BMI 826/CS 838 Homework Assignment 1

Deepan Das (ddas27@wisc.edu)

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.

1.1 Scale:

Using the provided image_resize function, it is possible to develop a class of operations called Scale that can take an input of type int or a tuple of integers. If the input argument is a tuple, the image is scaled to the given dimensions, whereas, if it's an integer, a set of special operations are applied on it. The shorter side of the original image is assigned a dimension, same as that of the passed integer value. The other side is adjusted according to the passed integer value, using the unitary method. We can inspect both cases by running two different instances of this class with different input parameters. The notebook code used is as follows:

```
scale1 = Scale((100,150))
scale2 = Scale(300)

img1 = scale1(image1)
img2 = scale2(image1)

plt.figure()
plt.subplot(121)
plt.imshow(img1)

plt.subplot(122)
plt.imshow(img2)
```

The first object instance scales the input image to the size (100,150) and the second instance scales it to a size of (300,341) as can be seen in the image results.

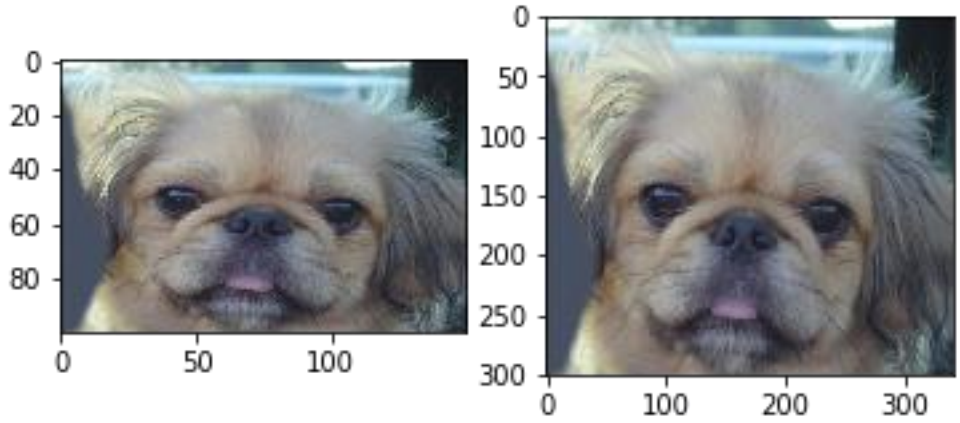


Figure 1: On the left we have the Scale ((100,150)) object being used and it can be seen that the image has indeed been resized to those dimensions. On right, we use Scale (300) and this leads to an image of the size (300,341) based on the arithmetic provided in the problem.

1.2 Random Sized Crop

Using the provided utility functions, one can build a class RandomSizedCrop that randomly crops a region, whose size is distributed evenly between a given range of the image area with aspect ratio constrained to a specific interval. Following this, the image is resized to a fixed size. There are two cases to be considered. The calculated dimensions based on the new aspect_ratio and new area is as follows:

$$\boldsymbol{dimension_1 = \sqrt{area * aspectratio}}$$

$$\boldsymbol{dimension_2 = \sqrt{\frac{area}{aspectratio}}}$$

Where, the area and aspect_ratio being mentioned are the adjusted ones based on the range being provided as an initialization argument for the class object. There are two broad cases to this problem, wherein, we can have either of $dimension_1$ or $dimension_2$ being treated as the (width, height) or (height, width) combination. This depends on the relative values of the evaluated dimensions and the original height and width of the image. The cropping dimensions have to be naturally smaller than the original image dimensions. In the case that after 10 random runs of such attempts to crop the image based on a range, it fails to generate even one case where the image is a valid crop, the image is cropped from the center based on the height and width of the original image (Fallback option). Following this, we get an image that has to be resized to a given user-defined shape. The cropped image is therefore always generated from the center of the image and the dimensions of cropping the image depends on the original aspect ratio, area and a certain range provided, to help generate the new adjusted parameters. This has been described in [1].

