Enhancing-Computer-Vision-with-PyTorch-Advanced-Image-Classification-Techniques

First, set up the environment by installing the necessary packages:

```
!pip3 install 'torch==0.4.0'
!pip3 install 'torchvision==0.2.1'
!pip3 install --no-cache-dir -I 'pillow==5.1.0'
!pip install git+https://github.com/aleju/imgaug
```

Proceed to import the required libraries and configure matplotlib for improved plot visualization:

```
import PIL
import numpy as np
import torch
import torchvision
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib as mpl
mpl.rcParams['axes.grid'] = False
mpl.rcParams['image.interpolation'] = 'nearest'
mpl.rcParams['figure.figsize'] = [15, 25]
from torchvision.utils import make_grid
from torchvision import transforms

def show_dataset(dataset, n=6):
    imgs = torch.stack([dataset[i][0] for _ in range(n) for i in
range(len(dataset))])
    grid = make_grid(imgs).numpy()
    plt.imshow(np.transpose(grid, (1, 2, 0)), interpolation='nearest')
    plt.axis('off')
```

PyTorch simplifies data augmentation, particularly using torchvision.transforms. Chain multiple transforms with torchvision.transforms.Compose:

```
tfs = torchvision.transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ColorJitter(hue=.05, saturation=.05),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(20, resample=PIL.Image.BILINEAR),
    transforms.ToTensor()
])
```

Apply these transformations to your dataset through torchvision.datasets.ImageFolder:

```
dataset = torchvision.datasets.ImageFolder('../data/classification/',
transform=tfs)
show_dataset(dataset)
```

For more complex augmentations, imgaug is an excellent choice:

```
from imgaug import augmenters as iaa
import imgaug as ia

class ImgAugTransform:
    def __init__(self):
        self.aug = iaa.Sequential([
            iaa.Scale((224, 224)),
            iaa.Sometimes(0.25, iaa.GaussianBlur(sigma=(0, 3.0))),
            iaa.Fliplr(0.5),
            iaa.Affine(rotate=(-20, 20), mode='symmetric'),
            iaa.Sometimes(0.25, iaa.OneOf([iaa.Dropout(p=(0, 0.1)),
iaa.CoarseDropout(0.1, size_percent=0.5)])),
            iaa.AddToHueAndSaturation(value=(-10, 10), per_channel=True)
        ])

    def __call__(self, img):
        img = np.array(img)
        return self.aug.augment_image(img)

tfs = transforms.Compose([
```

```
    ImgAugTransform(),
    transforms.ToTensor()
])

dataset = torchvision.datasets.ImageFolder('../data/classification/',
transform=tfs)
show_dataset(dataset)
```

imgaug also supports batch processing of images:

```
dataset = torchvision.datasets.ImageFolder('../data/classification/')
aug = iaa.Affine(rotate=(-45, 45), mode='symmetric')
imgs = [np.array(dataset[0][0]) for _ in range(6)]
imgs = aug.augment_images(imgs)
plt.imshow(np.hstack(imgs))
```

Custom image transformations can be created using Python functions and integrated using transforms.Lambda or iaa.Lambda.

Finally, compare the performance of imgaug and PyTorch augmentation techniques:

```
import time

transforms_pytorch = torchvision.transforms.Compose([
    torchvision.transforms.Resize((224,224)),
    torchvision.transforms.ColorJitter(hue=.05, saturation=.05),
    torchvision.transforms.RandomHorizontalFlip(),
    torchvision.transforms.RandomRotation(20)
])

class ImgAugTransform:
    def __init__(self):
        self.aug = iaa.Sequential([
            iaa.Scale((224, 224)),
            iaa.AddToHueAndSaturation(value=(-20, 20), per_channel=True),
            iaa.Fliplr(0.5),
```

```
            iaa.Affine(rotate=(-20, 20), mode='constant'),
        ])

    def __call__(self, img):
        img = np.array(img)
        return self.aug.augment_image(img)

transforms_imgaug = ImgAugTransform()

datasets = {
'pytorch': torchvision.datasets.ImageFolder('../data/classification/',
transform=transforms_pytorch),
'imgaug' : torchvision.datasets.ImageFolder('../data/classification/',
transform=transforms_imgaug)
}

times = {'pytorch': [], 'imgaug': []}
for _ in range(20):
    for mode in ('pytorch', 'imgaug'):
        start = time.time()
        img_pytorch =
np.vstack((np.hstack((np.asarray(datasets[mode][i][0]) for _ in range(6)))
for i in range(4)))
        end = time.time()
        times[mode].append(end - start)

for mode in ('pytorch', 'imgaug'):
    t = np.array(times[mode])
    print("{}: {:.04f}".format(mode, t.min()))
```