Anime Face Detection & Recognition Project Report

Zhicheng Yan
CSC420 Fall 2019
1002643142
yanzhic1

Abstract

Japanese anime plays a significant role for entertainment to spend their leisure time, it is one of the three parts of the ACG(Anime, Comic, and Game). People maybe don't recognize the anime faces who they are for the characters which in anime they have watched. Our project is to find the anime faces and to determine who are they when given an anime image.

1. Introduction

There are three major tasks for this project. The first task is to get the suitable datasets for the next two tasks. I am responsible for Dataset Collection, to find the reliable resources. My partner is responsible for dataset allocation and augmentation. Based on the dataset I have found my partner generate sufficient number of typical datasets, which make suitable for detection and recognition. The second task is to detect the faces by the given anime images, which is my main part of the project. I train the classifier by the Histogram of Gradients of the images[3](HOG) with Support Vector Machine[4](SVM), which will be described with more details in Part IV. For the third task, is to determine who the characters are, after the faces are detected, which is the one my partner is responsible for. After the face is detected, it will be analysis by a Neural Network(VGG16)[7]. Then it will determine who is the character by the given face.

2. Discoveries of Anime Face Detection

Before to identify who the character are by a given anime image. It is needed to detect and locate the faces of the anime faces. So the anime face detection is necessary and significant for anime face recognition when given a whole anime image instead of a single anime head portrait, so it is one of the major tasks of the project.

2.1. General Discovery

I have discovered that most of this type of similar problems(e.g. object detection) has the same two major goals. The first one is to scan different locations of the image. The second one is determine whether the location has this feature or not from this image. From the current course I have learned. There are several topics of concepts and algorithms may be necessary or recommended to use for anime face detection. The first one is image processing. For find the anime faces, it may be required to transform the original image to something else(e.g. grayscale, resize, etc), which make easier to identify whether it is Anime Faces. The second one is image features and matching. It is obvious that the anime face has its special feature since artists use their hands and pencils to create and draw them instead of taking a photo. It may be useful to match the patch to existing identified anime faces to determine it is also an anime face or not. The third one is the algorithm to find the region, it is essential to find a way to scan the image.

2.2. Brain Storming

I brainstormed different approaches, I think training finding the histogram of gradient(HOG) feature descriptors and **train them as classifier(mentioned as one of the options from proposal)** is the competitive way for anime faces related to other methods. Anime faces are 2D art instead of photo which projected from 3D, so the shapes and patterns of faces will show much more clearly by the given region. Other kinds of descriptor like SIFT is not suitable because it is focus on single points instead of regions.

2.3. Hypothesis

For hypothesis, I expected the method I planned can mostly successfully locate the anime faces by given anime images when the algorithm "knows" what anime faces are after "teach" it by massive amount of anime faces. And, I don't expect it to identify other things(hands, clothes, heads of non-human characters, etc) as anime faces. How to train the method by HOG feature from existing anime faces, how to detect multiple anime faces precisely from an image, and

how to scan the image are three main challenges. There are 2 major limitations may happened. First, it is not robust to anime faces which have complete different drawing style by different artists which rarely seen before. Second, it may not be robust to the anime faces which have been created very detailed textures and different direction of the faces, since it is the disadvantage of HOG feature.

3. Methods

The following are the methods for anime face detection

3.1. Histogram of Gradients(HOG)

To get the HOG as a feature of the given image[1][5]. Initially, for prepossessing, the image should be resized to a suitable size (64x128 by default). Then, we need to calculate the horizontal and vertical gradients(can be calculated either by kernel of $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}$ and $\begin{bmatrix}-1 & 0 & 1\end{bmatrix}^\top$, or from sobel by cv2 method with kernel size 1. After that, we divide the image into cells(size of 8x8 by default). The gradient of each pixel of the cell has its magnitudes and directions for the patch. So there are a total of 128 numbers(i.e. 8x8x2) for each cell if using default setting. Then we can put them into a histogram(9 number of bins, by cells), can be treated as an array of size of number of bins(As the figures[1] shown below). After that, we do the block normalization(16x16 by default) to normalize the vector 3 channel channels to remove the scale. Finally, get the final a vector by a concatenation of all the normalized histogram vectors.

