

# ECE 60146 HW2 Report

Zhengxin Jiang  
(jiang839@purdue.edu)

January 26, 2023

## 1 Explanation of The Mystery

Because `ToTensor()` simply applies scaling by dividing pixel values with 255. It could be the case that the max value in the batch is also 255.

## 2 Transform of Stop Sign



Figure 1: The stop sign from two perspective



Figure 2: After projective transform applied

For this task I use the function `vtv.functional.perspective()`. Four point-correlations in the two stop sign images are manually found to perform the transform.

### 3 The Custom Dataset

For the augmentation transforms, `CenterCrop()`, `ColorJitter()` and `RandomAffine()` are used. Cutting some surrounding pixels can make the object take more area in the image. Using `ColorJitter()` may help to adjust brightness/contrast/saturation level on omages.



Figure 3: Original and augmented image



Figure 4: Original and augmented image



Figure 5: Original and augmented image

## 4 DataLoader



Figure 6: Batch returned by DataLoader

Batch Size	Time(s)
DataSet	1.8115
32	1.4199
16	1.6060
8	1.8988
4	1.9022

Table 1: Time cost for loading 1000 samples (with num\_workers set to 0).

The time cost for loading samples decreases when the batch size increases. For multi-threads, due to the known issue on Windows, I can only set num\_workers to 0.

## 5 Source code

```
# ECE60146 HW2
# Zhengxin Jiang
# jiang839

from PIL import Image
import torch
import torchvision
import torchvision.transforms as tvt
from torch.utils.data import DataLoader
import matplotlib.pyplot as plt
import os
import numpy as np
import time

# Custom dataset class
class MyDataset(torch.utils.data.Dataset):

    def __init__(self, root):
        super().__init__()
        self.root = root
        self.fn_list = os.listdir(root)

    def __len__(self):
        # return len(self.fn_list)
        return 1000

    def __getitem__(self, index):
        img = Image.open(os.path.join(self.root, self.fn_list[index%10]))

        tr = tvt.Compose([
            tvt.CenterCrop(200),
            tvt.ColorJitter(),
            tvt.RandomAffine(10),
            tvt.ToTensor(),
        ])

        ts = tr(img)

        return ts, np.random.randint(10)

# Task 3.2
img1 = Image.open('stop1.jpg')
img2 = Image.open('stop2.jpg')

img2p = tvt.functional.perspective(img2, [[97,40], [154,29], [141,210], [82,208]],
    [[95,51], [158,51], [159,200], [96,201]])
# img2p.show()

img2p.save('./result_images/stop.jpg')

# Task 3.3
my_dataset = MyDataset('./data')
```

```

print(len(my_dataset))

for i in range(3):
    torchvision.utils.save_image(my_dataset[i][0], './result_images/'+str(i)+'.jpg')

# Task 3.4

md = MyDataset('./data')
data = []

start = time.time()
for i in range(1000):
    idx = np.random.randint(10)
    data.append(md[idx])

end = time.time()

print(end-start)

md = MyDataset('./data')
md_loader = DataLoader(md, batch_size=32, num_workers=2, shuffle=True)

start = time.time()
for batch, labels in md_loader:
    # for i in range(4):
    # torchvision.utils.save_image(batch[i], './result_images/'+str(i)+'b.jpg')

    continue
end = time.time()
print(end-start)

```