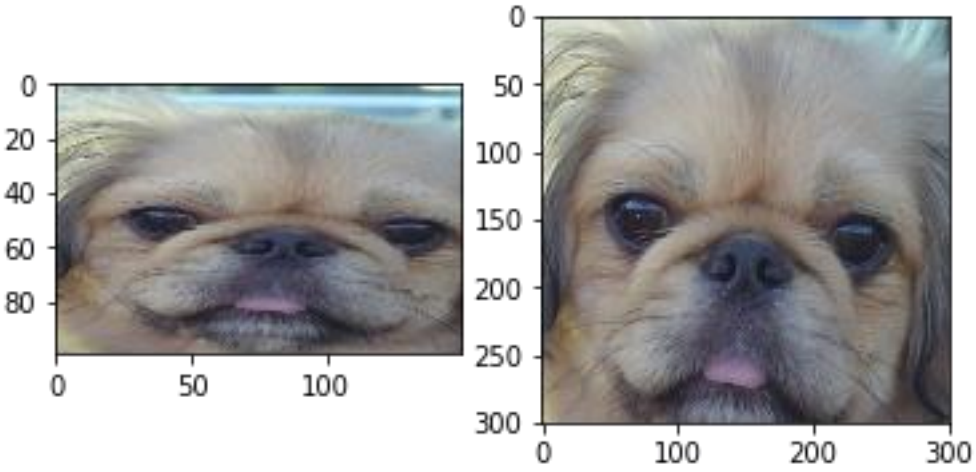


Figure 2: On the left, we have the image cropped randomly based on an aspect ratio range of (0.6,0.7) and an area_range of (0.5,0.9). The output image is resized to (100,150). On the right, we can see that the default arguments for the ranges are used and cropped image is resized to (300,300). The crops are always from the center of the image.

1.3 Color Jitter

Based on the problem statement, a small perturbation in the color space can lead to images with drastically different pixel values, and yet be perceptually realistic. This class of operation takes in a ratio in the range [0,1], for each channel and multiplies the intensity value in that specific channel with a randomly sampled factor *alpha* that is sampled in the interval [1-r, 1+r], where r is the ratio in the range [0,1]. So, a new *alpha* is generated for each channel and the channel intensity values are altered based on that. An inspection of the class operation is done on similar lines as to previous ones and the results have been shown below:

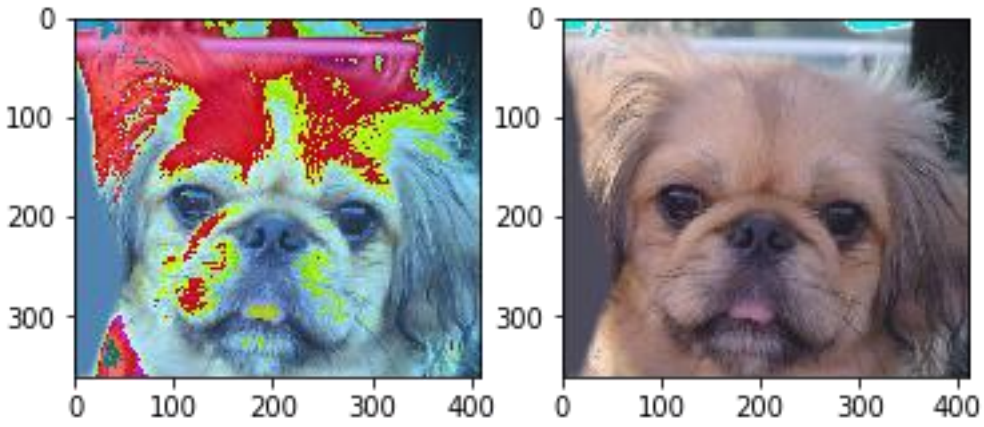


Figure 3: On the left, we have used *r* to be 0.9. Therefore, alpha can have any random value between 0.1 and 1.9. The disturbance produced here is seen to be huge. On the right, we have used *r* to be 0.1, leading to small perturbations.

1.4 Random Rotate

Based on the conditions provided in the question, the original image is to be rotated by a certain angle α, sampled from a range (-β, β). This rotation operation introduces black regions around the rotated image as the rotated image is adjusted in a bigger space now. The task thereafter is to extract the continuous maximum area of original image from this rotated image. This is done with the help of a set of trigonometrical and geometrical transformations as described in an attached screenshot below. The results can be seen in the image immediately below this.

