BMI / CS 771 Fall 2025:
Homework Assignment 1

September 2025

1 Overview

The goal of this assignment is to implement basic image processing functions and integrate them into a data augmentation pipeline for deep learning. You will gain hands-on experience with fundamental techniques such as image resizing, cropping, color adjustment, and rotation, while also learning how to construct an efficient data input pipeline to support model training.

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 with 2 bonus points.

3 Details

This project assumes some basic knowledge about Python and image processing. If you do not have previous experience in Python or image processing, please refer to the resources in our Tutorial 1, which can be found on Canvas.

3.1 Setup the Computing Environment (1 Pt)

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

- Install Anaconda or Miniconda (recommended). We suggest using Conda to manage your packages.
- The following packages are needed: OpenCV, NumPy, Matplotlib, Jupyter Notebook, PyTorch, and torchvision. You might need to do a bit of research for how to install all packages (and hopefully their latest versions). **Upon successful installation, run**
  conda list > ./results/packages.txt

3.2 Image Processing for Data Augmentation

The next step is to implement a set of image processing functions that randomly transform an input image. This is also known as data augmentation, aiming to generate new images that preserve the original semantics while varying the pixel values. To illustrate the concept, we provide an example implementation of image cropping. Your task is to implement additional augmentation functions, including resizing, color jittering, and rotation.

Example: Image Cropping: Cropping selects a region within an image. As a starting point, we have provided 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 randomly 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. The implementation can be found in *class* **RandomSizedCrop**. Please review the code and its comments before your implementation.

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, your task is to support two resizing strategies: (1) resizing the image directly to a fixed target size, which may alter the original aspect ratio, and (2) resizing the image so that its shortest side matches a specified length, which preserves the aspect ratio. 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.

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

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

meaningful. You will implement a version of color jitters in the *class* **RandomColor**. Specifically, 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 all pixel values in the corresponding 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.

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

Rotation (3 Pts): A 2D rotation, which is a special case of affine transformations, rotates all pixels around the image center by a specified angle. A challenge with rotation is that it typically introduces empty (black) pixels along the corners of the output image (see Figure 1). To address this, it is often desirable to crop the rotated image to remove regions containing empty pixels. While there are multiple strategies for cropping, in this assignment we focus on identifying the largest possible axis-aligned rectangular region that contains no empty pixels. You will need to implement this in *class* **RandomRotate**. Specifically, your implementation will sample a random rotation angle (uniformly sampled within a pre-specified 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 *a brief explanation of how to find the maximum area region* in the report.
• **Hint:** Check cv2.warpAffine for the ease of your implementation.

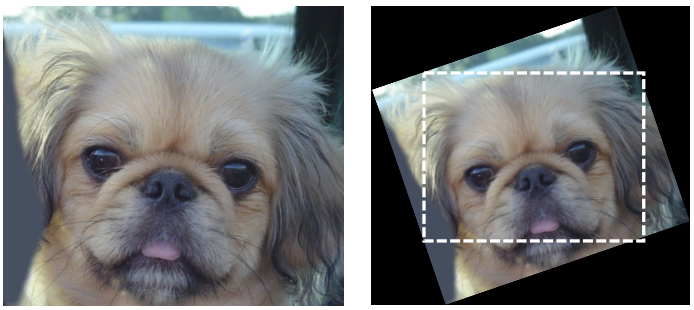

Figure 1: How can you find a rectangular region of the max area without an empty pixel after a random 2D rotation? Note: This example is illustrative.

Bonus: Max Area Rectangular after Rotation (+2 Pts): Bonus points will be provided for rigorously deriving the analytic solution of finding the rectangular region of the max area without an empty pixel after any image rotation. Additional notes and hints are provided in the appendix to guide the derivation.

• **Rubric:** Partial credit will be awarded for a a clear sketch of the derivation that states all key steps. Full credit will be awarded for a complete, rigorous derivation with justified steps.

3.3 Data Augmentation and Input Pipeline

Putting things together, we have included a full data augmentation and input pipeline for training and evaluating a neural network at the end of the notebook. Similar to our tutorial, the data IO is handled by PyTorch dataloader class, and a simple neural network is trained to classify cat vs. dog images. 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 3.2 to get this part working.

• **Hint**: As random augmentation is employed, you can run the "Create More Results" section multiple times to test your code, in case errors produced by some corner cases were not captured.

Questions:

• **Q1 (1 Pt):** What is the benefit of PyTorch dataloader? Please compare the dataloader to a solution that simply loops over a file list and batches the images.

• **Q2 (2 Pts):** Note that this assignment uses the same training conditions as in Tutorial 1. As part of your report, please compare the accuracy and loss trend you obtain here with those from the tutorial. Can you provide possible explanations for their differences?

