testing_nn

January 13, 2020

[20]:
```
from nn import CameraClassifier
import numpy as np

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

import matplotlib.pyplot as plt

import os
import sys
sys.path.append('../checkpoints')
sys.path.append('../training_data')
sys.path.append('../bin')
```

[21]:
```
model = CameraClassifier()
model.load_state_dict(torch.load('final_model_weights'))
model.eval()
```

[21]:
```
CameraClassifier(
  (hidden): Linear(in_features=25, out_features=20, bias=True)
  (output): Linear(in_features=20, out_features=16, bias=True)
)
```

[23]:
```
test_data = np.load('./training_data/piano.npz')
criterion = nn.CrossEntropyLoss()
```

[25]:
```
correct = 0
total = 0
total_loss = 0
with torch.no_grad():
    for x, y in zip(test_data['x'], test_data['y']):
        inputs = torch.Tensor(x)
        labels = torch.tensor([y], dtype=torch.long)

        outputs = model(inputs)
```

```
        _, predicted = torch.max(outputs.data, 1)
        loss = criterion(outputs, labels)
        total_loss += loss.item()

        total += labels.size(0)
        correct += (predicted == labels).sum().item()

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

```
Accuracy of the network: 40 %
```