

Mini Batch Sampling

In 1

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.patches as patches
4 from torchvision import transforms
5 from data.loaders.DataLoader import RedWebDataset , Rescale , RandomCrop , ToTensor
```

Executed at 2023.12.02 11:12:22 in 1s 485ms

> /Users/adi/opt/anaconda3/lib/python3.10/site-packages/torchvision/io/image.py:13: UserWarning: Failed to load image Pyth

# lets choose a sample image to show how mini batch sampling works to calculate ordinal relations

In 2


```
1 transformed_dataset = RedWebDataset(root_dir='../data/ReDWeb_V1',transform=transforms.Compose([
2     Rescale(256),
3     RandomCrop(225)
4 ]))
```


Executed at 2023.12.02 11:12:22 in 41ms

In 3

```
1 sample_image = transformed_dataset[150]
2 fig,ax = plt.subplot_mosaic("AB")
3 ax["A"].imshow(sample_image["mono"])
4 ax["B"].imshow(sample_image["heat"])
5
6 for k in ax.keys():
7     ax[k].axis("off")
8     ax[k].grid("off")
9
10 plt.show()
11
12
```

Executed at 2023.12.02 11:12:22 in 106ms





In 4

```
1 # now we derive n ordinal relations from these images
2 # we will work out one such pair of example
3 # since we have random cropped and returned @ 225 . we can hardcode these values for any image
4
5 # Generate a random pixel
6 # low and high adjusting for window size. this would be adjusted and corrected while making the loss function
7 pi = np.random.randint(low=50, high=175, size= 2)
8 pj = np.random.randint(low=50, high=175, size=2)
```

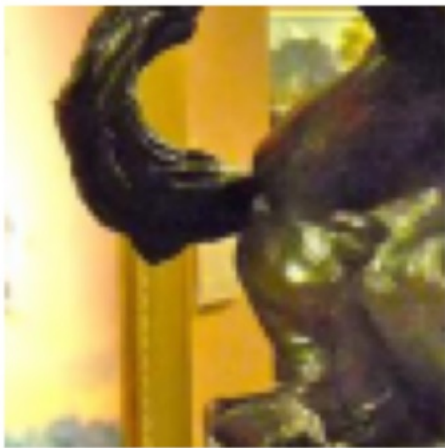
Executed at 2023.12.02 11:12:22 in 3ms

In 5


```
1 # window padding to get idea about the region we are going to talk about
2 fig , ax = plt.subplot_mosaic("AB;CD")
3 ax["A"].set_title("Point i",fontsize=6)
4 ax["A"].imshow(sample_image["mono"][pi[0]-40 : pi[0]+40 , pi[1]-40:pi[1]+40 , :])
5 ax["B"].set_title("Point j",fontsize=6)
6 ax["B"].imshow(sample_image["mono"][pj[0]-40 : pj[0]+40 , pj[1]-40:pj[1]+40 , :])
7 ax["C"].set_title("Ground truth i",fontsize=6)
8 ax["C"].imshow(sample_image["heat"][pi[0]-40 : pi[0]+40 , pi[1]-40:pi[1]+40])
9 ax["D"].set_title("Ground truth j",fontsize=6)
10 ax["D"].imshow(sample_image["heat"][pj[0]-40 : pj[0]+40 , pj[1]-40:pj[1]+40])
11
12 for k in ax.keys():
13     ax[k].axis("off")
14     ax[k].grid("off")
15
```

Executed at 2023.12.02 11:12:22 in 117ms


Point i




Point j



Ground truth i



Ground truth j



In 6

```
1 # lets calculate the ordinal relation of the pair of ordinal relation we are going to calculate
2 gi =sample_image["heat"][pi[0],pi[1]]
3 gj =sample_image["heat"][pj[0],pj[1]]
4 print(gi,gj)
```

Executed at 2023.12.02 11:12:22 in 3ms

0.1798663350281251 0.08230662784977966

In 7

```
1 def ordinal_relation(gi,gj,sigma=0.02):
2     if gi/gj > 1+sigma : return +1
3     if gi/gj < 1+sigma : return -1
4     return 0
```

Executed at 2023.12.02 11:12:23 in 2ms

In 8

```
1 # so the boundary of our ordinal relation pair is
2 [ordinal_relation(101,100) , ordinal_relation(102,100) , ordinal_relation(103,100)]
```

Executed at 2023.12.02 11:12:23 in 7ms

Out 8

[-1, 0, 1]

Out 8

[-1, 0, 1]

In 9

```
1 # here our ordinal relation will be
2 ordinal_relation(gi,gj)
```

Executed at 2023.12.02 11:12:23 in 22ms

Out 9

1

# now we just repeat it n times for a given image to genreate such ordinal mappings

pthSense > mini\_batch\_sampling > mini\_batch\_sampling.ipynb

30:1 LF UTF-8 4 spaces /Users/adi/opt/anaconda3