# Data Augmentation Programming

Sai Teja P
May 21 · 4 min read

Data Augmentation is a technique that can be used for making updated copies of images in the data set to artificially increase the size of a training dataset.This technique is very useful when the training data set is very small.

There are already many good articles published on this concept.We can refer to some of these articles at,learn about when to use data augmentation and concepts of Data agumentation can refer to data augmentation.

Imagine that you are afraid of Thanos and you believe that he is real and will visit Earth one day. As a token of measure you want to build a defence system which has a camera and the system activates when he arrives on Earth by classifying his image from the camera feed. In order to do that we need to train a reliable model for activating the defence system. If we have only 10 pictures of Thanos its very difficult for us to build a good model that can capture his presence. So in order to have multiple pictures for training set we can consider Data augmentation. A better example and scenarios of when to use augmentation is mentioned at Lets consider below image for which we want to perform Data augmentation click here.

Image Source : Google

In this articel I'm going to solely concetrate on the coding part of Data Agumenation.

At first we will look at , how this can be done using numpy and then we discuss about the image preprocessing Data Augmentation class in keras that brings simplicity for this task.

## Using Numpy

Importing required modules

```
import numpy as np
import scipy
import seaborn as sns
from scipy.ndimage import rotate
import pandas.util.testing as tm
import matplotlib.pyplot as plt


sns.set(color_codes=True)
```

**Importing_modules.py** hosted with ❤ by **GitHub** **view raw**

Loading an image to work on.

```
# reading image - modify with the path accordingly
image = np.array(plt.imread('thanos.jpg'))
# copying the image to a temporary image
img = image.copy()
print(img.shape)
# stroing sriginal width and height to use for cropping
original_width,original_height,_ = image.shape
```

**import.py** hosted with ❤ by **GitHub** **view raw**

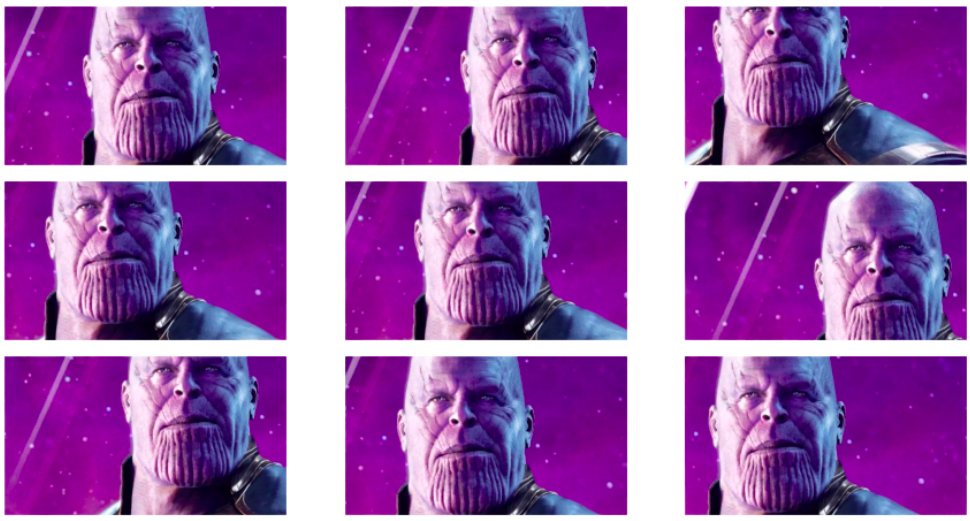
**Cropping :** with cropping we can capture the required parts of the images. Here we are cropping at random to caputer random windows of the images. Cropping too small images from original image can cause information loss.

```
def image_cropping(img, crop_size=(int(original_width*0.8), int(original_height*0.8))):
    assert crop_size[0] <= img.shape[0], "Crop width size should be less than image widt
    assert crop_size[1] <= img.shape[1], "Crop height size should be less than image hei
    w, h = img.shape[:2]
    x, y = np.random.randint(w-crop_size[0]), np.random.randint(h-crop_size[1])
    # print(x,y)
    img = img[y:y+crop_size[0], x:x+crop_size[1]]
    # print(img.shape)
    return img


# difine the number of crop ratios that we want to try
# not prefered below 70% as we might loose valuble information in the image
crop_ratio = [0.7,0.8,0.9]
# number for iteration for each crop size
num_of_iterations = 2
# lets define a loop so that we can try out multiple crop ratios and multipel iterations
for cr in crop_ratio:
    # print("cropping window = {0}%",format(cr))
    for i in range(num_of_iterations):
        plot_grid([random_cropping(img, crop_size=(crop_width , crop_height)),
                   random_cropping(img, crop_size=(crop_width , crop_height)),
                   random_cropping(img, crop_size=(crop_width , crop_height))],
```

```
            1, 3, figsize=(15, 10))
        # use below code to save the images instead of displaying them.
        # matplotlib.image.imsave('thanos_' + str(cr) + '_' +str(i) + '.png', random_cro
```

