

Introduction to Artificial Intelligence HW 1 Report

110550088 李杰穎

March 16, 2022

1 Code and Explanation

Code 1: Part 1 (`datasets.py`)

```
1 import os
2 import cv2
3 import numpy as np
4 def loadImages(dataPath):
5     """
6     Load all Images in the folder and transfer a List of tuples.
7     The first element is the numpy array of shape (m, n) representing the image.
8     (remember to resize and convert the parking space images to 36 x 16 grayscale
9     → images.)
10    The second element is its classification (1 or 0)
11    Parameters:
12    dataPath: The folder path.
13    Returns:
14    dataset: The List of tuples.
15    """
16    # Begin your code (Part 1)
17    dataset = [] # Declare an empty list to save the grayscale images
```

```

17
18 # Process images in "car" directory
19 for item in os.listdir(os.path.join(dataPath, "car")): # Use os.path.join to
    ↳ generate paths
20     img = cv2.imread(os.path.join(dataPath, "car", item)) # Read image from files
21     img = cv2.resize(img, (36, 16)) # Resize the image from (360, 160) to (36,
    ↳ 16)
22     img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # Convert image to grayscale
    ↳ image
23     data = (img, 1) # Create a tuple to store image and label and
24     # because all images in "car" folder is the occupied parking space, the label
    ↳ is set to 1
25     dataset.append(data) # Append the tuple to the dataset list
26
27 for item in os.listdir(os.path.join(dataPath, "non-car")): # Do the same thing as
    ↳ above but this time is for "non-car" folder
28     img = cv2.imread(os.path.join(dataPath, "non-car", item))
29     img = cv2.resize(img, (36, 16))
30     img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
31     data = (img, 0) # Not occupied parking space, label is set to 0
32     dataset.append(data)
33 # End your code (Part 1)
34
35 return dataset

```

Code 2: Part 2 (adaboost.py/selectBest)

```

1 def selectBest(self, featureVals, iis, labels, features, weights):
2     """
3     Finds the appropriate weak classifier for each feature.
4     Selects the best weak classifier for the given weights.
5     Parameters:

```

```

6     featureVals: A numpy array of shape (len(features), len(dataset)).
7         Each row represents the values of a single feature for each training sample.
8     iis: A list of numpy array with shape (m, n) representing the integral images.
9     labels: A list of integer.
10         The ith element is the classification of the ith training sample.
11     features: A numpy array of HaarFeature class.
12     weights: A numpy array with shape(len(dataset)).
13         The ith element is the weight assigned to the ith training sample.
14     Returns:
15     bestClf: The best WeakClassifier Class
16     bestError: The error of the best classifier
17 """
18 # Begin your code (Part 2)
19 # Init. WeakClassifier by the feature in features list
20 # And append them all into the clfs list
21 clfs = [WeakClassifier(feature=feature) for feature in features]
22
23 # Declare bestClf and bestError to store the currently best classifier and its error
24 bestClf = None
25 bestError = sum(weights) # The max error is the sum of weights
26
27 for clf in clfs: # Iterate all classifier in clfs
28     error = 0 # Declare a variable to track the error of the current clf
29     for i in range(len(iis)): # Iterate all image sample
30         if clf.classify(iis[i]) != labels[i]: # When the prediction of the model is
31             ↪ different with the lable
32             error += weights[i] # Add weights to error
33     if error < bestError: # If the error is smaller than the best error, then
34         ↪ classifier is the currently best classifier
35         bestError = error # Change bestError to current error
36         bestClf = clf # Save this classifier as bestClf

```

```

36 # End your code (Part 2)
37 return bestClf, bestError

```

Code 3: Part 4 (detection.py/detect)

```

1  def detect(dataPath, clf, t=10):
2      """
3      Please read detectData.txt to understand the format.
4      Use cv2.VideoCapture() to load the video.gif.
5      Use crop() to crop each frame (frame size = 1280 x 800) of video to get parking
6      ↪ space images. (image size = 360 x 160)
7      Convert each parking space image into 36 x 16 and grayscale.
8      Use clf.classify() function to detect car, If the result is True, draw the green
9      ↪ box on the image like the example provided on the spec.
10     Then, you have to show the first frame with the bounding boxes in your report.
11     Save the predictions as .txt file (Adaboost_pred.txt), the format is the same as
12     ↪ GroundTruth.txt.
13     (in order to draw the plot in Yolov5_sample_code.ipynb)
14
15     Parameters:
16         dataPath: the path of detectData.txt
17
18     Returns:
19         No returns.
20     """
21     # Begin your code (Part 4)
22     cords = []
23     with open(dataPath) as file:
24         num_of_parking = int(file.readline())
25         for _ in range(num_of_parking):
26             tmp = file.readline()
27             tmp = tmp.split(" ")
28             res = tuple(map(int, tmp))