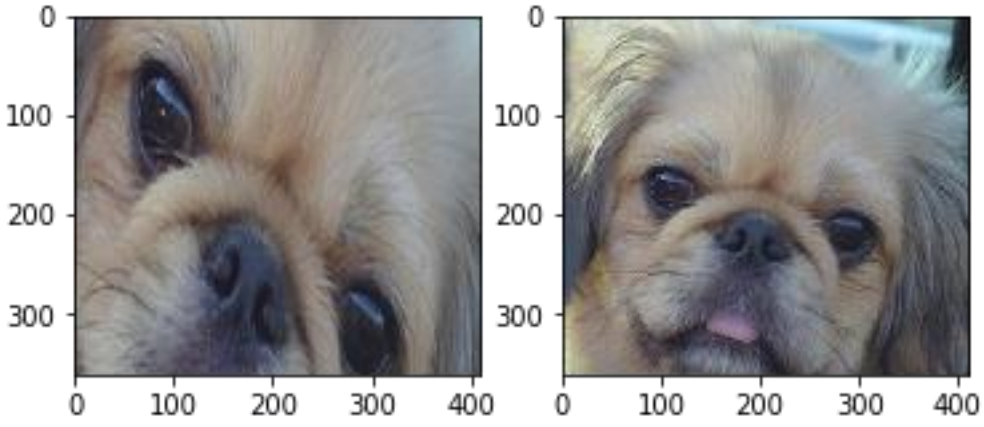
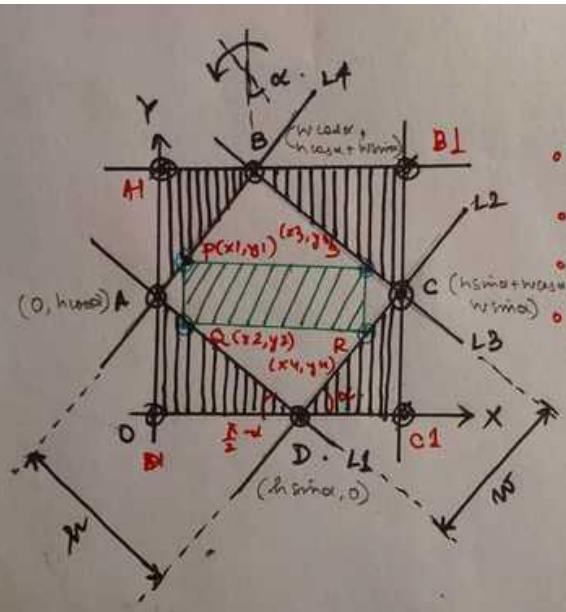


Figure 4: On the left, the original image has been rotated by 60 degrees clockwise and on the right, it has been rotated by 20 degrees. We can see that the extracted region is far more zoomed in if the angle of rotation is more, but it has to be noted that if the image is rotated in multiples of 90 degrees, the entire image could be retrieved.



- To be found: Coordinates of P, Q, R, S.
- Original Image = A1, B1, C1, D1
- Rotated Image = A, B, C, D.
- Maximum area of image content inside rotated image = P, Q, R, S.

Finding coordinates of A, B, C, D and forming equations for lines L1, L2, L3, L4.

- A (0, hcosα).
- B (wcosα, hcosα + wsinα)
- D (hsinα, 0)
- C (hsinα + wcosα, wsinα).

Finding equations for L1, L2, L3, L4:

$m1 = \frac{-h\cos\alpha}{h\sin\alpha} \Rightarrow$ L1: $y = \frac{-h\cos\alpha}{h\sin\alpha}(x - h\sin\alpha)$

$m2 = \frac{w\sin\alpha}{w\cos\alpha} \Rightarrow$ L2: $y = \frac{w\sin\alpha}{w\cos\alpha}(x - h\sin\alpha)$

$m3 = \frac{-h\cos\alpha}{h\sin\alpha} \Rightarrow$ L3: $y - w\sin\alpha = \frac{-h\cos\alpha}{h\sin\alpha}(x - h\sin\alpha - w\cos\alpha)$

$m4 = \frac{w\sin\alpha}{w\cos\alpha} \Rightarrow$ L4: $y - h\cos\alpha = \frac{w\sin\alpha}{w\cos\alpha}(x)$

⇒ L1: $yh\sin\alpha + xh\cos\alpha - h^2\sin\alpha\cos\alpha = 0$

⇒ L2: $yw\cos\alpha - xw\sin\alpha + hw\sin^2\alpha = 0$

⇒ L3: $yh\sin\alpha + xh\cos\alpha - h^2\sin\alpha\cos\alpha - hw = 0$

⇒ L4: $yw\cos\alpha - xw\sin\alpha - hw\cos^2\alpha = 0$.

Putting in values of $P(x_1, y_1)$, $Q(x_2, y_2)$, $R(x_4, y_4)$, $S(x_3, y_3)$ in the corresponding line equations, we can get the expression for each of the coordinates.

2. Results

The results after an entire set of transformations is applied to the entire image has been shown here. The transformations can be applied any number of times to generate as many altered images for a particular image and this helps in augmenting the existing dataset by improving the distribution of the data and populating it as well. The results here show the generation of one single image as well as a set of altered images.

