CAVA-TEMA2

Huțan Mihai-Alexandru Gr.343

January 2024

Contents

1 Introduction

This homework revolves around three main steps: patch extraction, classification, and detection. These steps are comprehensively explained in the following sections.

2 Patch Extraction

2.1 Approach

Upon analyzing the ground truth detection, it was observed that faces are mostly detected as squares, typically with a width and height of 80px. To simplify the implementation, I opted for 40px patches, aiming to downscale the images efficiently during implementation using a sliding window approach.

2.2 Positives

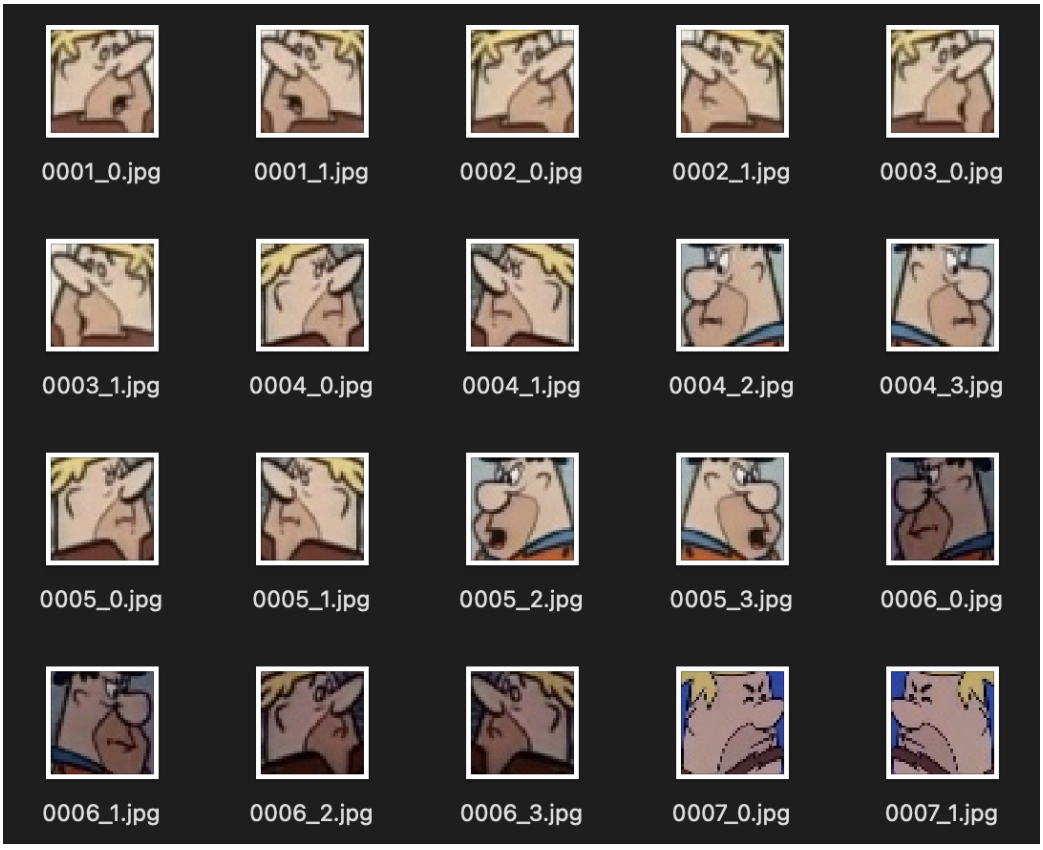

Figure 1: positives

The process for handling positives is straightforward: parsing the ground truth annotations and images to extract patches. These patches are resized to 40x40px, and both the original and flipped variants are saved for additional data. I experimented with a minor tweak ignoring small detections with an area of less than 400 in the original image but it didn't yield significant improvements.