```

```

25         cords.append(res)
26     cap = cv2.VideoCapture(os.path.join(dataPath, "..", "video.gif"))
27     frame = 0
28     output_gif = []
29     while True:
30         detect_label = []
31         frame += 1
32         print(f"frame: {frame}")
33         _, img = cap.read()
34         if img is None:
35             break
36         for cord in cords:
37             pic = crop(*cord, img)
38             pic = cv2.resize(pic, (36, 16))
39             pic = cv2.cvtColor(pic, cv2.COLOR_RGB2GRAY)
40             detect_label.append(clf.classify(pic))
41         for i, label in enumerate(detect_label):
42             if label:
43                 pos = [[cords[i][idx], cords[i][idx+1]] for idx in range(0, 8, 2)]
44                 pos[2], pos[3] = pos[3], pos[2]
45                 pos = np.array(pos, np.int32)
46                 cv2.polylines(img, [pos], color=(0, 255, 0), isClosed=True)
47
48         output_gif.append(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
49     imageio.mimsave(f'results_{t}.gif', output_gif, fps=2)
50     # End your code (Part 4)

```

Code 4: Part 5 (yolov5_sample_code.ipynb)

```

1 dataPath = os.path.join("HW1_material", "detect", "detectData.txt")
2 cords = []
3 with open(dataPath) as file:

```

```

4     num_of_parking = int(file.readline())
5     for _ in range(num_of_parking):
6         tmp = file.readline()
7         tmp = tmp.split(" ")
8         res = tuple(map(int, tmp))
9         cords.append(res)
10    cap = cv2.VideoCapture(os.path.join("HW1_material", "detect", "video.gif"))
11    frame = 0
12    output_gif = []
13    first_frame = True
14    while True:
15        detect_label = []
16        frame += 1
17        print(f"frame: {frame}")
18        _, img = cap.read()
19        if img is None:
20            break
21        for cord in cords:
22            pic = crop(*cord, img)
23            pic = cv2.resize(pic, (36, 16))
24            detect_label.append(yolov5_func.classify(pic, weight_path,
25                                                    ↪ confidence_threshold, (36, 16)))
26        for i, label in enumerate(detect_label):
27            if label:
28                pos = [[cords[i][idx], cords[i][idx+1]] for idx in range(0, 8, 2)]
29                pos[2], pos[3] = pos[3], pos[2]
30                pos = np.array(pos, np.int32)
31                cv2.polylines(img, [pos], color=(0, 255, 0), isClosed=True)
32        if first_frame:
33            cv2.imwrite(img_save_path, img)
34            first_frame = False
35    output_gif.append(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))

```

```

35     with open(txt_save_path, "a") as txt:
36         res = ""
37         for i, label in enumerate(detect_label):
38             if label:
39                 res += "1"
40             else:
41                 res += "0"
42             if i != len(detect_label) - 1:
43                 res += " "
44             else:
45                 res += "\n"
46         txt.write(res)
47 imageio.mimsave(f'results.gif', output_gif, fps=2)

```

2 Results

2.1 Measurements

2.1.1 Accuracy

In binary classification problem, we often define four kinds of accuracy, which are True Positive, True Negative, False Positive and False Negative to show the performance of classifiers. Their definitions are written below:

- True Positive (TP): The real label is **true**, and our model predicts that it's **true**.
- True Negative (TN): The real label is **false**, and our model predicts that it's **false**.
- False Positive (FP): The real label is **true**, but our model predicts that it's **false**.
- False Negative (FN): The real label is **false**, but our model predicts that it's **true**.

2.1.2 F-Score

F-score is used to measure a test's accuracy. To define F-score, we first need to define two variables, "precision" and "recall".

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (1)$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (2)$$

And the definition of F-score is:

$$\text{F-score} = \frac{(1 + \beta^2) \text{precision} \times \text{recall}}{\beta^2 \text{precision} + \text{recall}} \quad (3)$$

In reality, we often use "F1-score", which means $\beta = 1$. So the "F1-score" can be written as:

$$\text{F1-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (4)$$

The range of F1-score is $[0, 1]$. If F1-score is higher (closer to 1), then the performance of classifier is better.

3 Problems

3.1 `alpha = math.log(1.0/beta)` ValueError: math domain error

3.2 The model has bad performance