Image Augmentation on the fly using Keras ImageDataGenerator !

Link

Image Augmentation

Situation when there is not much data to train my model !

Collecting new images: Not feasible in a real-world scenario 😟 ❌

Solution ⇒ Image augmentation 😊 ✅

Image Augmentation

Image augmentation is a technique of applying different transformations to original images which results in multiple transformed copies of the same image.

Each copy, however, is different from the other in certain aspects depending on the augmentation techniques you apply like shifting, rotating, flipping, ...

Image Augmentation

Applying these small amounts of variations on the original image does not change its target class ✗

Provides a new perspective of capturing the object in real life ✓

Expand the size of your dataset ✓

Incorporate a level of variation in the dataset ✓

Allows your model to generalize better on unseen data

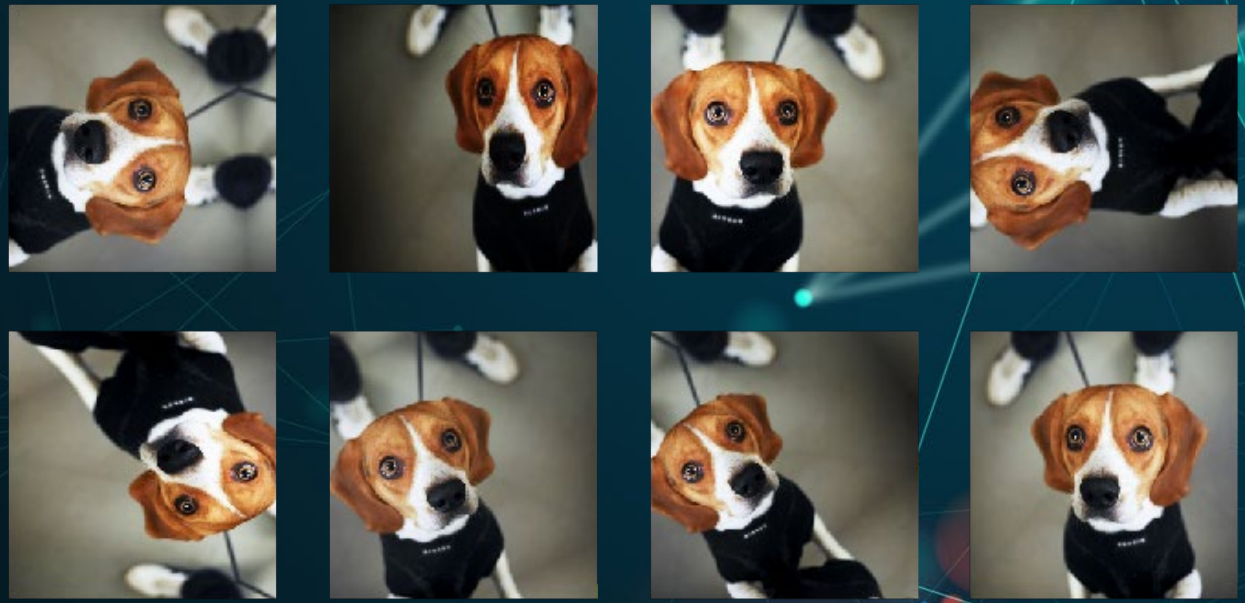
This will not only make your model robust but will also save up on the overhead memory!

Image Augmentation in Keras

The main benefits :

- Designed to provide real-time data augmentation (generating augmented images in the training stage)
- Receives new variations of the images at each epoch and does not add it to the original corpus (seeing the original images multiple times which would definitely overfit the model)
- Requires lower memory usage:
  - Without it ⇒ we load all the images at once
  - If we used it ⇒ on we are loading the images in batches which saves a lot of memory.

Augmentation Techniques With Ker
ImageDataGenerator class



1. Random Rotations

Parameter: rotation_range

- Allows you to randomly rotate images through any degree between 0 and 360

Parameter: fill_mode

- Image is rotated ⇒ Some pixels will move outside the image and leave an empty area
- Fill this with:
  - Constant value
  - Nearest: replaces the empty area with the nearest pixel values

2. Random Shifts

It may happen that the object may not always be in the center of the image 😕

Shift the pixels of the image either horizontally or vertically 💡

Parameter: **height_shift_range:** Vertical shift of image

Parameter: **width_shift_range:** Horizontal shift of image

- Float number: percentage of width or height of the image to shift
- Integer : the width or height are shifted by those many pixel values.

3. Random Flips

Parameter: vertical_flip: Flipping along the vertical axis

Parameter: horizontal_flip: Flipping along the horizontal axis

This technique should be according to the object in the image

4. Random Brightness

Most of the time our object will not be under perfect lighting condition

Parameter: **brightness_range:**

- Values less than 1.0 darkens the image
- Values above 1.0 brighten the image

Example: **brightness_range = [0.4 , 1.5]**

5. Random Zoom

Parameter: zoom_range:

- List with two values specifying the lower and the upper limit
- Float: Zoom will be done in the range [1-zoom_range , 1+zoom_range]
  - Any value smaller than 1 will zoom in on the image
  - Any value greater than 1 will zoom out on the image

ImageDataGenerator Methods

1. Flow from directory

Method: flow_from_directory():

Read the images directly from the directory: The method expects that images belonging to different classes are present in different folders but are inside the same parent folder

```
# ImageDataGenerator flow_from_directory
train_generator = datagen.flow_from_directory(
                    directory=home_path + r'/train/',
                    target_size=(400, 400), # resize to this size (Size of the input image)
                    color_mode="rgb", # for coloured images (grayscale)
                    batch_size=1, # number of images to extract from folder for every batch
                    class_mode="binary", # classes to predict (Set to binary is for 1-D
                    binary labels whereas categorical is for 2-D one-hot encoded labels)
                    seed=2020 # to make the result reproducible )
```

2. Flow from DataFrame

Method: flow_from_dataframe():

Allows you to directly augment images by reading its name and target value from a dataframe (you can use it when you have all the images stored within the same folder)

```
train_generator_df = datagen.flow_from_dataframe(
    dataframe=df_train, (DataFrame that contains the image names and target values)
    directory=home_path+'/images/', (The path to the folder that contains all the images)
    x_col="image_names", (The column name in the DataFrame that has the image names)
    y_col="emergency_or_not", (The column name in the DataFrame that has the target
    values)
    class_mode="binary", (or categorical)
    target_size=(200, 200), (Size of input images)
    batch_size=1, (Size of the batches of data)
    rescale=1.0/255,
```

Keras Fit Generator Method (1/2)

Method: fit_generator():

You have created the iterators for augmenting the images ✓

How do you feed it to the neural network so that it can augment on the fly? 😕

fit.() → fit_generator()

Method applied on the neural network model along with epochs, batch_size, and other important arguments

Keras Fit Generator Method (2/2)

```
# Directly use .flow()
model.fit_generator(datagen.flow(x_train, y_train, batch_size=batch_size),
      epochs=epochs, (the number of forward/backward passes of the training data)
      steps_per_epoch=x_train.shape[0]//batch_size, (It specifies the number of batches
of
      images that are in a single epoch)
      validation_data=(x_test, y_test), (takes the validation dataset or the validation
      generator output from the generator method)
      validation_steps=x_test.shape[0]//batch_size) ( similar to steps_per_epoch, but for
      validation data)
# or use iterator from .flow_from_directory()
model.fit_generator(train_generator, …)
# or use iterator from .flow_from_dataframe()
model.fit_generator(train_generator_df, …)
```

Thank You For Your Attention