2.3 Negatives

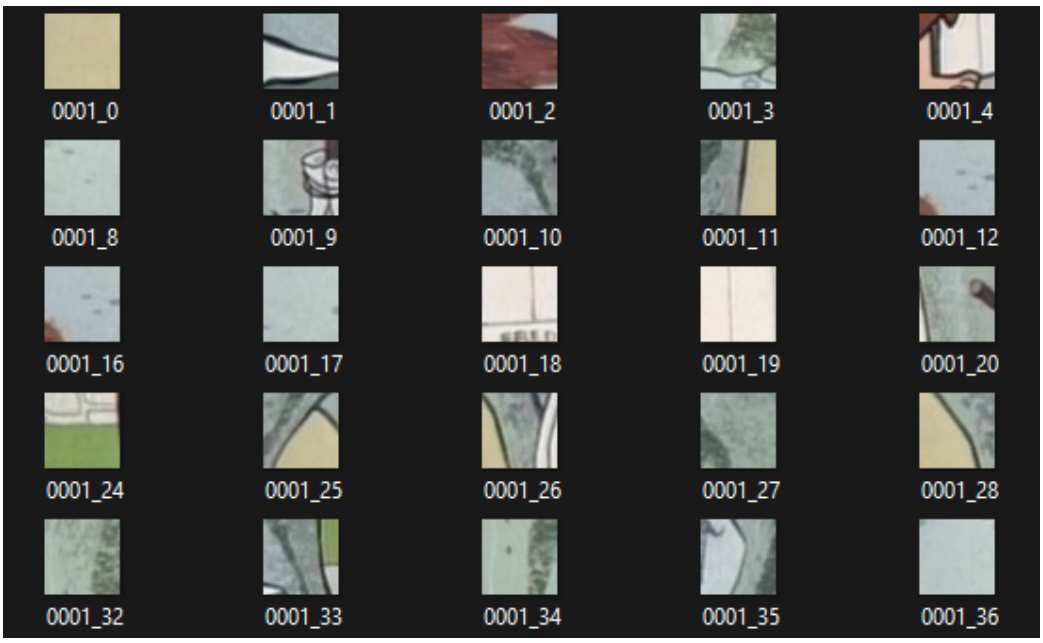

Figure 2: negatives

For negative patches, I pursued a random approach, generating random xmin and ymin coordinates and calculating xmax and ymax by adding 40. To ensure non-overlap with any ground truth detection, strict avoidance measures were taken.

Additionally, I saved 50 negatives for each ground truth detection, which is further detailed in the binary classification section.

3 Classification

3.1 Approach

For both tasks, Convolutional Neural Networks (CNNs) were chosen. Initially, attempts with hog descriptors didn't exceed an average precision of 0.550 for task1. Task1 employed a binary classifier elaborated in detail here, while task2 utilized a multi-class classifier described here.

3.2 Binary Classifier

The binary classifier involved minimal preprocessing: converting patches to grayscale images. This choice aimed to avoid potential color-based false positives and encourage the network to focus more on lines. Normalizing pixel values between 0 and 1 was also performed by dividing each by 255.

Initial attempts with a small CNN yielded good validation results (more than 0.99 accuracy), but practical application with the sliding window revealed numerous false positives. To address this, the network was intentionally overfitted on the training images, leveraging the nearly identical appearance of characters. This overfitting was achieved by employing a considerably more complex CNN architecture and providing over 300k negative examples to counter the data imbalance. This deliberate bias towards classifying as negative aligned with the sliding window approach where most sent images are negatives.

```
model = Sequential(
nn.Conv2d(in_channels=1, out_channels=16,
    kernel_size=3, padding=1),
nn.ReLU(),
nn.MaxPool2d(2, 2),
nn.Conv2d(in_channels=16, out_channels=32,
    kernel_size=3, padding=1),
nn.ReLU(),
nn.MaxPool2d(2, 2),
```

```
nn.Conv2d(in_channels=32, out_channels=64,
    kernel_size=3, padding=1),
nn.ReLU(),
nn.MaxPool2d(2, 2),
nn.Conv2d(in_channels=64, out_channels=64,
    kernel_size=3, padding=1),
nn.ReLU(),
nn.MaxPool2d(2, 2),
nn.Flatten(),
nn.Dropout(0.25),
nn.Linear(256, 256),
nn.ReLU(),
nn.Dropout(0.25),
nn.Linear(256, 1),
nn.Sigmoid())
```

The network achieved over 0.99 accuracy on both test and validation sets. Optimizer: Adam; learning rate: 1e-4; loss function: binary cross-entropy; batch size: 64.

3.3 Multi-Class Classifier

Preprocessing involved converting patches from BGR to RGB, and normalizing pixels between 0 and 1. Ground truth labels were mapped to integers, resulting in five classes. The architecture for the multi-class classifier is almost the same as the binary classifier, for similar reasons, forced overfitting. The only change being that we use 3 channels images (RGB) and changed the sigmoid activation function to softmax.

```
model = Sequential(
nn.Conv2d(in_channels=3, out_channels=16,
    kernel_size=3, padding=1),
nn.ReLU(),
nn.MaxPool2d(2, 2),
nn.Conv2d(in_channels=16, out_channels=32,
    kernel_size=3, padding=1),
```

```
nn.ReLU(),
nn.MaxPool2d(2, 2),
nn.Conv2d(in_channels=32, out_channels=64,
    kernel_size=3, padding=1),
nn.ReLU(),
nn.MaxPool2d(2, 2),
nn.Conv2d(in_channels=64, out_channels=64,
    kernel_size=3, padding=1),
nn.ReLU(),
nn.MaxPool2d(2, 2),
nn.Flatten(),
nn.Dropout(0.25),
nn.Linear(256, 256),
nn.ReLU(),
nn.Dropout(0.25),
nn.Linear(256, 5),
nn.Softmax(1))
```

The network achieved over 0.99 accuracy on both test and validation sets. Optimizer: Adam; learning rate: 1e-4; loss function: cross-entropy loss; batch size: 64.

4 Detection

4.1 Sliding Window

The sliding window employed a 40x40px window with a stride of 2 and various downscale factors (0.9, 0.5, and 0.3) to accommodate faces of different sizes. Post-cnn processing included applying a threshold of 0.9 for predictions and non-maximal suppression (0.1 IoU threshold) to finalize confident detections. Task2 utilized results from Task1, passing detections to the multi-class CNN classifier for character classification.