image_cropping.py hosted with ❤ by GitHub view raw



Randomly cropped images from original image

**Rotating Images :** rotating the images to capture the real time effect of capturing pictures at different angles.

```
def image_rotation(img, angle, bg_patch=(5,5)):
    assert len(img.shape) <= 3, "image shape dimensions are incorrect"
    rgb = len(img.shape) == 3
    if not rgb:
        bg_color = np.mean(img[:bg_patch[0], :bg_patch[1]])
    else:
        bg_color = np.mean(img[:bg_patch[0], :bg_patch[1], :], axis=(0,1))
    img = rotate(img, angle, reshape=False)
    mask = [img <= 0, np.any(img <= 0, axis=-1)][rgb]
    img[mask] = bg_color
    return img

 # specify the angle required to rotate the image
# for random selection of angle use np.rand
for ang in [15,30]:
    plot_grid([rotate_img(img, angle=-ang),rotate_img(img, angle=ang)],1, 2, figsize=(15
```

image_ratation.py hosted with ❤ by GitHub view raw

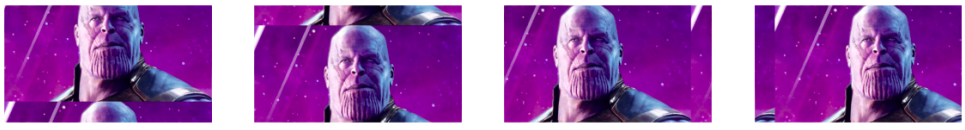
Sample output after ratating images

**Image Shifting or otherwise called Image translation :** this is noting but shifting pixels of a picture in some direction and adding back the shifted pixels back to the opposite direction.

```python
def image_shifting(img, shift=100, direction='right', roll=True):
    assert direction in ['right', 'left', 'down', 'up'], 'Directions should be top|up|le
    img = img.copy()
    if direction == 'right':
        right_slice = img[:, -shift:].copy()
        img[:, shift:] = img[:, :-shift]
        if roll:
            img[:,:shift] = np.fliplr(right_slice)
    if direction == 'left':
        left_slice = img[:, :shift].copy()
        img[:, :-shift] = img[:, shift:]
        if roll:
            img[:, -shift:] = left_slice
    if direction == 'down':
        down_slice = img[-shift:, :].copy()
        img[shift:, :] = img[:-shift,:]
        if roll:
            img[:shift, :] = down_slice
    if direction == 'up':
        upper_slice = img[:shift, :].copy()
        img[:-shift, :] = img[shift:, :]
        if roll:
            img[-shift:,:] = upper_slice
    return img

plot_grid([image_shifting(img, direction='up', shift=100),image_shifting(img, direction=
           image_shifting(img, direction='left', shift=100),image_shifting(img, directio
           1, 4, figsize=(20, 10))
```

**image_shifting.py** hosted with ❤ by **GitHub** **view raw**



Sample output after shifting images

For more better results we can combine some of these techniques , as we will get augmented pictures of different styles.

We have seen that using numpy makes us to manually change the values of the image array which is both computationally expensive and requires lot of code as mentioned above.

Now, we can try augmentation using Keras Neural Network framework, which makes our job lot easier.

## Using Tensor flow and Keras

TensorFlow has a separate class which deals with data augmentation with a lot of different options rather than just flipping, zooming and cropping the images.

By using Keras, there is no need for manual adjustment of pixels. Keras will automatically take care of these things. So the code required for augmentation with keras is way less along with multiple options.

