Imports & private packages

[ ] ↳ *8 cells hidden*

Training routine

[ ] ↳ *2 cells hidden*

Run

Network definition

```
# Hyperparameters
seq_len = 10
measurement = 'C'
batch_size = 256
nr_epochs = 250
lr = 0.001
hidden_size_linear = 128
hidden_layers = 2
step = 0.05
```

```
def create_data_loaders(elements, splits, seq_len, validation=False, measurement='G', u
    dc = DatasetCreator(elements=elements,  splits=splits, validation=validation, seq_l
    train_dataset, test_dataset, val_dataset = dc.get_datasets()

    # Create the DataLoaders
    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=shuffle)
    test_loader = DataLoader(test_dataset, batch_size=1, shuffle=shuffle)
    if val_dataset:
        val_loader = DataLoader(val_dataset)
    else:
        val_loader = None

    return train_loader, test_loader, val_loader
```

```
# Create the network
t = True
net = ElementClassifier(train, in_features=seq_len * 2, hidden_size_linear=hidden_size_

# Check if cuda is available and send net to device
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

```

```
net = net.to(device)

optimizer = Adam(net.parameters(), lr=lr)
```

Train

```
elements = None

train_loader, test_loader, val_loader = create_data_loaders(elements, (0.8, 0.2), seq_l
```

```
Dataset shape:  (205584, 10, 3)
```

```
best_net, losses, accuracies = train(net, optimizer, train_loader, test_loader, batch_s
test(best_net, test_loader)
```

```
Training accuracy:  0.9656198925986458
Epoch  70
Loss:  0.08894135977440602
Training accuracy:  0.9687670246711806
Epoch  80
Loss:  0.07645864214014444
Training accuracy:  0.9721914156743715
Epoch  90
Loss:  0.07235778130848232
Training accuracy:  0.9734123278076114
Epoch  100
Loss:  0.07310931078739599
Training accuracy:  0.973529068409993
Epoch  110
Loss:  0.06916697218135584
Training accuracy:  0.9753288193633746
Epoch  120
Loss:  0.06475094252416091
Training accuracy:  0.9766518795236984

Epoch  130
Loss:  0.06306611697752244
Training accuracy:  0.9768318546190365
Epoch  140
Loss:  0.047187146112278186
Training accuracy:  0.9827807611487275
Epoch  150
Loss:  0.056403295289335034
Training accuracy:  0.9797844190209355
Epoch  160
Loss:  0.05802413404088435
Training accuracy:  0.9802367888551639
Epoch  170
Loss:  0.05823907273998003
Training accuracy:  0.9828342672581524
Epoch  180
Loss:  0.0567828185855473
Training accuracy:  0.979998443458635
Epoch  190
Loss:  0.053901590229192886
```

```
Training accuracy:  0.9808934547435598
Epoch  200
Loss:  0.060412026237199164
Training accuracy:  0.978490544011207
Epoch  210
Loss:  0.049866712421489455
Training accuracy:  0.9820608607673749
Epoch  220
Loss:  0.05946626950996536
Training accuracy:  0.9788894077360106
Epoch  230
Loss:  0.05092071947955818
Training accuracy:  0.9831115261888085
Epoch  240
Loss:  0.07191513026737877
Training accuracy:  0.9853539185928866
Test accuracy:  0.9501263362487852
0.9501263362487852
```

```
test(best_net, test_loader)
```

+ Code + Text

```
torch.save(best_net, 'ElementClassifier_9806.pth')
# PLOTS!!!
```

```
print(type(losses))
fig, ax = plt.subplots(1, 2, figsize=(14,7))
ax[0].plot(losses)
ax[0].set_xlabel('Epochs'), ax[0].set_ylabel('Loss')
ax[0].set_title('Losses over time')
ax[0].grid()
ax[1].plot(accuracies)
ax[1].set_xlabel('Epochs'), ax[1].set_ylabel('Training accuracy')
ax[1].set_title('Training accuracy over time')
ax[1].grid()
plt.show()
```

Test on Barin data

```
net = torch.load('/content/ElementClassifier_9782_3.pth').to(device)
```

```
inp = torch.tensor([[[ .3,   11.403],
                    [ .7,   22.25],
                    [ .8,   23.364],
                    [.9,   24.248],
                    [1.0,   24.979]]]).to(device)
```

```
out = net(inp)
```

```
print(Encoder()(out.argmax(dim=-1).item()))
print(Softmax(dim=-1)(out).amax(dim=-1))
```

```
B
tensor([1.0000], grad_fn=<AmaxBackward0>)
```

```
test(net, test_loader)
```

```
Test accuracy:  0.9772111058201572
0.9772111058201572
```

✓ 22m 2s completed at 2:46 PM