LAB1: IMAGE AUGMENTATION & PREPARARTION

1- Image Augmentation with OpenCV

Data augmentation is a strategy that significantly increases the **diversity of data** available for **training** deep learning models, without collecting new data. Several images are created of same image by various transformations such as flipping, rotating, sharpening, cropping, etc.

1.1 Installation and Setup

Create a new notebook and to install OpenCV, you can type:

```
!pip install opencv-python
```

To import the OpenCV library to enable computer vision

```
import cv2
```

1.2 Loading and Displaying Image

Read a color image (`image = cv2.imread`) and display it in a window (`cv2.imshow`)

1.3 Image Transformations

Define a function (`def image_augmentation(image, path):`) which applies the following transformations to `image`, displays them using matplotlib (`plt.subplot(2,4,i)`) and saves the obtained images to disk (`cv2.imwrite`):

- Flipping: `cv2.flip`
- Cropping: `image[x:xend,y:yend]`
- Scaling: `cv2.resize`
- Rotating:

  ```
  M = v2.getRotationMatrix2D((cols/2,rows/2),deg,1)
  rotated=cv2.warpAffine(image,M,(cols, rows))
  ```
- Brightness:

  ```
  matrix = np.ones(image.shape, dtype = "uint8") * 120
  image = cv2.add(image, matrix)
  ```
- Median Blur: `cv2.medianBlur`
- Sharpening:

  ```
  Kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
  sharpened = cv2.filter2D(image, -1, kernel)
  ```

2 Image Preparation and Augmentation for Deep Learning with Keras

Keras is an open-source high-level Neural Network library, written in Python and running on top of the machine learning platform.

2.1 Installation and Setup

Being the fact that Keras runs on the top of Tensorflow, we need to install TensorFlow first.

```
# Requires the latest pip
!pip install --upgrade pip

# Current stable release’s Tensorflow for CPU and GPU
!pip install tensorflow

# Install Keras
!pip install keras
```

2.2 Keras Image Augmentation API

Keras provides the ImageDataGenerator class that defines the configuration for image data preparation and augmentation.

- Horizontal and Vertical Shift Augmentation

Copy and execute this script to perform horizontal shifts:

```
# example of horizontal shift image augmentation
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
# load the image
img = load_img('bird.jpg')
# convert to numpy array
data = img_to_array(img)
# expand dimension to one sample
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(width_shift_range=[-200,200])
# prepare iterator
it = datagen.flow(samples, batch_size=1)
# generate samples and plot
for i in range(9):
        # define subplot
        plt.subplot(330 + 1 + i)
        # generate batch of images
        batch = it.next()
        # convert to unsigned integers for viewing
        image = batch[0].astype('uint8')
        # plot raw pixel data
        plt.imshow(image)
# show the figure
plt.show()
```

What do you notice?

Perform vertical shifts of the image via the `height_shift_range=0.5` argument. It specifys the percentage of the image to shift as 0.5 the height of the image.

We can notice that in some cases the replicated pixels at the edge of the image may not make sense to a model.

- Horizontal and Vertical Flip Augmentation

Image flipping reverses the rows or columns of pixels in the case of a vertical or horizontal flip respectively. The flip augmentation is specified by the argument `horizontal_flip=True` or `vertical_flip=True` to the `ImageDataGenerator()` constructor.

Test to above script with these arguments.

- Random Rotation Augmentation

A rotation augmentation rotates the image clockwise randomly by a given number of degrees from 0 to 360.

Perform random rotations to the image between 0 and 90 degrees via the `rotation_range=90` argument to the `ImageDataGenerator()` constructor.

- Random Brightness Augmentation

The brightness of the image can be augmented by either randomly darkening images, brightening images, or both. This can be achieved by specifying the `brightness_range` argument to the `ImageDataGenerator()` constructor. Values less than 1.0 darken the image, whereas values larger than 1.0 brighten the image.

Perform a brightness image augmentation, allowing the generator to randomly darken the image between 1.0 (no change) and 0.3.

- Random Zoom Augmentation

Perform a zooming image augmentation be configuring the argument `zoom_range=[0.5,1.5]` to the `ImageDataGenerator()` constructor.

2.3 Application to MNIST Handwritten Digit Dataset

Let's take a look at the first 9 images in the training dataset.

```
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
# load dbata
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# create a grid of 3x3 images
fig, ax = plt.subplots(3, 3, figsize=(4,4))
for i in range(3):
    for j in range(3):
        ax[i][j].imshow(X_train[i*3+j], cmap=plt.get_cmap("gray"))
# show the plot
plt.show()
```

- Feature Standardization

It is possible to standardize pixel values across the entire dataset. (Note: Standardization is often performed for each column in a tabular dataset).

We can perform feature standardization by setting the arguments `featurewise_center=True` and `featurewise_std_normalization=True` to the `ImageDataGenerator()` constructor.

Test this script:

```
X_train = X_train.reshape((X_train.shape[0], 28, 28, 1))
X_test = X_test.reshape((X_test.shape[0], 28, 28, 1))
# convert from int to float
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
# define data preparation
datagen = ImageDataGenerator(featurewise_center=True,
featurewise_std_normalization=True)
# fit parameters from data
datagen.fit(X_train)
# configure batch size and retrieve one batch of images
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9,
shuffle=False):
    print(X_batch.min(), X_batch.mean(), X_batch.max())
    # create a grid of 3x3 images
    fig, ax = plt.subplots(3, 3, sharex=True, sharey=True,
figsize=(4,4))
    for i in range(3):
        for j in range(3):
            ax[i][j].imshow(X_batch[i*3+j], cmap=plt.get_cmap("gray"))
    # show the plot
    plt.show()
    break
```

- Image Whitening

An image whitening is a linear operation that reduces the redundancy in images. Less redundancy is intended to better highlight the structures and features in the image to the learning algorithm.

Change the above script by this command:

```
datagen = ImageDataGenerator(featurewise_center=True,
featurewise_std_normalization=True, zca_whitening=True)
```

- Random Rotations, Shifts and Flips

Add these image augmentations to the `ImageDataGenerator()` constructor (see above subsection)

- Saving Augmented Images to File

Modify the script by:

```
datagen.flow(X_train, y_train, batch_size=9, shuffle=False,
save_to_dir='images', save_prefix='aug', save_format='png')
```