

CSC420 Project Report – Dogs VS. Cats

Abstract:

In our project, we investigated identifying the certain kinds of dogs (Chihuahua, Newfoundland, Pug, Saint Bernard and Samoyed) and cats (Bengal, Bombay, Maine Coon, Ragdoll and Russian Blue) from an input image. If there are dogs or cats in the categories we studied, we would detect them and indicate which kind of dog or cat it is, otherwise we would indicate that there is no dog or cat that we want. We picked this topic since this can help people classify different kinds of dogs and cats in an image as well as help people learn about different kinds of dogs and cats, and we found the previous related work from Stanford^[1] and Oxford^[2]. In our project, we combine the datasets from the Stanford and Oxford's previous work.

Following are some contributions we made in this project: first of all, we found the datasets for our 5 kinds of dogs and 5 kinds of cats which are considered as our positive datasets, and then we find the negative datasets which can be used in our detection part. Secondly, we introduced a classifier after training our datasets. Next, we begin the detection parts which composed of using **HOG**^[3] (Histogram of Oriented Gradients), which simplifies our image by extracting the useful information (features) in our training images and discarding the extra-neous information (features), and using **SVM**^[4] (Support Vector Machine) to categorize new examples using our labeled training data. Lastly, we entered the classification part, and use the **CNN** (Convolutional Neural Network) model we have trained to predict which class should our detected object be in.

In the final result, we use the CNN that we had trained with the validation accuracy about 0.85 to classify the object (i.e.: dog or cat) that we detected from our test images with the accuracy rate about 0.9 Then, our project can output the correct category which our detected object belongs to.

1. Introduction:

We expected that when we input an image with dog or cat which is indeed one of the categories we studied, our project would detect it and determine which class it belongs to. However, if there is a cat or dog which not belongs to any one of the class we studied, it may still be detected by our detector and may be put into the categorization which is the most similar one. There are two main challenges of this project, first, we need to find the CNN model and the appropriate parameters with high validation accuracy and low validation loss, second, the detection part is the key of the whole project, since only when our detector can detect the dog and the cat in the image correctly, can we classify it correctly. Since we use the SVM approach in the detection part, we need both positive and negative datasets. It is easy to find the positive dataset but it may be hard to find the negative datasets^[6], since our negative datasets should have some similarities and differences with our positive datasets.

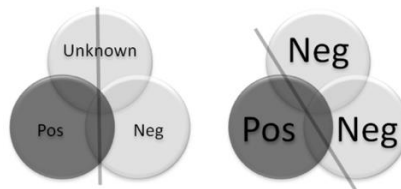


Figure 1.1 Illustrates the benefit of using examples from as many negative classes as possible.

My job in this project is to find the datasets and use the datasets I found to train the CNN and adjust the parameters in order to reach the highest validation accuracy and lowest validation loss. Also, I am responsible for the classification part which is after the object in the input image has been detected and extracted in the detection part. Since we use SVM method in the detection part, we need both positive and negative datasets. I am responsible for finding both positive and negative datasets for the training part in the detection.

In my part of job, there are several CNN architectures online^[7]: LeNet, AlexNet, VGG, GoogLeNet, ResNet, etc. But some of them, for example, original VGG16 is too slow and the parameters for sample_per_epochs, epochs, batch size and learning rate have a great effect on the validation accuracy, validation loss and the training time. In our code, I choose two types of CNN models for our detection. The first one is

‘smallVGG16’ in our CNNclassification class, in which I reused keras’ built-in vgg16^[8] and modified it with adding some layers, and then introduced keras’ built-in optimizer sgd^[9]. When I use this modified CNN version to train my datasets, the validation accuracy can reach around 0.85 which we consider it as a good model. By contrast, we include another model ‘traditionalCNN’ which is the traditional CNN model with multiple layers. This model cost a long time for us to train our datasets, and the accuracy is about 0.65 which is not as good as the previous one. Therefore, our project actually uses ‘smallVGG16’ model while ‘traditionalCNN’ model is used as a comparison.

2. Datasets and Evaluations: [WLOG, we consider the datasets for dogs for example]

2.1 Positive and Negative Datasets for the Detection Part^[10]

Positive Training Dataset The images in the positive training dataset should include what I want to detect, i.e.: dogs, therefore, I collected the images from different perspectives of different dogs. Since SVM just wants to extract the features of the dogs from our positive training dataset, we randomly pick different kinds of dogs.