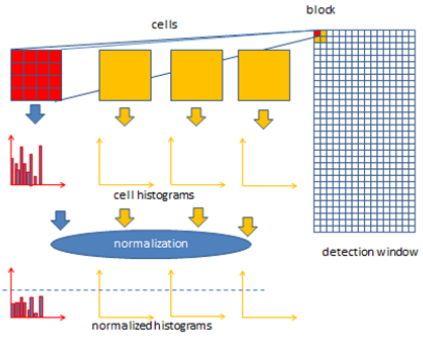

Figure 1. The figure of HOG Procedure[1]

3.2. Support Vector Machine(SVM)

After getting the HOG feature vectors from the images. We need to put them into support vector machines[4]. It is a relatively strong machine learning algorithm for binary classification problems. Choose an SVC[6] and certain number of training examples to fit the SVM. It will generate a hyperplane which separates the two classes to maximize the distance to the closest point for each class.

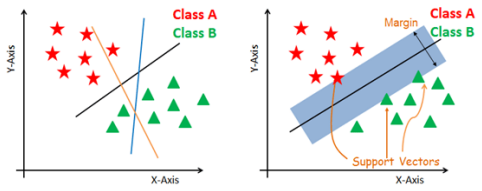

Figure 2. The figure of SVM[6]

3.3. Sliding Window Algorithm & Bounding Boxes

After train the SVM classifier. We can input the image to find the anime faces. First setting thresholds for min/max width(w), min/max height(h), size stride and sliding stride. Then by these thresholds, we can get a list of bounding boxes in the order of size(width and height) of the window and location(x and y coordinate). The tuple of (x, y, w, h) can represent a bounding box where (x, y) is the top left corner of the window. For each window, convert them to hog feature and put it to linearly classifier to predict it is anime face or not. After getting all of the satisfied window, we need to filter out the window which has high IOU(intersection over union) with at least one of the other bounding boxes of windows.

For anime face recognition, my partner uses the neural network VGG16 for multi-class classification to identify who is the character.

4. Procedures

4.1. Read and transform the datasets

1. Using cv2 to read the images from both positive training data images(anime faces) and negative training data images(not anime faces), give the limit for the number of positive data and negative data if necessary.

2. For each image, do the following in order:

   (a) Resize the image to suitable size (64x128 by default)

   (b) Convert to HOG feature descriptor with block size 16x16, cell size 8x8, and 9 number of bins(default setting)

   (c) Add the HOG feature to the dataset

   (d) If the image is from the positive training test, set "1" as label, if it is from the negative set, set "0" as label

4.2. Train and Test the classifier

1. Pick an SVC(Support Vector Classification) for the SVM model[6](i.e. choose hyperparameters), and set the number limit of positive and negative training data if needed
2. Put the X(HOG feature) and Y(labels) into the SVM model.
3. Fit the model
4. Test the model by given test images(for both negative and positive)
5. Test separately for positive and negative images from the test data images. Getting two accuracies, one for identifying Positive images, another one for identifying the Negative images.
6. Save the support vector classifier coefficients to a pkl file.

4.3. Sliding Window & Bounding Boxes

1. Load the trained support vector classifier coefficients from the pkl file.
2. Setting minimum, maximum, and stride for the window side(usually depends on the size of the original image), also the stride for sliding the image.
3. Gather the location of the top-left corner, width, and height (in default using square) of all of the windows and put them into a list, the list is ordered by width, height, x location, and y location accordingly.
4. For each window, do the following in the order:
   (a) Gather the patch from image by the given info from window
   (b) Resize the patch to suitable size (64x128 by default)
   (c) Convert to HOG feature descriptor with block size 16x16, cell size 8x8, and 9 number of bins(default setting)
   (d) Input the HOG descriptor and predict the label with the trained classifier
   (e) Save the label to a label list
5. For label list, if it is predicted as an anime image, save the index a positive index list which is for collect the windows which are determined as an anime face.
6. For each window which predicted as an anime image, do the following in order:
   (a) Calculate the intersection over union(IOU) as rectangles to other windows which is predicted as anime images
   (b) If the window has high IOU(greater than a threshold, 0.3 by default) with other rectangles, setting the label "0" in the label list
   (c) Update the positive index list by the current label list
7. For each window which is finalized what it is predicted as an anime image:
   (a) Get the location of the window
   (b) Draw the bounding box of the window by given info, may enlarge the box proportionally if it is necessary

4.4. After Detection

After detection, the size bounding box will be enlarged(usually by 2), and put to the VGG16 which has been trained by my partner to identify who the character is. I combined my implementation for detection and my partner's implementation for recognition to make the flows work sequentially.

