Multimedia Project SoSe 23

Assignment 3

Multimedia Computing Lab
Prof. Dr. Rainer Lienhart
Katja Ludwig
Julian Lorenz

Submission: May 10, 2023

Please take a look at assignment 0 on how to submit your work.

Exercise 3.1 Dataset Parsing — 8 Points

One of the most important steps in any machine learning project is to properly parse the required training data. This is what you will do in this exercise. The dataset you will use is a subset of the MSCOCO (Microsoft Common Objects in Context) dataset with bounding box annotations for humans. The first step is to download the data from `https://mediastore.rz.uni-augsburg.de/get/In69eUrzen/` (approx. 43MB).

(a) Familiarize yourself with the annotation format. It is essential that you understand how relevant information is provided. Look into the *README.txt* file provided with the dataset. Implement a Python class `AnnotationRect` that wraps the following functionality:

1. The constructor takes values $x1, y1, x2, y2$ that represent the upper left and lower right coordinates of a groundtruth bounding box. Note that the annotations provided do **not** guarantee that $(x1, y1)$ is the upper left and $(x2, y2)$ is the lower right coordinate of the box. Your constructor needs to handle this and store the correct upper left and lower right coordinates inside of your class.
2. Add a function `def area(self)` that returns the area of the box.
3. Add a function `def __array__(self)` that returns the coordinates in a 1D-numpy array with the following structure: `[x1,y1,x2,y2]`. You should now be able to convert an `AnnotationRect` to its NumPy representation by calling `np.array(rect)`
4. Add a static function `fromarray(arr)` to your class that creates an `AnnotationRect` object from the above NumPy representation.

(b) Write a function `read_groundtruth_file(path)` (not part of `AnnotationRect`) that reads a groundtruth annotation file and returns a list of `AnnotationRect` objects.

(c) Determine the maximum number of annotations for a single image of the training set. Visualize an image with the maximum number of annotations together with its annotated boxes. Submit the resulting image.

Exercise 3.2 PyTorch Data Pipeline — 8 Points

In this exercise, you will implement a pipeline for data reading and data preprocessing for PyTorch.

(a) Create a new class called `MMP_Dataset` that inherits from `torch.utils.data.Dataset`. Pass the path to the `MMP_Dataset` in the constructor, e.g., `"dataset_mmp/train"` and the image size. The prototype of your constructor should be:

```
def __init__(self, path_to_data, image_size)
```

Retrieve a *sorted* list of all image filenames that can be found under `path_to_data` and store them as a class attribute. For each image, load all annotations using the function from exercise 3.1b.

(b) Implement the method `def __getitem__(self, idx)` that returns an image and the number of corresponding annotations in the following way:

1. Load the image with the given index. Use your image filename list for that purpose. Pad the image (with black pixels) to minimal quadratic size. Add the padding only on the bottom and right side.
2. Resize the image to the given size (saved in the constructor) and convert it to a PyTorch Tensor. Normalize the image with ImageNet statistics.
3. Load the corresponding annotations. For now, return 0 if there is one person in the image and return 1 if there are more.

(c) Implement `__len__(self)`.

(d) Implement the method
`def get_dataloader(path_to_data, image_size, batch_size, num_workers)`
that creates an `MMP_Dataset` and wraps it in a PyTorch `DataLoader` with the given batch size and number of workers.

Exercise 3.3 Training — 5 Points

Combine your code from the current and previous assignment and train a network that can distinguish whether there are one or more persons in an image.

Report the final accuracy on the validation set. Justify why the accuracy is a good or bad choice in this case.