Negative Training Dataset The images in the negative training dataset should include everything except what I want to detect. Therefore, I considered choosing the images of different objects and landscapes from the online resource.^[11]

2.2 Training and Testing Datasets for Training CNN Model

Our project is supposed to detect dog (or cat) and then classify them into the correct category, we consider the following 5 kinds dogs (similarly, 5 kinds of cats - Bengal, Bombay, Maine Coon, Ragdoll and Russian Blue, with similar number of images for each type to train and test):


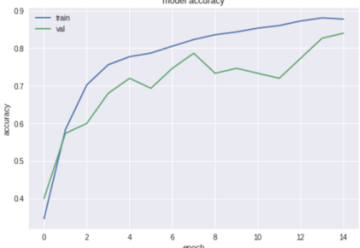
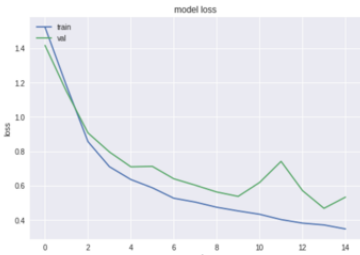




Breed	Sample Image	#Training	#Testing	Final Training Result
Chihuahua		388	21	Model Accuracy  Model Loss 
Newfoundland		381	22	
Pug		386	21	
Saint Bernard		356	21	
Samoyed		404	21	
#Total		1915	106	Data for Final Epoch: loss: 0.3478 - acc: 0.8777 val_loss: 0.5331 - val_acc: 0.8400

Figure2.2 Statistic of the CNN training dataset

In order to get the dog datasets for this part, I selected 5 same types of dogs from the datasets of Stanford and Oxford and combines them together to derive our own datasets for training the CNN model.

3. Methodology

3.1 Detection

HOG - Histogram of Oriented Gradients^[3] On the high-level, HOG algorithm extract the ‘useful’ features from our images and then from these ‘useful’ features, the algorithm can detect the objects from the image. In

order to reach the expectation, HOG algorithm processes as follow: first of all, the algorithm would compute the gradient values. In most cases, the technique applies the 1-D centered, point discrete derivative mask in both horizontal and vertical directions. There is one thing should be pointed out in this step, this method requires filtering the intensity of the image with the filter kernels: $[-1, 0, 1]$ and $[-1, 0, 1]^T$. Next, the algorithm creates the cell histograms which is based on the result we calculated in the previous gradient computation step. Depend on whether the gradient is ‘unsigned’ or ‘signed’, the histogram channels can be evenly spread over 0 to 180 degrees which is what we show in the below Figure3.1.1, or 0 to 360 degrees. In the next step, the algorithm groups the cells together into larger, spatially connected blocks in order to illustrate the changes in illumination and contrast. Then, our desired HOG descriptor is the concatenated vector of the components of the normalized cell histograms from all of the block regions. Last but not least, the algorithm computes the normalized vector containing all histograms in a given block. Finally, the HOG descriptors can be used as features to a machine learning algorithm, i.e.: SVM in our next step, for the further object detection process.

SVM - Support Vector Machine^[4] On the high-level, SVM algorithm is a machine learning algorithm which in our project combine with the previous HOG algorithm assigns new examples to one category or the other which making it a non-probabilistic binary linear classifier. SVM mechanism constructs a hyperplane or set of hyperplanes in a high-dimensional space, then the algorithm uses the constructed space for classification, regression, or outlier detection. Figure3.1.2 shows the example for the kernel machines which compute a non-linearly separable functions into a higher dimension linearly separable function.

In our project, without loss of generality, consider dog, we want to detect whether the object is dog or not. In the HOG step, we have already calculated the feature vectors for each image and labelled them dog (i.e.: 0) or not dog (i.e.: 1). Then, SVC algorithm would find an optimal hyperplane which can separate the labels that we calculate in the HOG step as Figure3.1.3.

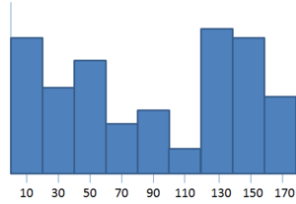


Figure3.1.1 Cell histograms in the HOG

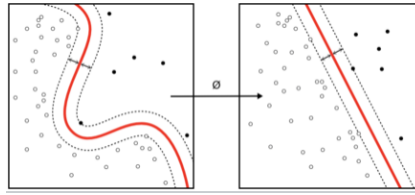


Figure3.1.2 Kernel machine mechanism

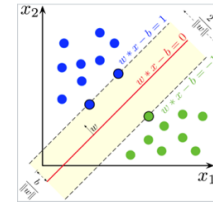


Figure3.1.3 Linear SVM algorithm

Sliding Window In the last step of detection part, we detect the object from our input image using sliding window and combine with HOG process to extract the feature vectors from the images, then we put the extracted feature vectors into our trained SVM. We draw the rectangle to the window which is labeled 1, i.e.: indicated as dog object. Last but not least, we group together the sliding windows we drew in the previous step into a large window. Finally, we have detected a dog from our input image.