5. Diagram of Workflow

The following is a diagram of the overall work flow of the Project

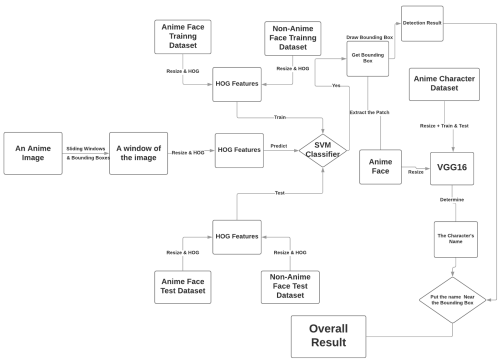

Figure 3. Diagram of Workflow

6. Results and Discussions

Figure 4 shows a flow for detection by trained SVM classifier (Original Image, get satisfied bounding boxes, and filter out bounding boxes).

Figure 4. Procedure

Figure 5. Good Examples

6.1. Good Examples

Figure 5 shows good examples. The main reason why it works perfect for the images(even the one is from taking the photo) above is because they have similar drawing style compared to the training set. Also, the background of these images are not very complex which can avoid to get the false positive bounding boxes.

6.2. Poor Examples

Figure 6 shows poor examples (First one has False Negative and second one has False Positive). The reason why it will get a false positive is because the drawing style of the image is different compared to most of the anime faces from the dataset, which makes the HOG feature not to be identified as an anime face by the SVM classifier. It also may cause false negative is because the categories of negative example should be massively vary since anything else other than the anime faces should be negative. From the training set, it may not contain feathers, the shape of feather may complex and in the false positive bounding box. The two feathers may mistakenly identify as eyes and part of the arm from the anime character may be identified as the mouth of the face.

Figure 6. Poor Examples

6.3. Overall Workflow

Figure 7 and 8 shows two examples of sequential process of anime face detection and recognition. The second image is the intermediate result(result of the detection, after resizing the original image by 0.5). The last image is the final result(enlarge the size of the bounding box by 2, and

1.4 respectively).

Figure 7. Overall Workflow Example 1

7. Main Challenges

There are three main challenges when working on the anime face detection.

7.1. The Way to Check Overall Accuracy

When working on training the SVM classifier, initially I separated the whole dataset and whole training set complete randomly by a given proportion. Firstly, I observe that the test accuracy is very close to 1. At that time I believe myself that it will be strong enough to detect anime faces. But I was wrong, then detect some other anime faces(positive examples) which is not from the original dataset, only half of them are detected. So I believe calculating accuracy by test test only it is not a good way to evaluate how the classifier performs. Then I manually separated the dataset to positive training set, positive test set, negative training set, negative test set. In the folder "dataset/detection", it has 4 sub-folders which are "positive", "positive_test", "negative", "negative_test", which are related to positive training set, positive test set, negative training set, negative test set

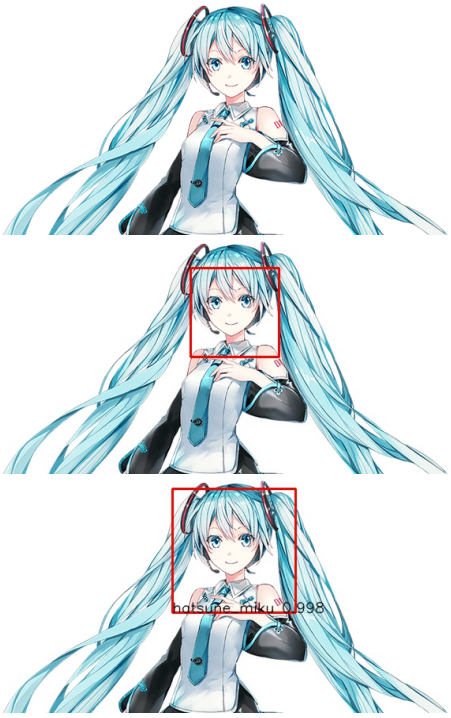

Figure 8. Overall Workflow Example 2

respectively. After that, I can check the performance of the classifier in more details, to see how many of them are false positive or false negative when testing.

