Dexter Detector
Detection of Faces in the Dexter Cartoon

Ilinca-Ana Moraru

March 13, 2025

1 Introduction

This project aims to detect faces of the characters in the Dexter Cartoon. The first model has the scope of detecting all faces of the images it receives, with no regard of the characters names. The next four models are build for the four main characters: Dexter, Deedee, Mom and Dad and should only detect their faces, one model for each of them.

2 The General Approach

The models are using greycale Hog descriptors of 36x36 pixels with hog cells of 6X6 pixels. I used positive examples, negative examples and hard negative examples - which were detections the models made that were false positives. The positive examples received the value of 1 whereas the rest had a 0 value. After resizing them to 36x36 pixels, they were put into linear SVM's of varying Cs. The model that was able to separate the data the best was the one saved and used. The models iterates through different image resizings in order to find all possible faces, big, small, narrow or wide. Next, non-maximal suppression is performed, in order to avoid having overlapping detections, as a single face would be detected in multiple windows. The windows that result from this filtering get a score and the model is evaluated using the precision - recall area metric.

3 Data Gathering

3.1 All Faces Model

I constructed my dataset on the train data I received which contained 1000 different images for each of the 4 characters, in which that character was present. Other characters, known or unknown could be present. All faces were annotated. I started by gathering all the faces for each of the four main

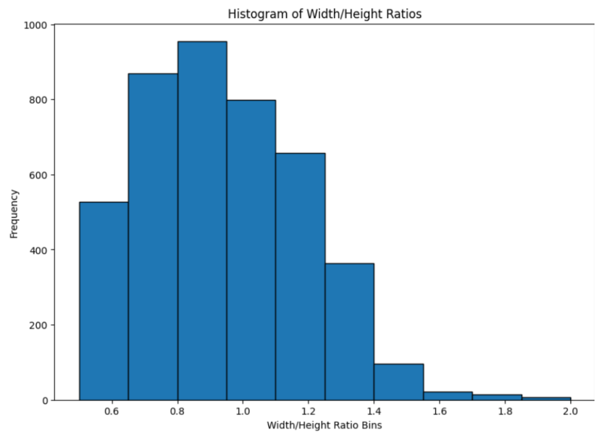

Figure 1: Distribution of face ratios on all faces

characters and all unique faces of the rest of the characters, as the same frame would appear in several images of the dataset, resulting in about 4 5000 positive examples. I flipped them left to right to gain more data, resulting in 9 000 examples. Negative examples were chosen randomly, but I used the annotations to make sure that they did not intersect with a face. I chose 40 000 examples, 10 000 each of the sizes: 110X100, 80X80, 60X60, 40X40. After finding the best 10 ratios for detecting faces, I ran the model and saved the false positives. I ordered them in descending order by detector's score and took the top 10 000 from each character dataset as hard negatives, therefore I used 40 000 hard negative examples.

3.2 Character Models

The datasets were constructed on the train data I received. The positive examples were all 1000 faces of the character and negative examples were chosen in the same manner as the model for all faces, except I added all faces of other characters, known or unknown. I have about 50 000 negative examples. I constructed the hard negative examples in the same manner as for the All Faces Model, selecting about 3 000 examples. The ratios chosen experimentally as either the same as in the previous model either the most common for the character.

4 Tehnical approach

I used a class named "FacialDetector" that gets a new instance for every model that is ran. This class has several methods: "get_positive_descriptors", "get_negative_descriptors", "get_hard_mining_descriptors" - in which the model recieves the extracted images from directories: "exemplePozitive", "exempleNegative", "exempleHardMining" and calculates their hog descriptors to be used in training. These descriptors are saved so that the run time is reduced for future runs where the same examples are used.

Method "train_classifier" trains the model using a linear SVM with varying Cs: 0.00001, 0.0001, 0.001, 0.01, 0.1, 1 and saves the model that separates the data the best for the current and future runs.

"intersection_over_union" calculates the overlap of 2 detections by dividing the area of intersection to the sum of the detections areas minus the area of intersection. The detection of the bigger area gets normalized to the area of the smaller one, so that smaller detections are not appearing over the bigger ones. I found that by doing so, my models score before adding hard negative examples doubled.

Function "non_maximal_suppression" receives all the detections of an image and returns only the detections that are overlapping less than a percent with over valid detections. In case 2 detections overlap more than the threshold, the one with the bigger score gets saved. I chose the threshold at 30 %. Also, for the models that target only specific characters, I saved only the detections that have a bigger score than of the biggest one divided by 2, as I found that the biggest detection score is very likely to correspond to be the best detection.

The most important method, named "run" iterates through images and uses a sliding window, fitting the cropped images of that window in the SVM and returning a score of confidence. The model takes the images one by by one and iterates through 20 resizings equally spaced between 0 and 1 of the image so that all faces have a chance of being detected, no matter how big or small they appear in the image. In the size iteration, it is also performed a ratio iteration, where the entire image is warped to different ratios, based on the positive examples ratio distribution. The ratio is the inverse of the ratio of the faces I wish to capture and warp them to 36X36. In this manner, the windows will capture all sizes and ratios I found work best. I chose this approach, so that even if the model runs on a faces of a characters that has a different width/height ratio than what is typical for them, the face would still resemble the usual faces after warping. Also, this decision makes possible having a single SVM model that can be more precised than several that would receive less data in training. After calculating the detections, ""non_maximal_suppression" function is called and the resulting detections are saved.

Method "compute_average_precision" calculates the score of the model based on the precision and recall received as parameters and gets called in "eval_detections". Here, based on the ground truth, the detections that are overlapping with the ground truth more than 30% are counted as true positives and the rest as false positives.

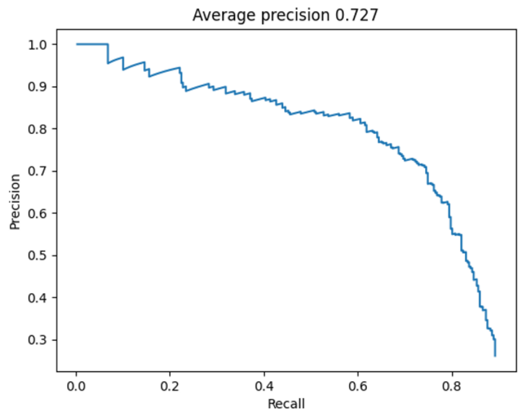

Figure 2: Precision-Recall Metric on All Faces

5 Conclusions

I really enjoyed working on this project, as I was motivated by the metric of the models and aimed to get a bigger score. This project taught me a lot about Hog Descriptors and the flow of machine learning project. I was surprised of how much time it took for extracting hard negative examples and for calculating the detections. If I were to start over again, I would aim keep my focus on the bigger picture and not get so distracted with small, minor adjustments. Overall, I found working on this project an enjoyable way of learning about HOG Descriptors and I am glad to have gained hands-on experience on this topic.