3.2 Classification

CNN – Convolutional Neural Network In the classification part of our project, we use the modified version of the VGG16 model (i.e.: ‘smallVGG16’) in the *CNNclassification* class. In our project, we reuse the convolution layers of the built-in VGG16 model from the *keras*. This built-in VGGNet model consists of 16 convolutional layers and is very appealing because of its very uniform architecture. The general architecture shows as Figure3.2.1. In our ‘smallVGG16’ model, we introduce these 16 convolution layers, however, we construct our own flatten and dense layers. We now explore deeper into these two layers: (1) Flatten layer: in this layer there are no parameters to learn, and it only reformats the data. After the pixels are flattened, the network consists of a sequence of (2) Dense layer: each Dense layer are densely-connected neural layers. In the Dense layer, it learns from the parameters the number of nodes should have in the dense layer. Each node contains a score that indicates the probability that the current image belongs to one of the class we studied (i.e.: for dog is 5 classes: Chihuahua, Newfoundland, Pug, Saint Bernard and Samoyed). There is one other thing can be pointed out that we also use the Keras built-in *sgd* optimizer which is required for compiling a Keras model.

Even though we do not use the traditional CNN model in our project, ‘*traditionalCNN*’ model in the *CNN-classification* class reflects the basic technique of the traditional CNN model and we would experience how it works in general.^[12] In order to build the model, there are four main operations in the ConvNet: convolution, non-linearity, max pooling and classification. The process shows as Figure3.2.2, and these operations are the basic building blocks of every CNN model.

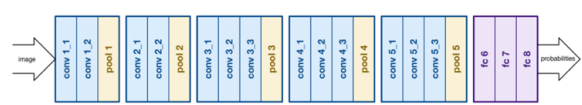


Figure3.2.2 Traditional CNN Process

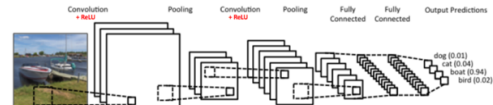
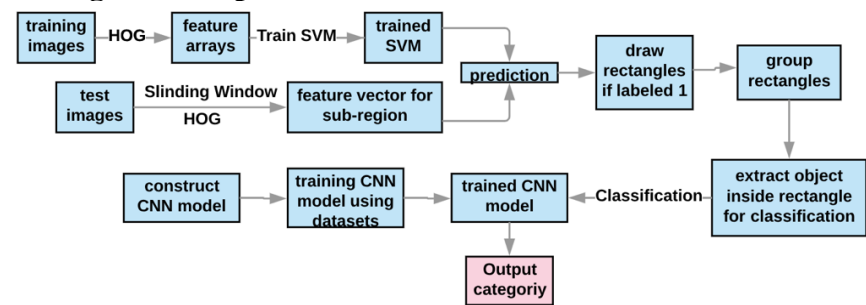


Figure3.2.1 VGG16 Built-in Model

4. Results and Discussion

4.1 Algorithm Pipeline



4.1 Intermediate Step Result

Image before grouping	Image after grouping	Extracted object	Final classification output