7.2. Generalization

Initially, the outcome was terrible, the test accuracy of positive set was 1, and the test accuracy of the negative set was 0. I thought the size of the dataset is not large enough. Then I use extra numbers of datasets to train. The result of the test didn't change. Finally I realized that most of the patches of anime image are not anime faces. I tackle that there should be fewer positive examples than negative example. After I reduce the size of the patch, the outcome is the opposite compared to the previous one. Then I try different ratio of the positive test and negative sets, and finally train 1:7 or 1:8 as the ratio of positive test and negative sets. Which make the accuracy to around 60% for positive testing set, and very close 1 for negative testing set. For support vector classification(SVC), I initially used the default setting(radial basis function), which didn't perform good result. Finally I chose the Linear SVC to avoid the SVM to be overfit.

7.3. Bounding Boxes

It is very time consuming and it will take the whole memory for whole computer to get all of the bounding boxes of the image, I set the threshold to limit the difference win-

dow sizes and sliding strides to make the algorithm perform faster, and/or resize to smaller resolution if needed. But I need to tune them manually, Figure 9 is an example shows the incorrect setting of the bounding box threshold(the max limit of the max box size is too small) for a higher resolution image. Another thing which need to tune manually is the scale of enlarging the bounding boxes after detection and before recognition. If the scale is too big or too small, it will be identified as a wrong character(Figure 10).

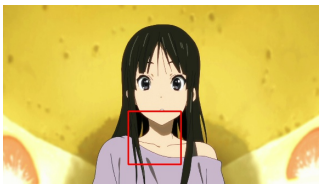

Figure 9. Max box size is too small, or the resolution is too high

Figure 10. Wrong character (she is not Hatsune Miku), the bounding box is not big enough for recognition

8. Conclusion & Future Work

Overall, for our project first we collected and augmented our datasets. We use anime faces and non-anime faces by SVM after converting to the HOG Feature for anime face detection. And we use anime faces of different characters to train VGG16 to identify who they are. And then given an anime image, the SVM+HOG with sliding window and bounding boxes will locate the anime faces, after that will the bounding box will pass to VGG16 to identify who they are. The current limitations of the anime face detection are relatively lower accuracy and the slowness of finding the bounding box(currently need to manually set the limit). Also, it is not resolution invariant We still need to improve the accuracy and speed by trying some different method in the future, The SVM is still relatively underfitting and less strong than some other powerful classifiers like convolutional neural networks(CNN). Also the size HOG descriptor is relatively large and numeric, which make the calculation slower. Vectorized Binary descriptor(Local Binary Pattern[8]) may perform faster compared to the numeric one. Region of Interest Pooling(RoI) is another good option, it can locate things more directly instead. Faster-RCNN[2] may be a good choice since it combines the binary classification and find the location together to a single large neural network.

Datasets

1. Anime Faces
`https://www.kaggle.com/soumikrakshit/anime-faces`
2. Anime Face Dataset
`https://www.kaggle.com/splcher/animefacedataset`
3. Safebooru - Anime Image Metadata
`https://www.kaggle.com/alamson/safebooru`
4. Tagged Anime Illustrations
`https://www.kaggle.com/mylesoneill/tagged-anime-illustrations`

References

[1] *Histogram of Oriented Gradients (HOG) Descriptor*. 18-Nov-2019,[Online]. Available: `https://software.intel.com/en-us/ipp-dev-reference-histogram-of-oriented-gradients-hog-descriptor`.

[2] X. Chen and A. Gupta. *An Implementation of Faster RCNN with Study for Region Sampling*. 07-Feb-2017.[Online].Available: `https://arxiv.org/pdf/1702.02138.pdf`.

[3] N. Dalal and B. Triggs. *Histograms of Oriented Gradients for Human Detection*. 2005, [Online] Available: `https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf`.

[4] T. Fletcher. *Support Vector Machines Explained*. 23-Dec-2008. [Online]. Available: `https://cling.csd.uwo.ca/cs860/papers/SVM_Explained.pdf`.

[5] S. Mallick. *Histogram of Oriented Gradients*. 06-Dec-2016. [Online]. Available: `https://www.learnopencv.com/histogram-of-oriented-gradients/`.

[6] A. Navlani. *Support Vector Machines in Scikit-learn*. 12-Jul-2018.[Online]. Available: `https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python`.

[7] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 10-Apr-2015.[Online].Available: `https://arxiv.org/pdf/1409.1556.pdf`.

[8] H. Wang, J. Hu, and W. Deng. *Face Feature Extraction: A Complete Review*. 09-Mar-2018.[Online]. Available: `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8225635`.