Localizarea si Detectia Personajelor din Familia Flintstone

Ionescu Radu-Constantin, grupa 334

1. Tema Proiectului

Tema acestui proiect de vedere artificiala a fost localizarea si detectarea faciala pentru personajele din desenul animat "Familia Flinstone". Atat localizarea cat si detectarea faciala sunt descrise prin bounding boxes care incadreaza cat mai bine cu putinta fetele personajelor. In plus, pentru cele 4 personaje principale, se clasifica o fata ca fiind sau nu a unuia dintre ele

2. Strategia de Rezolvare

I: Punctul de plecare

Solutia se bazeaza puternic pe codul de la laborator, principalele functionalitati (construirea descriptorilor HOG, antrenarea clasificatorilor, glisarea ferestrelor peste imagine, suprimarea non-maximelor, evaluarea, vizualizarea, etc) fiind refolosite, cu mici modificari

II: Obtinerea exemplelor pozitive si negative

Spre deosebire de exemplul din laborator, exemplele pozitive si negative au trebuit extrase din pozele de antrenare si fake test. In acest sens, pentru exemplele pozitive am decupat fetele din imagini conform adnotarilor si pentru imaginile negative alegem coordonate random pentru ferestre patratice din imagini si alegem doar ferestrele care nu se suprapun cu un bounding box din imagine. De mentionat este ca, pentru a pregati pasul 2, specificam pentru fiecare imagine rezultata cine este personajul real din imagine (adica cel din adnotare)

```
labeled_images_dir = 'antrenare'
charcters = ['barney', 'betty', 'fred', 'wilma']

for character in charcters:
    character_images_dir = os.path.join(labeled_images_dir, character)
    character_annotations_path = os.path.join(labeled_images_dir, f"{character}_annotations.txt")

    with open(character_annotations_path, 'r') as f:
        content = f.readlines()

    content = [x.strip() for x in content]

    for file in os.listdir(character_images_dir):
        image = cv.imread(os.path.join(character_images_dir, file))

        annotations = [x for x in content if x.startswith(file)]

        for i, annotation in enumerate(annotations):
            annotation = annotation.split(' ')

            real_character = character
            if annotation[-1] != character:
                real_character = annotation[-1]

            bbox = [int(x) for x in annotation[1:-1]]

            cut_image = image[bbox[1]:bbox[3], bbox[0]:bbox[2]]

            cut_image_path = os.path.join(positive_images_path, f"{real_character}_{file}_{i}0.jpg")

            cut_image = cv.resize(cut_image, (64, 64))

            cv.imwrite(cut_image_path, cut_image)
```

III: Descriptorii HOG

In urma experimentelor, am observat ca cea mai buna performanta este obtinuta pe descriptori HOG pe imagini de 64 x 64 de pixeli, cu fiecare celula hog avand latura de 8 pixeli. In plus, pe baza indicatiei din curs, am folosit descriptori HOG RGB, practic descriptori HOG pentru fiecare dintre cele 3 canale ale unei imagini color care apoi se concateneaza

```
# set the parameters
self.dim_window = 64  # exemplele pozitive (fete de oameni cropate) au 36x36 pixeli
self.dim_hog_cell = 8  # dimensiunea celulei
self.dim_descriptor_cell = 64  # dimensiunea descriptorului unei celule
self.number_positive_examples = len(os.listdir(self.dir_pos_examples))  # numarul exemplelor pozitive
self.number_negative_examples = 10000  # numarul exemplelor negative
self.overlap = 0.3
self.has_annotations = False
self.threshold = 0
```

```
hog_descriptors = []
for channel in cv.split(img):
    features = hog(channel, pixels_per_cell=(self.params.dim_hog_cell, self.params.dim_hog_cell), cells_per_block=(2, 2), feature_vector=False)
    hog_descriptors.append(features)

num_cols = img.shape[1] // self.params.dim_hog_cell - 1

num_rows = img.shape[0] // self.params.dim_hog_cell - 1

num_cell_in_template = self.params.dim_window // self.params.dim_hog_cell - 1

for y in range(0, num_rows - num_cell_in_template):
    for x in range(0, num_cols - num_cell_in_template):
        descr_red = hog_descriptors[0][y:y + num_cell_in_template, x:x + num_cell_in_template].flatten()
        descr_green = hog_descriptors[1][y:y + num_cell_in_template, x:x + num_cell_in_template].flatten()
        descr_blue = hog_descriptors[2][y:y + num_cell_in_template, x:x + num_cell_in_template].flatten()
        descr = np.concatenate((descr_red, descr_green, descr_blue))
```

