# BMI 826 / CS 838 Homework Assignment 1

February 2019

## 1 Overview

The goal of this assignment is to write basic image processing functions and assemble them into a simple data augmentation pipeline. These functions will be used as the front-end for machine learning models. The assignment will cover image resizing, cropping, color manipulation and rotation.

## 2 Setup

- Install Anaconda. We recommend using Conda to manage your packages.
- The following packages are needed: OpenCV3, Numpy, Pillow, Matplotlib, Jupyter. And you are in charge of installing them.
- 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 know Python or image processing, please refer to the resources in our tutorial.

**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 image (./code/utils/image_resize). And you will be tasked to fill in the missing code in *class* **Scale**. This class resizes an input image by matching

its shortest side to a pre-specified size. You must use the provided image_resize function. More details can be found in the comments.

**Image Cropping**: Cropping selects regions within an image. It is simple and quite useful. You will need to fill in the code in the *class* **RandomSizedCrop**. This class randomly crops a region, whose size is distributed evenly between a given range of the image area with aspect ratio constrained to a pre-specified interval. This region is finally resized to a fixed size. This technique is described in [2] and has been widely used for training deep networks. 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 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**. Concretely, this class samples $\alpha \in [1-r, 1+r]$ with $r$ as a pre-specified ratio between $[0, 1]$. This $\alpha$ is then multiplied to a color channel. This is done independently for each color channel. The technique is described in multiple papers, e.g., [1]. See the comments for more details.

**Rotation**: 2D image rotation (around the center of the image) is a simple form of parametric warping. 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. And you will need to implement this in *class* **RandomRotate**. Specifically, this

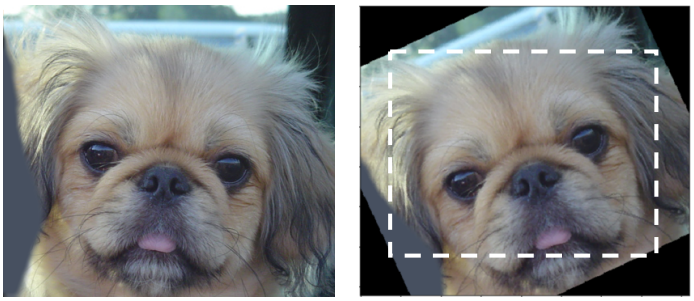

Figure 1: How can you find the rectangular region with the max area without an empty pixel after an arbitrary 2D rotation?

class samples a rotation angel (in degrees) from a given interval, rotates the image accordingly and crops the region with the maximum area. You will find more details in the code. **Hints:** you might want to check cv2.warpAffine.

**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?

# 4 Writeup

For this assignment, and all other assignments, you must submit a project report in PDF. If an assignment is team based, every team member should send the same copy of the report. Please clearly identify the contribution of all the team members. In the report you will describe your algorithm and any decisions you made to write your algorithm a particular way. Then you will show and discuss the results of your algorithm. 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. Curran Associates, Inc., 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.