Lets look at keras's image prepossessing ImageDataGenerator class:

```
tf.keras.preprocessing.image.ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    zca_epsilon=1e-06,
    rotation_range=0,
    width_shift_range=0.0,
    height_shift_range=0.0,
    brightness_range=None,
    shear_range=0.0,
    zoom_range=0.0,
    channel_shift_range=0.0,
    fill_mode="nearest",
    cval=0.0,
    horizontal_flip=False,
    vertical_flip=False,
    rescale=None,
    preprocessing_function=None,
    data_format=None,
    validation_split=0.0,
    dtype=None,
)
```

**keras.py** hosted with ❤ by **GitHub** **view raw**

Lets look at important arguments that are used for common data argumentation techniques:

- **rotation_range:** Int. Degree range for random rotations.
- **width_shift_range:** Float, 1-D array-like or int — fraction of total width
- **height_shift_range:** Float, 1-D array-like or int — fraction of total height
- **brightness_range:** Tuple or list of two floats. Range for picking a brightness shift value from.
- **shear_range:** Float. Shear Intensity (Shear angle in counter-clockwise direction in degrees)
- **zoom_range:** Float or [lower, upper]. Range for random zoom. If a float, [lower, upper] = [1-zoom_range, 1+zoom_range]. Fraction of

total image to be zoomed.

- **horizontal_flip:** Boolean. Randomly flip inputs horizontally.
- **vertical_flip:** Boolean. Randomly flip inputs vertically.
- **rescale:** rescaling factor. Defaults to None. If None or 0, no rescaling is applied, otherwise we multiply the data by the value provided (after applying all other transformations).
- **preprocessing_function:** function that will be applied on each input. The function will run after the image is resized and augmented. The function should take one argument: one image (Numpy tensor with rank 3), and should output a Numpy tensor with the same shape.
- **data_format:** Image data format, either “channels_first” or “channels_last”.
- validation_split: Float. Fraction of images reserved for validation (strictly between 0 and 1).
- dtype: Dtype to use for the generated arrays.

For more details and arguments please check out **tf documentation.**

Lets augment our images with some of the most common techniques like flipping,rotation,width and height shiftting, varying brightness of the image,zooming and re-scaling the images.

```
from keras.preprocessing.image import *

# lets define a ImageDataGenerator object
# change the arguments below as per the requirment
idg = ImageDataGenerator(rescale = 1/255,
                                    horizontal_flip = True,
                                    rotation_range = 30,
                                    width_shift_range = 0.3,
                                    height_shift_range = 0.3,
                                    brightness_range=[0.2,1.0],
                                    zoom_range=[0.5,1.0]
                         )


# sample code to check if our agumentation is working for a single image
# lets read our image to be processed - change the directory as needed
image = load_img("thanos.jpg")
input_arr = img_to_array(image)
# reshaping the image to a 4D array to be used with keras flow function.
input_arr = input_arr.reshape((1,) + input_arr.shape)

i = 0
# keras flow function usually work for batches
# chnage the directory and number of iterations as required
for batch in idg.flow(input_arr, batch_size=1,
                          save_to_dir='/content/cat', save_prefix='cat', save_format='jp
    i += 1
    if i > 6:
        break  # need to break the loop otherwise it will run infinite times
```

keras_Example.py hosted with ❤ by GitHub view raw

Sample output after using Keras agumentation

Now lets look at how to augment a complete data set. We will consider cifar10 data set.

```python
from keras.datasets import cifar10
import numpy as np
from keras.utils.np_utils import to_categorical

# loading the data into train ans test sets
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

num_classes = 10
#converting the output labels to one hot encoding
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)

# creating data aguementaion object with required arguments
data_gen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    brightness_range=[0.2,1.0],
    zoom_range=[0.5,1.0],
    featurewise_center=True,
    featurewise_std_normalization=True,
    )

# fitting training data to out datagen object
data_gen.fit(x_train)

#build your model here and assume its name is "model"
# model = .........

# fits the model on batches with real-time data augmentation:
# fitting the model with agumented train data
model.fit(data_gen.flow(x_train, y_train, batch_size=32),
          steps_per_epoch=len(x_train) / 32, epochs=epochs)
#do something with the model you developed
```

We can notice from the above examples, its better to use keras for data augmentation that using numpy.

Hope these large set of augmented images can help you to activate your defence system and save our planet.

The complete Jupiter notebook can be found at my ***git hub*** **.**

This is my first article , please provide feedback to improve my articles from here on.

Deep Learning Data Augmentation Keras Numpy Machine Learning