IV: Ferestrele glisante – trucul redimensionarii imaginii

Pentru a simula fereastra glisanta de diferite dimensiuni, pentru a nu intra in probleme legate de dimensiunea descriptorilor HOG variabila in functie de dimensiunea ferestrei, am ales sa redimensionez imaginea cu difersi coeficienti pentru a testa ferestre de dimeniuni diferite. Ulterior se tine cont de acesti coeficienti pentru a calcula coordonatele bounding box-ului din imaginea de dimeniuni originale

```
for resize_quoefficient in np.arange(0.1, 2.1, 0.1):
    img = cv.resize(img_original, (0, 0), fx=resize_quoefficient, fy=resize_quoefficient)
```

V: Identificarea personajelor

Task-ul 2 a fost rezolvat intr-o maniera similara cu primul. Pentru fiecare dintre cele 4 personaje antrenam un clasificator care decide daca o regiune dintr-o imagine corespunde sau nu unui anumit personaj, cu un anumit scor de incredere. Daca pragul de incredere minim (scor 0) nu este atins pentru nici macar unul dintre cele 4 personaje, detectia este clasificata ca “unknown”. De mentionat este ca in selectarea datelor de antrenare pentru fiecare clasificator, exemplele pozitive sunt, evident, descriptorii fetelor persoanjului respectiv, in timp ce exemplele negative sunt exemplele negative generale (descriptorii imaginilor care nu contin fete) alaturi de descriptorii fetelor celorlalte personaje

```
# Pasul 6. Antrenam cate un clasificator pentru fiecare personaj
for character in characters[:-1]:
    other_characters_descriptors = np.empty((0, characters_pozitive_features[character].shape[1]))

    for other_character in characters:
        if other_character != character:
            other_characters_descriptors = np.concatenate((other_characters_descriptors, characters_pozitive_features[other_character]), axis=0)

    negative_features = np.concatenate((general_negative_features, other_characters_descriptors), axis=0)
```

```python
def train_classifier(self, training_examples, train_labels, has_to_train=False, character=''):
    svm_file_name = os.path.join(self.params.dir_save_files, 'best_model_%d_%d_%d%s' %
                                 (self.params.dim_hog_cell, self.params.number_negative_examples,
                                  self.params.number_positive_examples, character))

    if os.path.exists(svm_file_name) and has_to_train == False:
        if character == '':
            self.best_model = pickle.load(open(svm_file_name, 'rb'))
        else:
            self.best_models[character] = pickle.load(open(svm_file_name, 'rb'))
        return
```

```python
score = np.dot(descr, w)[0] + bias
if score > self.params.threshold:
    x_min = int((x * self.params.dim_hog_cell) / resize_quoefficient)
    y_min = int((y * self.params.dim_hog_cell) / resize_quoefficient)
    x_max = int(((x * self.params.dim_hog_cell) + self.params.dim_window) / resize_quoefficient)
    y_max = int(((y * self.params.dim_hog_cell) + self.params.dim_window) / resize_quoefficient)

    most_probable_character = ('unknown', 0)

    for character in characters:
        character_score = np.dot(descr, characters_weights_and_biases[character]['w'])[0] + characters_weights_and_biases[character]['bias']

        if character_score > most_probable_character[1]:
            most_probable_character = (character, character_score)

    image_detections.append([x_min, y_min, x_max, y_max])
    image_scores.append(score)
    image_character_detections.append(most_probable_character[0])
    image_character_scores.append(most_probable_character[1])
```

