Dog Breed Classification

Xinqi Shen

Abstract.

Object detection is one of the popular research directions in computer vision field. The main idea of object detection is localization and classification. The current approaches are mainly based on deep learning, such as R-CNN, Fast R-CNN and YOLO etc. They do provide better performance and accuracy. In this project, our team used a traditional machine learning based approach(HOG + SVM) to perform localization, and then use the idea of deep learning(VGG16BN) to do classification. Also, in order to get better visualization, we performed segmentation(UNET) as our final step.

1 Introduction

The goal of our project is to detect and localize the dog in the given image and then classify it based on breed. We have a total of 120 different types of dog breeds, such as Chihuahua, Pekinese and Boxer etc. Our goal is to help people who are not familiar with the breed of dogs, to identify its breed. In the end, we want to reach a relatively high accuracy (better than us). Before starting the project, we already learned many methods or skills related to this project in the course and also did some extra research, so we decided to use HOG + SVM + CNN to perform dog breed classification. If referencing back to concepts/algorithms we learned in the course, our approach covered topics of image processing, feature extraction, and deep learning. The main challenge we expect is that since we have a large amount of class/breed to classify, we firstly need enough data to train our neural network and it will take a very long time in the training process. Also, some dogs with different breeds may look very similar, it eventually will affect our accuracy of predication, so we need to find the CNN model with the best performance. After some comparison, we choose VGG16BN as our neural network model. I will discuss more detail in the following section.

2 Method

We divide our project into two parts, detection and classification. My teammate Haolin Zhang response for detection part using (HOG + SVM) and I response for classification(VGG16BN) and segmentation(UNET). Figure 1 visualized our workflow.

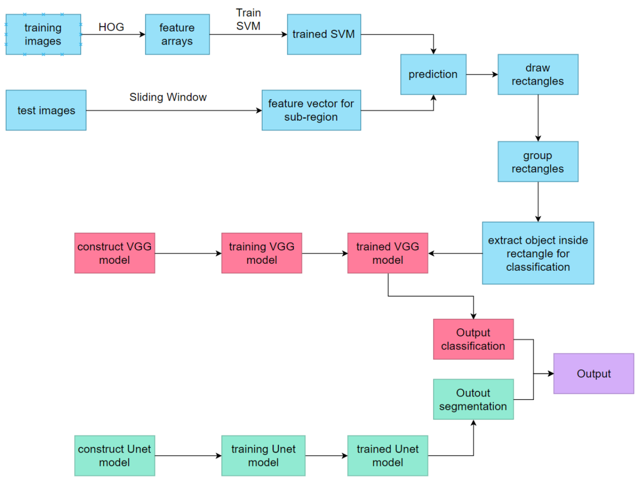


Figure 1: the workflow of our project

2.1 Detection

First of all, he used HOG to extract the features of all the positive(dog) and negative(non-dog) training images and then fed into linear SVM to train the dog classifier.[1] After that, he applied a sliding window with various sizes to scan the whole test image and extracted the HOG feature for each patch. Then, each patchs HOG feature array was passed into the trained SVM in order to decide whether the patch belongs to a dog. There are two special cases here: if no patch labelled as the dog, the given image did not include dog then we stopped; if multiple patches labelled as the dog, we grouped them to a larger patch(Figure 2(a)). Then, the final rectangle patch provided the bounding box of the dog in the given image (Figure 2(b)), and the bounding box part will be used to do classification. Note that, this part of code is in file 'ProjectSVM.ipynb'.

Figure 2: (a) multiple patches before grouped. (b) the final bounding box. Image credit to Haolin Zhang

2.2 Classification

I used the VGG16 model in this part, the idea is from the paper 'Very Deep Convolutional Networks for Large Scale Image Recognition'.[2] This model performed very well with respect to the classification task(winner in 2014 ImageNet competition). The biggest advantage for this model is simplicity since the model use the same 3x3 filter in every convolution layer and the same 2x2 max pooling operation for downsampling, and just stack multiple convolution and max pooling layers together. In other words, small filters, deeper networks. Noted that, I finally choice VGG16BN which simply adds the batch normalization layers to prevent overfitting. The complete structure is shown in Figure 3.

Note that, this part of code is in file 'ProjectCNN.ipynb'.

