network_test

December 19, 2019

1 Network Tests

This file is to keep a track of the performance of our models on the overall dataset.

[1]:
```
import torch
import numpy as np
import matplotlib.pyplot as plt
from torch.utils.data import Dataset, DataLoader
from torch.utils.data.sampler import SubsetRandomSampler
import torch.optim as optim
import torch.nn as nn
from torchvision import transforms, utils

# custom file
from lib.dataset import FaceEmotionsDataset
from lib.transform import Rescale, RandomCrop, ToTensor, Normalize
from lib.network import Net
```

1.0.1 Loading the data

[2]:
```
data_transform = transforms.Compose([
    Rescale(68),
    RandomCrop(64),
    ToTensor()
])

emotions = ['neutral',
            'happiness',
            'surprise',
            'sadness',
            'anger',
            'disgust',
            'fear',
            'contempt']

dataset = FaceEmotionsDataset(csv_file='csv/balanced.csv',
                                          root_dir='img/',
                                              classes=emotions,
                                              transform=data_transform)

dataloader = DataLoader(dataset, batch_size=4,
                        shuffle=True, num_workers=4)
```

1.0.2 Importing network v.1

[3]:

```
net = Net()
net.load_state_dict(torch.load('./models/network_v1.pth'))
net = net.double()
correct = 0
total = 0
with torch.no_grad():
    for data in dataloader:
        images, emotion_ids = data['image'], data['emotion']
        outputs = net(images.double())
        _, predicted = torch.max(outputs.data, 1)
        total += emotion_ids.size(0)
        correct += (predicted == emotion_ids).sum().item()

print('Accuracy of the network on the 2740 test images: %d %%' % (100*correct/
 ↪total))
```

```
Accuracy of the network on the 2740 test images: 29 %
```

[4]:

```
class_correct = list(0. for i in range(8))
class_total = list(0. for i in range(8))
with torch.no_grad():
    for data in dataloader:
        images, emotion_ids = data['image'], data['emotion']
        outputs = net(images.double())
        _, predicted = torch.max(outputs, 1)
        c = (predicted == emotion_ids).squeeze()
        try:
            for i in range(len(emotion_ids)):
                emotion_id = emotion_ids[i]
                class_correct[emotion_id] += c[i].item()
                class_total[emotion_id] += 1
        except:
            emotion_id = emotion_ids[0]
            class_correct[emotion_id] += c.item()
            class_total[emotion_id] += 1


for i in range(8):
    print('Accuracy of %5s : %2d %%' % (
        emotions[i], 100 * class_correct[i] / class_total[i]))
```

```
Accuracy of neutral : 71 %
Accuracy of happiness : 93 %
Accuracy of surprise :  0 %
Accuracy of sadness :  0 %
Accuracy of anger :  0 %
Accuracy of disgust :  0 %
Accuracy of  fear :  0 %
Accuracy of contempt :  0 %
```

1.0.3 Importing network v.2

[5]:

```
net = Net()
net.load_state_dict(torch.load('./models/network_v2.pth'))
net = net.double()
correct = 0
total = 0
with torch.no_grad():
    for data in dataloader:
        images, emotion_ids = data['image'], data['emotion']
        outputs = net(images.double())
        _, predicted = torch.max(outputs.data, 1)
        total += emotion_ids.size(0)
        correct += (predicted == emotion_ids).sum().item()

print('Accuracy of the network on the whole dataset: %d %%' % (100*correct/
 ↪total))
```

```
Accuracy of the network on the whole dataset: 74 %
```

[6]:

```
class_correct = list(0. for i in range(8))
class_total = list(0. for i in range(8))
with torch.no_grad():
    for data in dataloader:
        images, emotion_ids = data['image'], data['emotion']
        outputs = net(images.double())
        _, predicted = torch.max(outputs, 1)
        c = (predicted == emotion_ids).squeeze()
        try:
            for i in range(len(emotion_ids)):
                emotion_id = emotion_ids[i]
                class_correct[emotion_id] += c[i].item()
                class_total[emotion_id] += 1
        except:
            emotion_id = emotion_ids[0]
            class_correct[emotion_id] += c.item()
            class_total[emotion_id] += 1


for i in range(8):
    print('Accuracy of %5s : %2d %%' % (
        emotions[i], 100 * class_correct[i] / class_total[i]))
```

```
Accuracy of neutral : 91 %
Accuracy of happiness : 81 %
Accuracy of surprise : 73 %
Accuracy of sadness : 58 %
Accuracy of anger : 81 %
Accuracy of disgust : 74 %
Accuracy of  fear : 20 %
Accuracy of contempt : 11 %
```

[8]:

```
validation_split = 0.3
shuffle_dataset = True
random_seed = 0

# Creating data indices for training and validation splits:
dataset_size = len(dataset)
indices = list(range(dataset_size))
split = int(np.floor(validation_split * dataset_size))
if shuffle_dataset :
    np.random.seed(random_seed)
    np.random.shuffle(indices)
train_indices, val_indices = indices[split:], indices[:split]

# Creating PT data samplers and loaders:
train_sampler = SubsetRandomSampler(train_indices)
valid_sampler = SubsetRandomSampler(val_indices)

train_loader = DataLoader(dataset, batch_size=4, sampler=train_sampler)

test_loader = DataLoader(dataset, batch_size=4, sampler=valid_sampler)
```

[9]:

```
correct = 0
total = 0
with torch.no_grad():
    for data in train_loader:
        images, emotion_ids = data['image'], data['emotion']
        outputs = net(images.double())
        _, predicted = torch.max(outputs.data, 1)
        total += emotion_ids.size(0)
        correct += (predicted == emotion_ids).sum().item()
```

```
print('Accuracy of the network on the train dataset: %d %%' % (100*correct/
 ↪total))
```

```
Accuracy of the network on the train dataset: 79 %
```

[10]:

```
correct = 0
total = 0
with torch.no_grad():
    for data in test_loader:
        images, emotion_ids = data['image'], data['emotion']
        outputs = net(images.double())
        _, predicted = torch.max(outputs.data, 1)
        total += emotion_ids.size(0)
        correct += (predicted == emotion_ids).sum().item()

print('Accuracy of the network on the test dataset: %d %%' % (100*correct/
 ↪total))
```

```
Accuracy of the network on the test dataset: 66 %
```