VI: Hard Mining – antrenarea cu exemple puternic negative

Imbunatatirea preciziei medii de la taskul 1 conduce automat la incadrari mai bune ale fetelor pentru taskul 2, deci si scoruri mai bune (datele de test seamana mai mult cu datele de antrenare). Astfel, am implementat paradigma de antrenare cu exemple puternic negative. La fiecare rulare, printr-un parametru, se decide daca misclasificarile fals pozitive se doresc sau nu sa fie salvate ca descriptori puternic negativi pentru urmatoarea iteratie. Dupa introducerea de noi exemple puternic negative, clasificatorul se reantreneaza tinand cont de noile date

```python
params.use_hard_mining = True  # (optional)antrenare cu exemple puternic negative
params.provide_hard_negative_detections = False  # retinem sau nu detectiile fals pozitive pentru antrenarea cu exemple puternic negative
params.use_flip_images = True  # adauga imaginile cu fete oglindite
```

```python
# Pasul 10. Daca se doreste, salvam detectiile false pozitive pentru hard mining pentru localizare(task 1)
if params.provide_hard_negative_detections:
    if false_positive_files is None or false_pozitive_boxes is None:
        pass
    else:
        if not os.path.exists(params.dir_hard_negatives):
            os.makedirs(params.dir_hard_negatives)
            print('directory created: {} '.format(params.dir_hard_negatives))
        else:
            print('directory {} exists '.format(params.dir_hard_negatives))
            for file in os.listdir(params.dir_hard_negatives):
                os.remove(os.path.join(params.dir_hard_negatives, file))

            number_false_positive_files = len(false_positive_files)

            for i in range(number_false_positive_files):
                image = cv.imread(os.path.join(params.dir_test_examples, false_positive_files[i]))

                bbox = false_pozitive_boxes[i]

                cut_image = image[bbox[1]:bbox[3], bbox[0]:bbox[2]]

                cut_image = cv.resize(cut_image, (64, 64))

                cv.imwrite(os.path.join(params.dir_hard_negatives, str(i) + '.jpg'), cut_image)

            print('negative images from test generated')

            hard_negative_features_path = os.path.join(params.dir_save_files, 'new_descriptoriExemplePuternicNegative_' + str(params.dim_hog_cell) + '_' +
                                                       str(number_false_positive_files) +'.npy')

            if os.path.exists(hard_negative_features_path):
                pass
            else:
                hard_negative_features = facial_detector.get_hard_negative_descriptors()
                np.save(hard_negative_features_path, hard_negative_features)
                print('Am salvat descriptorii pentru exemplele puternic negative in fisierul %s' % hard_negative_features_path)

            for file in os.listdir(params.dir_hard_negatives):
                os.remove(os.path.join(params.dir_hard_negatives, file))
```

```python
has_to_train = False
if params.use_hard_mining:
    directory = params.dir_save_files
    prefix = 'descriptoriExemplePuternicNegative'

    files = [file for file in os.listdir(directory) if file.startswith(prefix)]

    num_cols = negative_features.shape[1] if negative_features.size else 0

    hard_negative_features = np.empty((0, num_cols))

    for file in files:
        path = os.path.join(directory, file)
        hard_negative_features = np.concatenate((hard_negative_features, np.load(path)), axis=0)

    prefix = 'new_descriptoriExemplePuternicNegative'

    files = [file for file in os.listdir(directory) if file.startswith(prefix)]

    if len(files) > 0:
        has_to_train = True

    for file in files:
        path = os.path.join(directory, file)
        hard_negative_features = np.concatenate((hard_negative_features, np.load(path)), axis=0)

        non_new_file = file.replace('new_', '')
        os.rename(os.path.join(directory, file), os.path.join(directory, non_new_file))

    negative_features = np.concatenate((negative_features, hard_negative_features), axis=0)
```

3. Rezultate

In urma aplicarii solutiei pe datele de test (dureaza aproximativ 30 de minute, 7-9 secunde per imagine pentru rezultate bune intr-un timp rezonabil) s-au obtinut urmatoarele scoruri de Average Precision (AP)