• **Q3 (1 Pt):** 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. **If AI tools are used for any part of this assignment, they must be acknowledged in the report.**

5 Handing in

This is very important as you will lose points if you do not follow instructions. Every time *after this assignment* 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 and from Section 3.1)

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.*

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.

Appendix

Our bonus problem can be re-stated as follows. *Given an image and a 2D rotation along its center, find the largest area rectangle within the rotated image, which is also axis-aligned to the original image.*

Problem Formulation. We denote the width and height of the original image as $2a$ and $2b$, and a counterclockwise rotation angle θ around the center of image (the origin for the coordinate system). Our parameterization and coordinate system are shown in Figure 2. Note that the y axis is pointing down, consistent with the common image coordinate system. Other setups will lead to the same solution, though the derivation might be different. Without loss of generality, we assume that $b \geq a$ and that $0 < \theta < \frac{\pi}{2}$. When $\theta = 0$ or $\theta = \frac{\pi}{2}$, the solution is the rotated image itself. Other cases can be reduced to $0 \leq \theta \leq \frac{\pi}{2}$ using symmetry in the problem.

Let $\mathcal{C}$ be the region of the rotated image. For any point $\boldsymbol{p} = [x, y]^T$ in the 2D plane, $\boldsymbol{p} \in \mathcal{C}$ iff the following constraints are satisfied.

$$\begin{bmatrix} -a \\ -b \end{bmatrix} \preceq R^T \boldsymbol{p} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}^T \begin{bmatrix} x \\ y \end{bmatrix} \preceq \begin{bmatrix} a \\ b \end{bmatrix}, \tag{1}$$

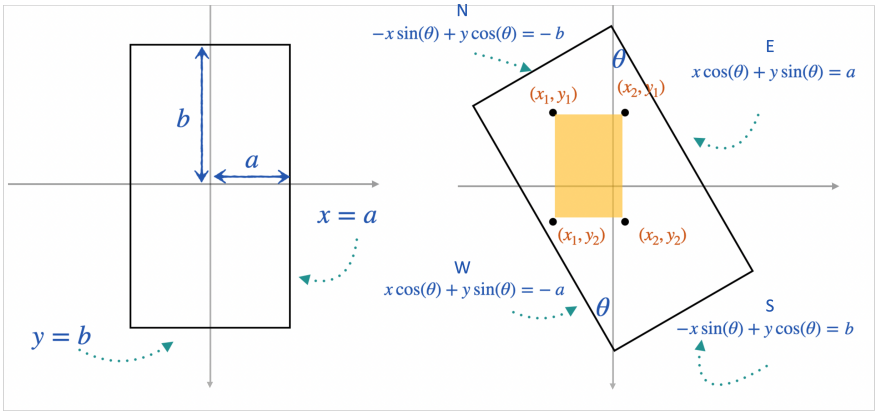

Figure 2: The parameterization and coordinates. Credit: anonymous student.

Here R is the rotation matrix with angle θ. $\preceq$ is the generalized inequality, commonly adopted in optimization literature, where $\boldsymbol{x} \preceq \boldsymbol{y}$ iff $\boldsymbol{y} - \boldsymbol{x} \in \mathcal{R}^+ \cup 0$. For convenience, we also denote the four borders of the rotated images as N, E, W, S, as shown in Figure 2.

Any axis-aligned rectangle within the rotated image can be represented by its four corners using

$$\boldsymbol{p_1} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \boldsymbol{p_2} = \begin{bmatrix} x_1 \\ y_2 \end{bmatrix}, \boldsymbol{p_3} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}, \boldsymbol{p_4} = \begin{bmatrix} x_2 \\ y_1 \end{bmatrix}. \tag{2}$$

Here we assume that $\boldsymbol{p_1}$ is the top left corner, and $\boldsymbol{p_2}$ to $\boldsymbol{p_3}$ follows the counterclockwise order. Thus, $x_2 > x_1$ and $y_2 > y_1$, which we denote as $\boldsymbol{p}_1 \prec \boldsymbol{p}_3$.

The area of this axis-aligned rectangle is given by

$$\mathcal{A} = (x_2 - x_1)(y_2 - x_1). \tag{3}$$

Finding the Largest Rectangle. Finding the largest area axis-aligned rectangle within the rotated image amounts to solve the following constrained optimization problem

$$\begin{aligned} \underset{\boldsymbol{p}_1,\boldsymbol{p}_2,\boldsymbol{p}_3,\boldsymbol{p}_4}{\arg\max} \quad & \mathcal{A} \\ s.t. \quad & \{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4\} \in \mathcal{C}, \\ & \boldsymbol{p}_1 \prec \boldsymbol{p}_3. \end{aligned} \tag{4}$$

While all constraints are linear, the objective function $\mathcal{A}$ is unfortunately nonlinear and non-convex.[2] *Can you figure out how to solve this problem?*

[2] $f(x, y) = xy$ is non-convex.