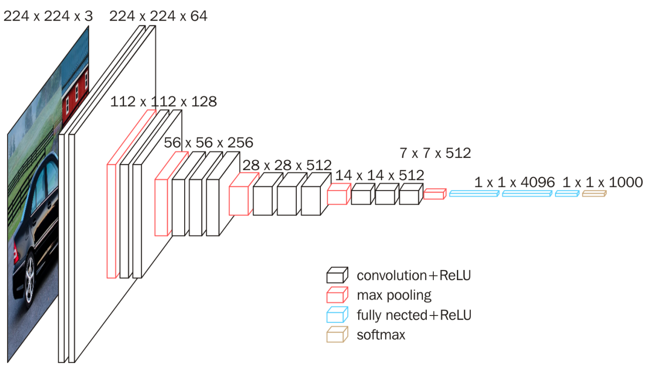


Figure 3: the structure of VGG. Image from online

In terms of classification, there are five steps in this part:

1. **Get dataset and Image pre-processing:** I used the Standford dogs dataset which contains images of 120 breeds of dogs.[3] To image pre-processing, I cropped all the images based on its bounding box annotation and split into training and test set. I will use the cropped images to train my model since the output from the detection part was fed in a cropped image as well.

2. **Build model:** I imported the VGG16BN model from pytorch[4], then I modified its structure. Since the original VGG16BN can classify 1000 classes, so I kept the feature layer the same and changed the final classifier to fit our purpose(120 classes).

3. **Transfer learning:** Since pytorch provided the pre-trained weight, I was able to transfer learning by loading the weight as our model initial weight.

4. **Training and Fine-tuning:** I trained the model using our cropped training dataset. This step is fine-tuning, since the weight from large dataset already contains many generic features, we want to become more specific to the details of higher-level features for our dog dataset.

5. **Test:** I kept tuning parameter and re-train our model until reached the highest accuracy in test image set. (epoch 30, learning rate 0.0001, batch size 16)

2.3 Segmentation

This part was similar to classification, the only difference is model. I used the UNET[5] here since we already know this model is appropriate to segmentation based on the assignment. The output should be a mask(Figure 4). Note that, this part of code is in file 'ProjectCNN.ipynb'.

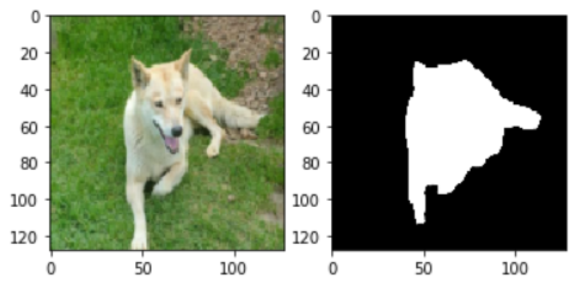
5 

Figure 4: the dog segmentation. Image credit to me

3 Results and discussions

3.1 Comparison

I mainly compared the following aspects:

- **VGG16 VS VGG16BN:** The additional batch normalization can increase the test accuracy around 2% - 5%, it prevented overfitting when epoch getting large
- **Transfer learning or not:** Transfer learning increase the test accuracy by around 15%, it is significantly improved.
- **Data augmentation or not:** I implemented random resized crop, random horizontal flip and normalization. The test accuracy can increase by about 8%. Noted that, If random vertical flip was added, it decreased the test accuracy sometimes!

If I used the VGG16BN with transfer learning and proper data augmentation, then at epoch 30, the test accuracy will reach around 83%.

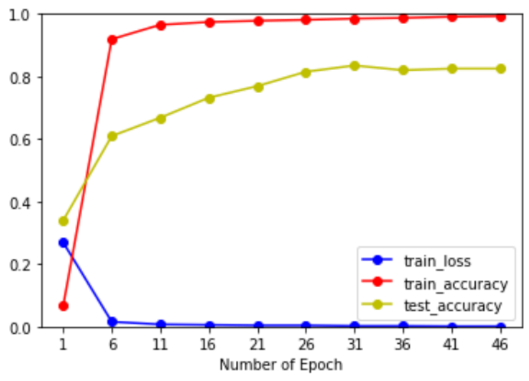
5 

Figure 5: the loss and accuracy plot. Image credit to me

3.2 The output results and discussions

Here are some correct outputs:

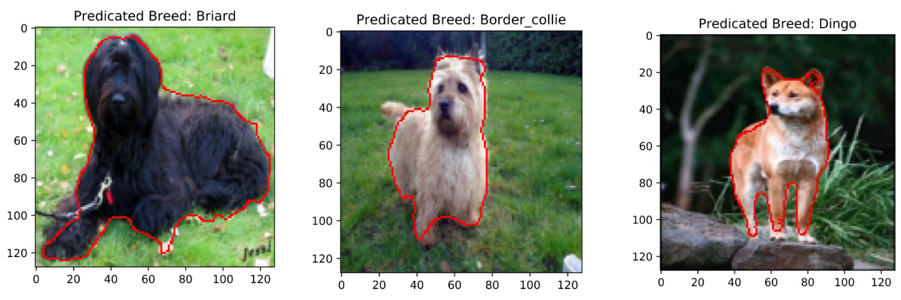


Figure 6: the dog segmentation with predicted breed name. Image credit to me

Moreover, I was also interested in which breeds have the highest and lowest accuracy from the model prediction.

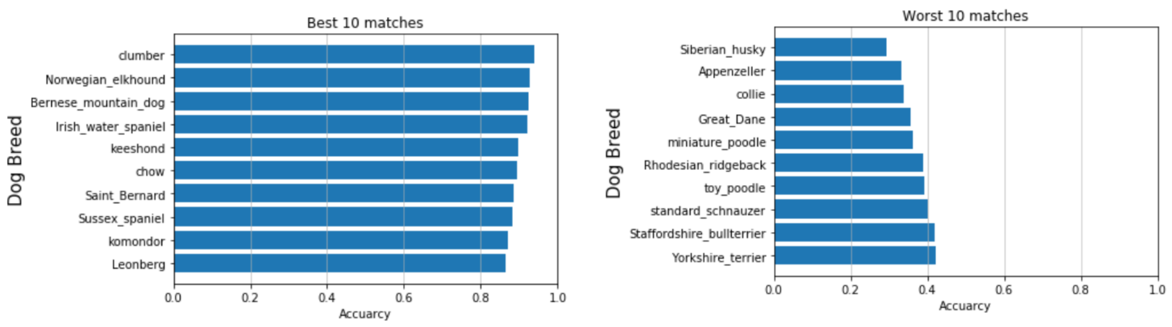


Figure 7: (a) the best 10 matches. (b) the worst 10 matches. Image credit to me

First of all, all the best 10 matches have over 85% accuracy in the test set. Our model performed very well in these dog breeds. If I look at the images of the dog with these breeds and investigate the common pattern, not surprisingly, these dog breeds have very obvious characteristics. Thus, our model can easily find and distinguish them. However, in terms of worst 10 matches, I found it may be caused by the following two reasons: the dog with that breed is very similar to other breeds, or the dog with the same breeds looks very different. Thus, the former reason may lead to our model to predict the breed that looks similar. I think it is an acceptable mistake to some extent since it is difficult to judge the breed of a dog by only its appearance even a dog expert. The latter reason may be caused by the limited number of training image sets, we can expand our training set to include more images for that dog breed and the common feature can be easily extracted.

4 Main challenges

One challenge is that the model takes too much time to train. Since we have 120 classes to classify and we need enough training images for each breed, the training set is very large. Thus, the parameter-tuning step is time-consuming. In terms of debugging, I spend many hours in the transfer learning step since I accidentally downloaded the pre-trained weight for VGG16 instead of VGG16BN. Also, I found the random vertical flip for data augmentation will decrease the test accuracy, so it takes me some time to find it.

5 Conclusions and future work

In summary, I contributed to the classification and segmentation part of this project. More specific, I firstly found and cleaned the dataset. Then, I investigated and built the best model fitting our purpose and then trained it with numerous parameter-tuning. In the future, I will use the deep learning approach, such as YOLO or Faster-RCNN, it should perform better and are more effective on the detection and localization of the object than our traditional approach.

References

[1] Navneet Dalal, Bill Triggs. Histograms of Oriented Gradients for Human Detection. International Conference on Computer Vision & Pattern Recognition (CVPR '05), Jun 2005, San Diego, United States. pp.886893.

[2] K. Simonyan and A. Zisserman. (2014). Very deep convolutional networks for large-scale image recognition..Available: https://arxiv.org/abs/1409.1556.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database. IEEE Computer Vision and Pattern Recognition (CVPR), 2009.

[4] VGG16BN Model: https://pytorch.org/docs/stable/_modules/torchvision/_models /vgg.html.

[5] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv, abs/1505.04597.

[6] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao and Li Fei-Fei. Novel dataset for Fine-Grained Image Categorization. First Workshop on Fine-Grained Visual Categorization (FGVC), IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.