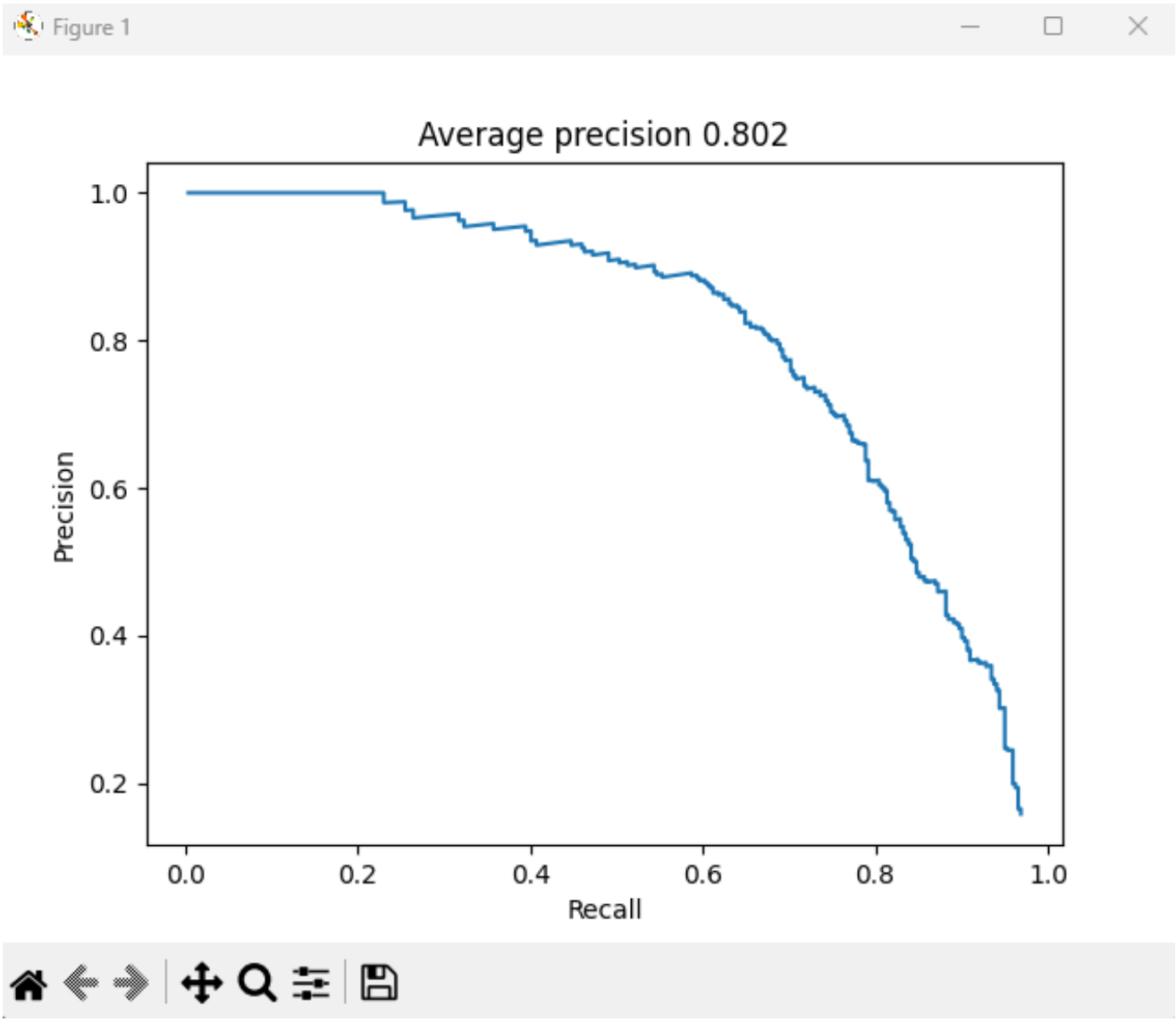


Figure 1
Average precision 0.802
1.0
0.8
0.6
0.4
0.2
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall