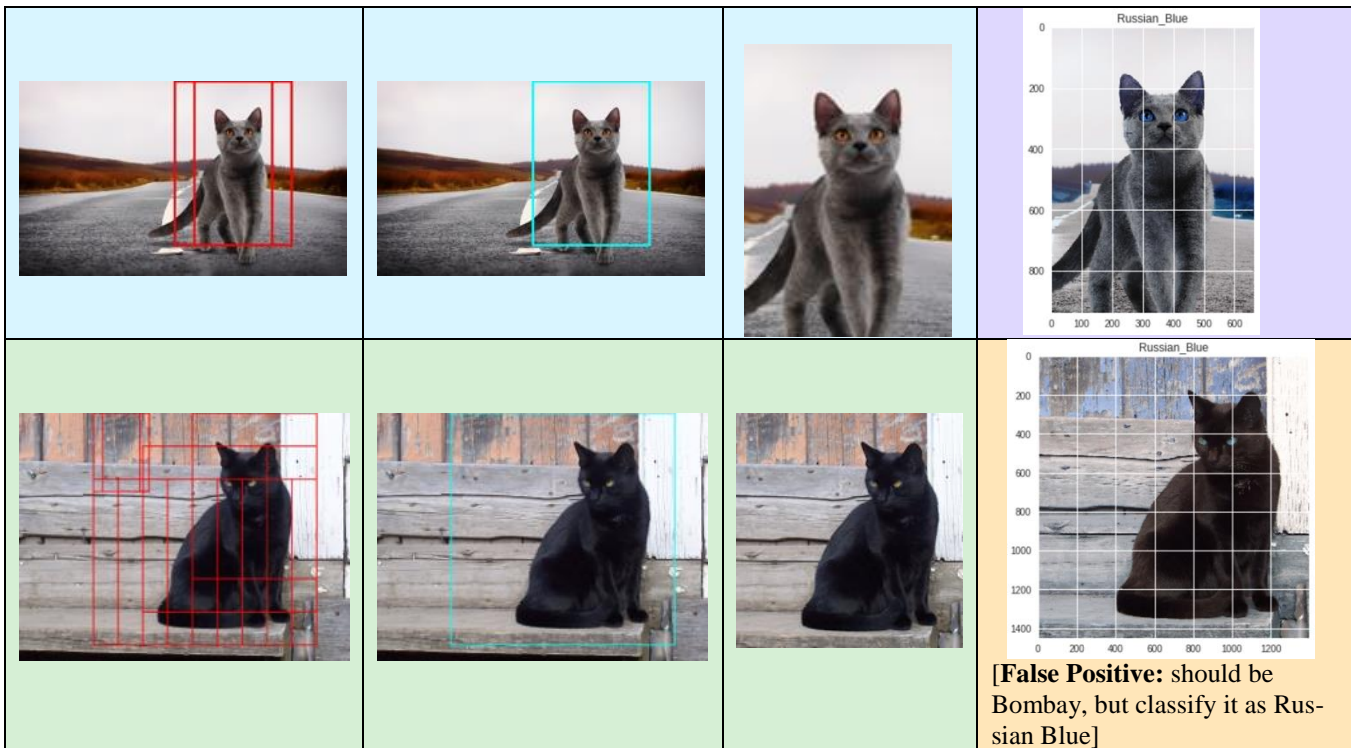


Figure 4.1 Result of Intermediate Step

In detection part, we have 3 steps: HOG, SVM and Sliding window. In the HOG part, we choose the configuration with window size = 64×64 ; block size = 16×16 ; block stride = 8×8 ; cell size = 8×8 ; bins = 9. This configuration gives use the feature vectors of shape (15876, 1). In the SVM part, we tried two different built-in SVM settings and compared them. When we used SVM implementation in OpenCV, along with its default support vectors array and alpha-rho value, but this did not give a good prediction and detection in the test step. Therefore, we changed our original approach and considered using sklearn's built-in SVM, with its SVC (gamma='scale') function, this gave much better result and reduced false positive detections. In the sliding window part, we have struggled with adjusting the size of the sliding window and the size of stride since these parameters have a great influence on our final result.

In the Figure 4.1, the first column are the image outputs including the sliding windows with the label 1, after grouping the windows, we have the second column, which are the images with just one bounding box around the object. Then, we extract out the object in the bounding box, and put them into our trained CNN model to classify the object.

In the CNN part, I trained the different models with different parameter and found that, in one CNN model, when increasing batch size, in most cases, training and validation loss converges quickly, and the final loss rate seems decrease slightly. The other parameter which may have some effects on the training and validation accuracy is learning rate, in most case, when we increase learning rate, the loss and validation accuracy and loss rate may not stable. When we pick the number of epochs too large, we may encounter overfit (i.e.: the validation loss begins to increase). Also, when I increase sample_per_epochs, the training time may increase.

At first, when we trained our detection part, our code produces the very bad result, and we found that our code can work for static object (i.e.: dog), but when the dog in the image with 'wired' state, for example, running, jumping, turn its back to the viewer or the light condition is bad, our code cannot detect correctly. We then found that this is because our positive datasets did not include enough different status of our target object. In order to fix this problem, we include more image about different status of the target object with different light condition into our positive datasets. And, finally, our code produced a fairly good result than before.

