BMI 826 / CS 838 Homework Assignment 1

Yin Li

September 2019

1 Overview

The goal of this assignment is to implement basic image processing functions and assemble them into a data augmentation pipeline. These functions are often used as the "front-end" for machine learning models. The assignment will cover image processing techniques (e.g., resizing, cropping, color manipulation and rotation) and basic data input pipeline.

2 Setup

- Install Anaconda. We recommend using Conda to manage your packages.
- The following packages are needed: OpenCV, NumPy, Pillow, Matplotlib, Jupyter, PyTorch. And you will need to figure out how to install them.
- You won't need Cloud Computing or GPU for this assignment.
- Run the notebook using:
  jupyter notebook ./code/proj1.ipynb
- You will need to fill in the missing code in:
  ./code/student_code.py
- Generate the submission once you've finished the project using:
  python zip_submission.py

3 Details

This project is intended to familiarize you with Python and image processing. If you do not have previous experience Python or image processing, please refer to the resources in our tutorial.

3.1 Image Processing

Image Resizing: Re-sampling is one of the fundamental operations in image processing. You can find many implementations in different packages, yet re-sampling can still be a bit tricky. We have provide you a helper function for resizing an input image (./code/utils/image_resize). Your goal is to implement a version of adaptive resizing, which is often used in machine learning models. Specifically, given an input image, you will need to resize the image to match its shortest side to a pre-specified size. 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: Cropping selects a target region from an image. You will be tasked to implement a more advanced version of image cropping. Concretely, you will need to crop a image by selecting a random region. The size of the region is drawn from a uniform distribution of a given range of image areas, and the aspect ratio is constrained to a pre-specified interval. This region is finally resized to a fixed size. This region is finally resized to a fixed size. This technique is described in [2] and has been widely used for training deep networks. Please fill in the code in the *class* **RandomSizedCrop**. We recommend using NumPy for cropping the region and using our provided image_resize function to resize the region. Again, more details can be found in the code and comments.

Color Jitters: A small perturbation in the color space can lead to images with drastically different pixel values yet are still perceptually realistic. You are asked to implement a simple 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 will be multiplied to the color channel. This is done independently for each color channel. The technique is described in several papers, e.g., [1]. Details can be found in the code and comments.
Bonus: If you multiply α to every pixel, the complexity will be $O(n)$, where n is the number of pixels in the image. Can you make it more efficient? Implement your solution and demonstrate the performance boost. **Hints:** RGB values are unsigned integers between 0 and 255.

Rotation: 2D rotation is a simple form of parametric warping. It rotates the image pixels around its center. In PIL, you can simply rotate an image using the function Image.rotate. However, this function 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 cropping the result 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.

Given an arbitrary 2D rotation, derive the analytic solution of finding the rectangular region of the max area without an empty pixel. You will need to implement this in *class* **RandomRotate**. Specifically, this class samples a rotation angel (in degrees) uniformly from a given interval, rotates the image

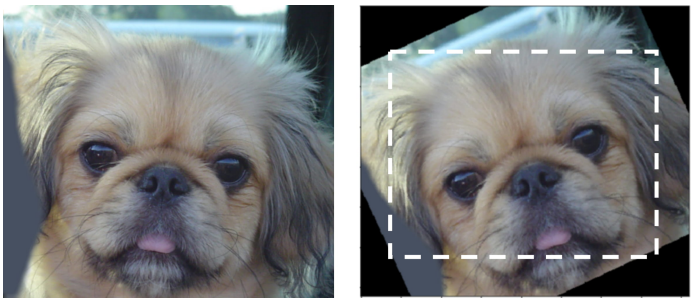

Figure 1: How can you find the rectangular region of the max area without an empty pixel after a random 2D rotation?

accordingly and crops the region of the maximum area. You will find more details in the code. **Hints:** you might want to check cv2.warpAffine.

3.2 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 and check the results. Note that you have to finish the code in Sec 3.1 to get this part working.

Composition of Image Transforms: We have provided sample implementation in helper code that composes a series of transforms and applies them to an input image. See *class* **Compose** for the details. Does the order of the transforms matter? And why? Describe your answers in the writeup.

Data Input: The training of machine learning models requires loading multiple image files at the same time. These images are formed into so called mini-batch. Each mini-batch consists of several images (2 in our example). PyTorch provides an easy-to-use data loader for this purpose. The loader is in charge of loading many small files (images and videos), pre-processing them (e.g., using specified transforms), and transferring them between CPUs and GPUs. The main challenge is to be able to do everything efficiently. Surprisingly, if the input pipeline is not well considered. It can be the major bottleneck of the training. Please describe design choices for making an efficient input pipeline in your writeup. You can explore some of the choices by looking at the parameters of the PyTorch dataloader.

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. In the case of this project, show the results of your transformed images (the test script saves such images already). 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 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%. The folder you hand in must contain the following:

- code/ - directory containing all your code for this assignment
- writeup/ - directory containing your report 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. Simply use relative paths as the starter code already does. Do not turn in the /data/ folder unless you have added new data. Hand in your project as a zip file through Canvas. You can create this zip file using *python zip_submission.py*.

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.