BMI / CS 771 Fall 2024: Homework Assignment 1

September 2024

1 Overview

The goal of this assignment is to implement basic image processing functions and assemble them into a data augmentation pipeline for training deep learning models. The assignment covers image processing techniques (e.g., resizing, cropping, color manipulation, and rotation) and the data input pipeline.

2 Setup

- You won't need Cloud Computing or GPU for this assignment.
- This assignment is **NOT** team-based and you must complete the assignment on your own.
- You will need to install some Python packages and fill in missing code in: *./code/student_code.py*
- You can test your code by running the provided notebook: *jupyter notebook ./code/proj1.ipynb*
- You can generate the submission file once you have finished the project using: *python zip_submission.py*
- This assignment is given 12 points.

3 Details

This project assumes some basic knowledge about Python and image processing. If you do not have previous experience Python or image processing, please refer to the resources in our first tutorial, which can be found on Canvas.

3.1 Setup the Computing Environment (1 Pts)

Our first step is to set up the computing environment for the assignment

- Install Anaconda or Miniconda (recommended). We recommend using Conda to manage your packages.
- The following packages are needed: OpenCV, NumPy, Matplotlib, Jupyter Notebook, PyTorch. You might need to do a bit of research for how to install all packages (and hopefully their latest versions). See a potential issue here `https://github.com/pytorch/vision/issues/4076`. **Upon successful installation, run**
  conda list > ./results/packages.txt

3.2 Image Processing

Our next step is to implement a set of functions that manipulate an input image. These functions include resizing, cropping, color jittering, and rotation.

Image Resizing (2 Pts): Re-sampling is one of the fundamental operations in image processing. You can find many implementations in different packages, yet re-sampling might be a bit tricky. We have provided you a helper function for resizing an input image (./code/utils/image_resize). Your goal is to implement a version of adaptive resizing, often used in training deep models. Specifically, given an input image, you will resize the image to match its shortest side to a pre-specified length. Please fill in the missing code in *class* **Scale**. You must use the provided image_resize function. More details can be found in the code and comments.

Image Cropping (2 Pts): Cropping selects a region within an image. You will implement a more advanced version of random image cropping. Concretely, this version crops an image by sampling a random region, where the size of the region / its aspect ratio[1] is drawn (uniformly) from a given range of areas / aspect ratios. This region is further resized to a fixed size. This technique is described in [2] and has been widely used for training deep networks. Please fill in the missing code in *class* **RandomSizedCrop**. We recommend using NumPy for cropping the region and using our provided image_resize function for image resizing. Again, more details can be found in the code and comments.

Color Jitters (2 Pts): Small perturbations in the color space can lead to images with drastically different pixel values, yet these images are still perceptually meaningful. You will implement a version of color jitters in the *class* **RandomColor**. For each of the color channel, your code will sample α from a uniform distribution $U(1-r, 1+r)$ with $r \in (0,1)$. α is then multiplied to the corresponding color channel. This is done independently for each color channel.

[1] An image's aspect ratio is defined as the ratio of width to height.

The technique is described in several papers [1]. Details can be found in the code and comments.

• **Hint:** Type conversion has to be handled properly to avoid potential artifacts.

Rotation (3 Pts): 2D rotation is a simple form of parametric warping. It rotates the image pixels around the center of the image by a certain angle. An issue with rotation that it will create empty black pixels in the result image. See an example in Figure 1. Oftentimes, we want to avoid these black pixels. This can be done by further cropping the rotated image. While there are many different ways of cropping, we are interested in finding a rectangular region with the maximum area that does not contain a single empty pixel.

You will need to incorporate this capacity in *class* **RandomRotate**. Specifically, your implementation will sample a random rotation angle (uniformly sampled within an interval), apply the rotation to the image, and crop the region with maximum area.

• **Rubric:** You might find a solution online. Full points are granted only for those solutions with a correct implementation and *an explanation of how to find the maximum area region* in the report.
• **Hint:** Check cv2.warpAffine for the ease of your implementation.

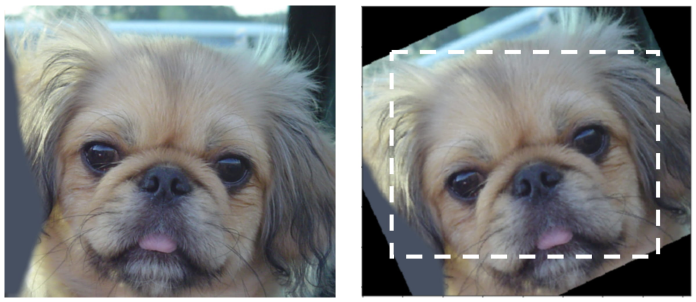

Figure 1: How can you find the rectangular region of the max area without an empty pixel after a random 2D rotation?

3.3 Data Augmentation and Input Pipeline

Putting things together, we have included a full data augmentation and input pipeline at the end of the notebook. This is done by using PyTorch dataloader class. Please go through the implementation, run the code, check the results, and **address the following questions in your report**. Note that you have to finish the code in Sec 3.1 to get this part working.

Technical Details: We have provided sample implementation that composes a series of transforms and applies them to an input image. As random augmentation is employed, you should run this part of code multiple times, in case errors produced by some corner cases were not captured.

Questions (2 Pts):

• **Q1:** What is the benefit of using PyTorch dataloader? Please compare the dataloader to a solution that simply loops over a file list and batches the images.

• **Q2:** Does the order of image transforms make a difference in the data augmentation pipeline? E.g., can you arbitrarily switch their order?

4 Writeup

For this assignment, and all other assignments, you must submit a project report in PDF. In the report you will describe your algorithm and any decisions you made to write your algorithm a particular way. You will show and discuss the results of your algorithm, and answer all questions in the assignment. *For this project, you are expected to discuss the results of your transformed images (generated by the notebook), explain your implementation for image rotation, and address questions in Sec. 3.3.* Also, discuss anything extra you did. Feel free to add any other information you feel is relevant. A good writeup doesn't just show results, it tries to share some insights or draw some conclusions from your experiments.

5 Handing in

This is very important as you will lose points if you do not follow instructions. Every time after the first that you do not follow instructions, you will lose 5%. Hand in your project as a zip file through Canvas. You can create this zip file using *python zip_submission.py*. The zip file you submit must contain the following folders:

- code/ - directory containing all your code for this assignment
- writeup/ - directory containing your report (PDF) for this assignment.
- results/ - directory containing your results (generated by the notebook)

Do not use absolute paths in your code (e.g. /user/classes/proj1). Your code will break if you use absolute paths and you will lose points because of it. Please use relative paths as the starter code already did. Do not turn in the /data/ folder unless you have added new data.

References

[1] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems 27*, pages 2366–2374. 2014.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.