5. Main Challenges

There are several challenges in our project: first of all, in the detection part, we spent tons of time finding the positive and negative datasets for the training process of the SVM algorithm. Figure 1.1 illustrates the ideal situation of the distribution of positive and negative datasets. But in reality, there always occurs some unexpected situation like Figure 5.1.

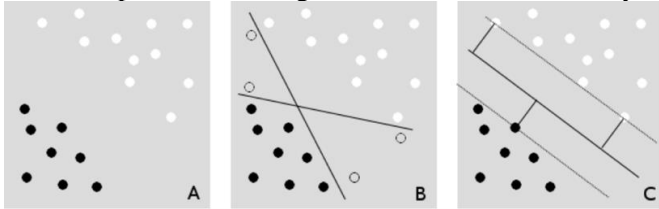


Figure 5.1 Possible situation when finding the datasets, different color represents different classes.

Part A of Figure 5.1 shows the data points from two different classes, whether is the target object or not. Part B of Figure 5.1 shows two of many possible separating lines. Part C of Figure 5.1 shows maximum margin separating line. In order to reach the perfect result, the positive dataset should include different movements of the target object (i.e.: dog), different types of the dog in different light condition from different point of view. It is difficult to include all the movements of our target object, i.e.: dog. By contrast, it is also difficult to find the negative datasets, since we are expected to include as many objects other than our target object as possible, but it is hard to include all of them, and there may also have false positives, since some objects may be mistaken by SVM as our target object. Another challenge in our project is to find the most suitable CNN model by training our datasets, this took us around a week to train an appropriate CNN model, since we not only need to find the correct CNN model but also need to adjust various variables, for example, sample_per_epochs, epochs, batch size and learning rate.

6. Conclusion and Future Work

In this project, we detected the dog and cat from the image using HOG, SVM and Sliding window methods and then we use CNN model to classify the object into the categories we studied. In the HOG, SVM and CNN model parts we need first trained the model and then use these trained models to test the input images. Even though we seems have approach the end of the project, there are still many problem wait to be dealt with, for example, in our project, in the classification part, if there is dog which is not in one of our studied categories, our code may identify it into the most similar class, which not quite make sense, therefore, if we still have time we would consider fix this part, and output the message if the detected dog does not belong to any class.

References:

- [1] "Stanford Dogs Dataset For Fine-Grained Visual Categorization". *Vision.Stanford.Edu*, 2018, <http://vision.stanford.edu/aditya86/ImageNetDogs/>.
- [2] "Visual Geometry Group: Oxford-IIIT Pet Dataset". *Robots.Ox.Ac.Uk*, 2018, <http://www.robots.ox.ac.uk/~vgg/data/pets/>.
- [3] Mallick, Satya. "Histogram Of Oriented Gradients | Learn Opencv". *Learnopencv.Com*, 2018, <https://www.learnopencv.com/histogram-of-oriented-gradients/>.
- [4] "Chapter 2 : SVM (Support Vector Machine) — Theory – Machine Learning 101 – Medium". *Medium*, 2018, <https://bit.ly/2pFAPFo>.
- [5] "Applications - Keras Documentation". *Keras.Io*, 2018, <https://keras.io/applications/>.
- [6] 'Selecting Negative Examples for Training an SVM Classifier' *Eit.Lth.Se*, 2018, <https://www.eit.lth.se/srapport.php?uid=301>.
- [7] "CNN Architectures: Lenet, Alexnet, VGG, Googlenet, Resnet And More". *Medium*, 2018, <https://bit.ly/2mWCUw1>.
- [8] "Applications - Keras Documentation". *Keras.Io*, 2018, <https://keras.io/applications/>.
- [9] "Optimizers - Keras Documentation". *Keras.Io*, 2018, <https://keras.io/optimizers/>.
- [10] "How Can I Train Svm With My Own Dataset? · Issue #1 · Ahmetozlu/Vehicle_Counting_Hog_Svm". *Github*, 2018, https://github.com/ahmetozlu/vehicle_counting_hog_svm/issues/1.
- [11] "Jaredjxyz/Carnd-Vehicle-Detection". *Github*, 2018, <https://github.com/jaredjxyz/CarND-Vehicle-Detection>.
- [12] "An Intuitive Explanation Of Convolutional Neural Networks". *The Data Science Blog*, 2018, <https://ujjwal-karn.me/2016/08/11/intuitive-explanation-convnets/>.