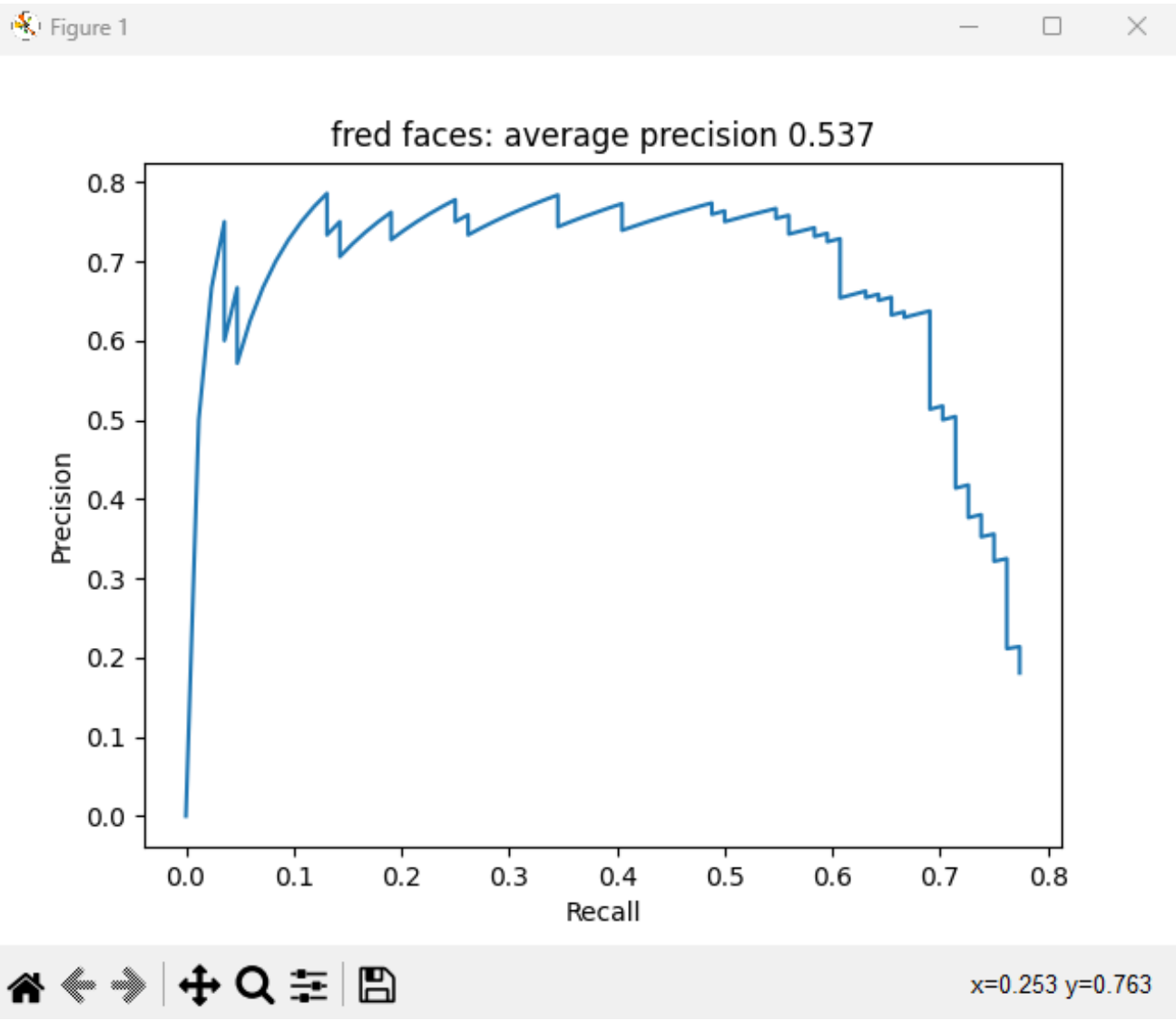


Figure 1
fred faces: average precision 0.537
Precision
Recall
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x=0.253 y=0.763