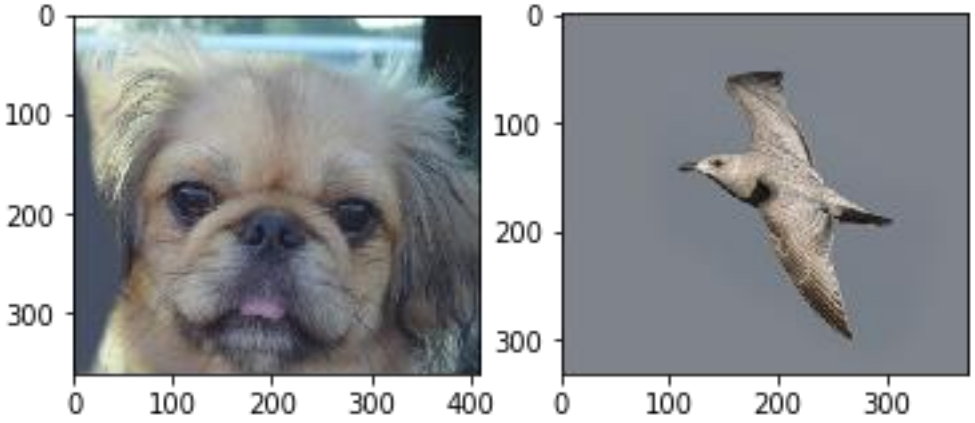


Figure 5: The original images

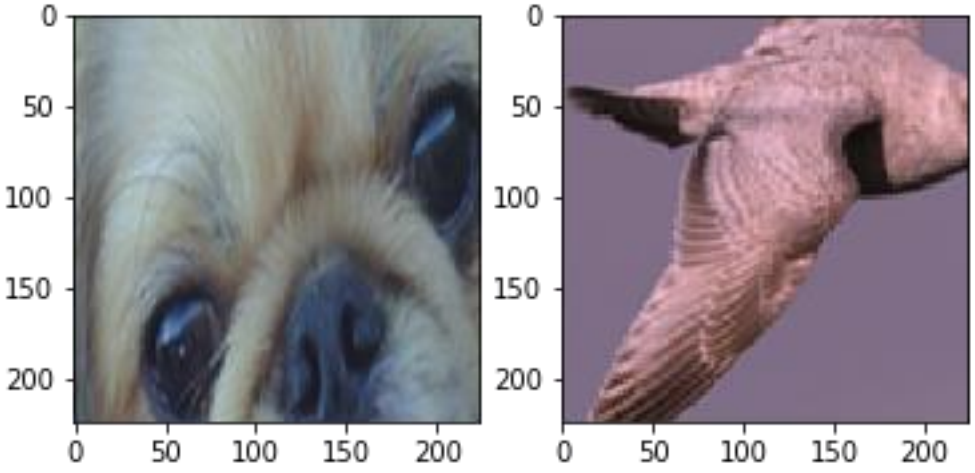


Figure 6: The resultant images after the entire set of transformations have been applied to the original images.

Figure 7: The set of 10 images generated by applying the set of transformations on Image1

Figure 7: The set of 10 images generated by applying the set of transformations on Image2

3. Discussion

As we can see, a specific set of transformations have been applied to an image to generate multiple number of alterations of the same image. This can be used to augment the existing dataset by using the original image. It is common knowledge that more data an ML algorithm has access to, the more effective it can be [2]. Data augmentation reduces overfitting on models and can be thought of as an alternative to standard regularization methods. A very generic and accepted practice for augmenting the dataset is to perform geometric and color augmentations, such as reflecting the image, cropping and translating the image, or changing the color palette of the image.

3.1 Order of transformations

All of these transformations mentioned above are essentially affine transformations, where a transformed image *y* can take the following form based on the original image *x*.

$$\boldsymbol{y = Wx + b}$$

This fundamentally means, that the order in which these transformations are applied doesn't really matter as the operations are linear in nature. This assignment explores traditional transformations using a combination of such affine transformations to manipulate the data. One can also explore other augmentation methods such as Generative Adversarial Networks, Variational Autoencoders etc. A performance comparison on standard classification task(Dogs vs Cats) has been shown here [2].

Dogs vs Cat	
Augmentation	Val. Acc.
None	0.705
Traditional	**0.775**
GANs	0.720
Neural + No Loss	0.765
Neural + Content Loss	0.770
Neural + Style	0.740
Control	0.710

Table II: Quantitative Results on Dogs vs Cats

Table 1: Table showing a comparison of performance of a standard Dog v/s Cats classification model after various Augmentation methods have been applied. It can be generally seen that Augmentation methods always improve performance, and traditional affine transformations perform the best, especially in this case.

References

[1] C. Szegedy *et al.*, “Going Deeper with Convolutions,” 2014.

[2] J. Wang and L. Perez, “The Effectiveness of Data Augmentation in Image Classification using Deep Learning.”