a pre-trained model for human detection and dog detection. We use OpenCV to detect human faces. For the dog detection, we use the VGG-16 implemented by torchvision. We assume the accuracy of detection to be more than or equal to 70%.

Next, we create a CNN model for dog breed classification, We develop the model from scratch and minimal test accuracy to pass this portion is set to 10%. each block is organized by a convolution layer (kernel size 3*3), a activation function(relu), a max pooling layer (kernel size 2*2) and a dropout layer to reduce overfitting. The fully connected layer is activated by sigmoid function. It is worthy to mention that, between conv layers and fc layers, there is a reshape step to make the features suit the shape of fc layer input. The code is below:

```
x = self.conv1(x)
x = F.relu(x)
x = self.pool(x)
x = self.drop(x)


x = self.conv2(x)
x = F.relu(x)
x = self.pool(x)
x = self.drop(x)

x = self.conv3(x)
x = F.relu(x)
x = self.pool(x)
x = self.drop(x)


x = x.reshape(x.size(0), -1)

x = self.fc1(x)
x = F.sigmoid(x)

x = self.fc2(x)
x = F.sigmoid(x)
```

The optim function is Adam and the loss is measure by cross entropy.

As for transfer learning, it is carried out as following:

```
## TODO: Specify model architecture
model_transfer = models.resnet18(pretrained=True)
for param in model_transfer.parameters():
    param.requires_grad = False

# Parameters of newly constructed modules have requ
num_ftrs = model_transfer.fc.in_features
model_transfer.fc = nn.Linear(num_ftrs, 133)
```

With the same loss function and a optim method SGD.

Refinement

1.More data
For deep learning (machine learning) tasks, more data means more input space, which can bring better training results. Data Augmentation and Generative Adversarial Networks can be used to extend the data set. At the same time, this method can also improve the robustness of the model.
2.Different Algorithms
OpenCV toolkit has now became a common way for visual object detection because of its convenience and efficiency. Despite of opencv, deep learning methods has been developed in this area. Various network structure like AlexNet, VGG and Resnet has improved the accuracy to a large extent. So if we want to get better user experience and accuracy, we can try more new object detection algorithms, such as some algorithms based on deep learning.
3.Multi-objective monitoring
Furthermore, we can accomplish the task of detecting multiple targets simultaneously in a picture by some advanced target recognition algorithms, such as RCNN, Fast-RCNN, Faster-RCNN or Masked-RCNN.

IV. Results

Model Evaluation and Validation

The main reasons for this architecture are as follows: 1) the convolution layer senses the features in the image locally, so that it can synthesize the local information from a higher level; 2) the pooling layer is used to reduce the dimension of features, compress the number of data and parameters, reduce over-fitting, and improve the fault tolerance of the model. Sex; MaxPooling can retain the strongest features and discard other weak ones. Global Average Pooling pools the features of the last layer to form a feature point, and composes these feature points into a final feature vector for soft Max calculation. 3) The fc layer is a highly purified feature, which will be classified finally. The gradient of sigmoid is very gentle in the saturated region, close to 0, and very tolerant. It is easy to cause the problem of gradient disappearance. The gradient of Relu is constant in most cases, which is helpful to solve the convergence problem of deep network.

Justification

All the implementation part in the code meet the pass accuracy. The transfer learning model even achieve a accuracy about 75%, almost equal with the result of

my benchmark model VGG_19, which has a accuracy of 78%. If there are more iterations or more training time, it could be better.