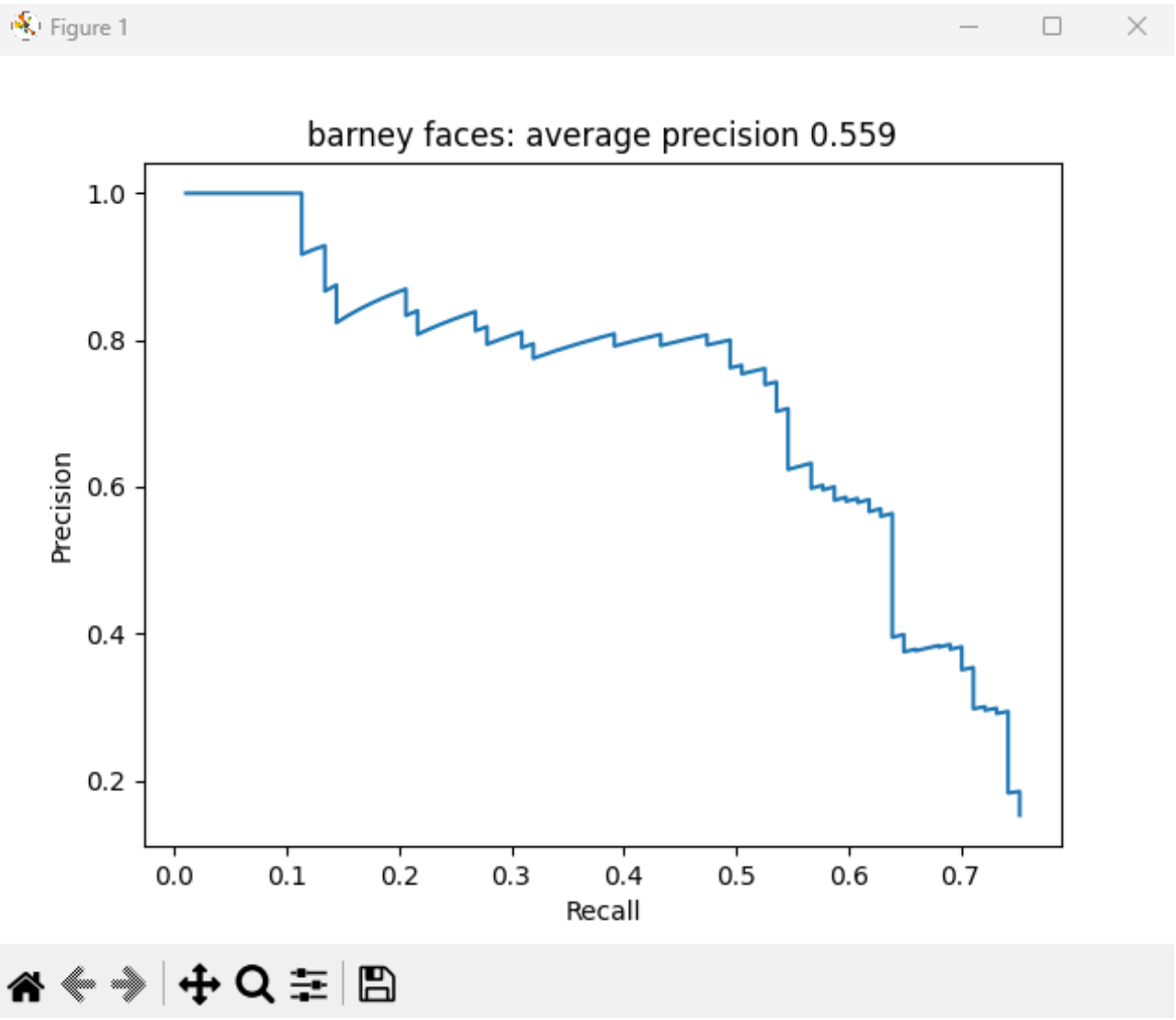


Figure 1
barney faces: average precision 0.559
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

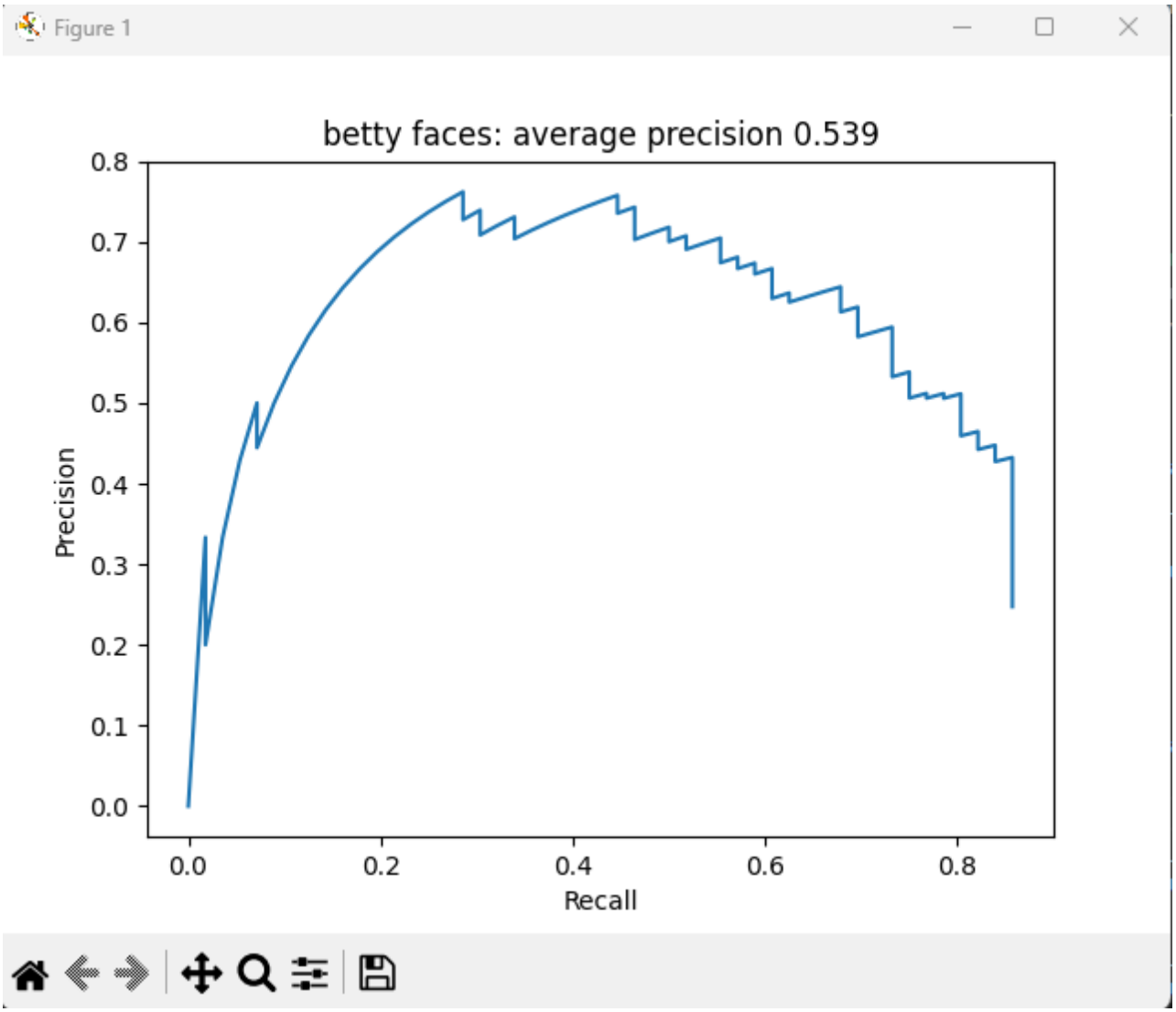
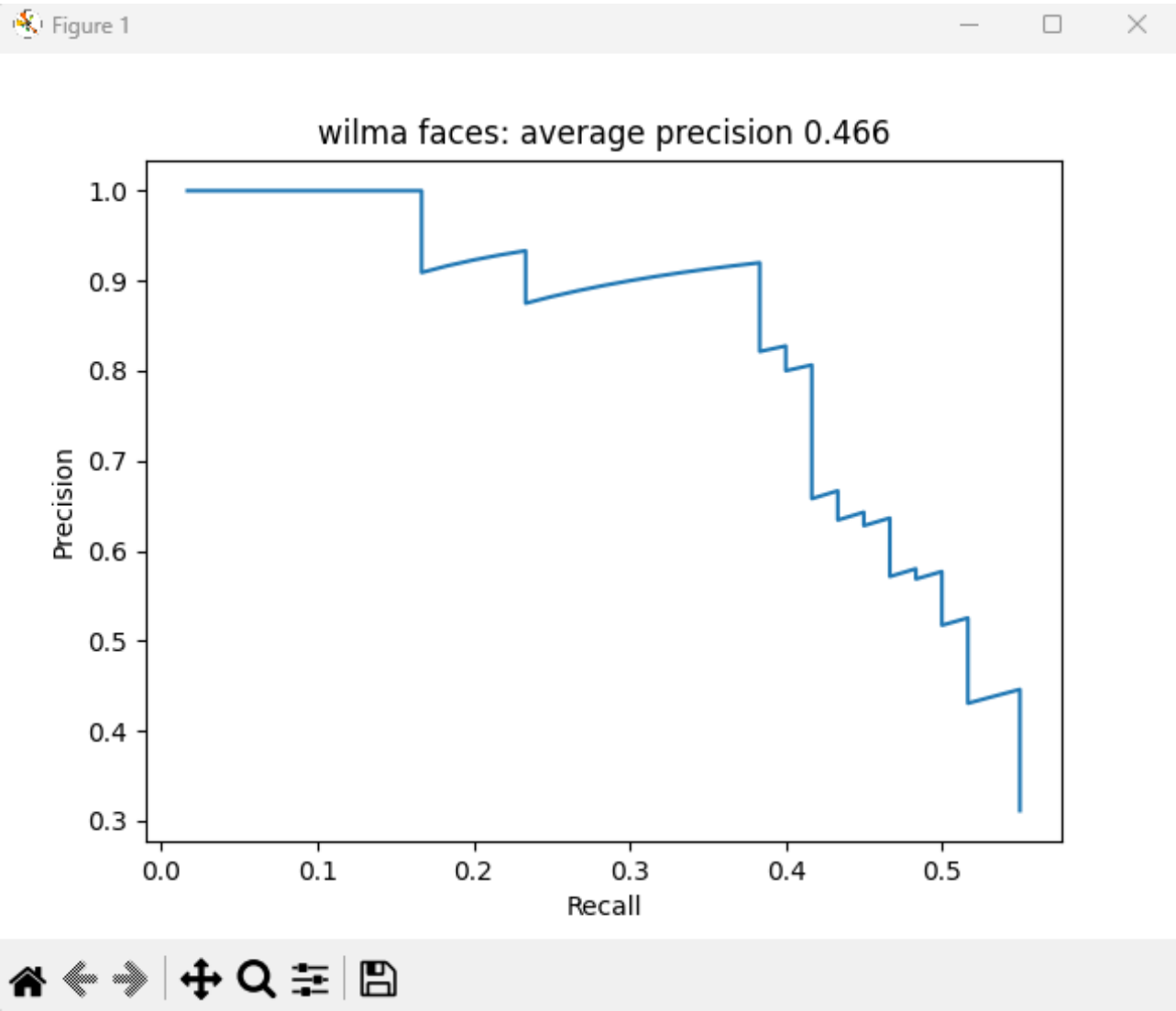


Figure 1
betty faces: average precision 0.539
Precision
Recall
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.0
0.2
0.4
0.6
0.8



4. Concluzii

Solutia prezentata implementeaza paradigma istorica de detectare faciala, anume varianta ferestrei glisante cu clasificator cu vectori suport (SVC). Performanta este una medie, nu este scazuta dar nici impresionanta.