Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CS7GV1 Computer Vision – assignment2.1

Recognizing Tiny Images of Fashion Components

Matej Ulicny (ulinm@scss.tcd.ie)
Prof. Rozenn Dahyot
Date 1/11/18

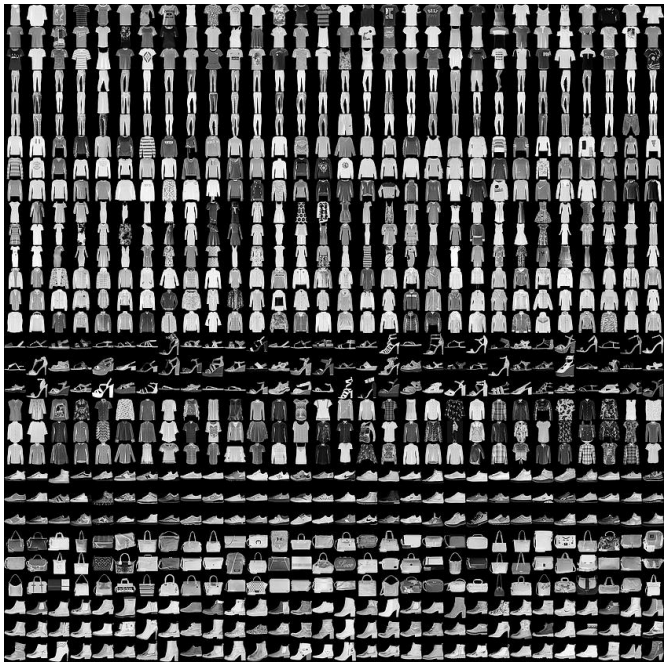
Problem definition

Fashion Component Classification

Assign correct label to an image

- Dataset: Fashion-MNIST (https://github.com/zalandoresearch/fashion-mnist)
- 28x28 grayscale images
- 60k train / 10k test
- More challenging image content than MNIST

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot



Data Loading

Pytorch wrapper

- https://pytorch.org/docs/stable/torchvision/datasets.html#fashion-mnist

  class **torchvision.datasets.FashionMNIST**(*root, train=True, transform=None, target_transform=None, download=False*)

- You can simply replace MNIST with FashionMNIST

Task

Submission Deadline 25/11/2018

Your task is to design and train a model (for example Convolutional Neural Network) to achieve the best accuracy on test set with pytorch.

Your submission should include:

- A short report (2-3 pages long with images) that explains your results and motivation for your architectural choices. The report has to include:
  - Detail of the model (table or diagram)
  - Graph showing train & test error and/or loss during training
- Source code
- Weights of the trained model
- Softmax probabilities for all test samples maintaining the default ordering. Predictions should be saved to 'predictions.txt' containing a line of 10 floating-point numbers per sample separated by space, consisting in total 10 000 lines.

Marking scheme (30 marks)

- **Quality of the report**
- **Quality of the code**
- **Performance of the designed network**

Saving and Loading Model

- You can save model by:

```
model = MyModel()

torch.save(model.state_dict(), 'mymodel.pt')
```

- Load weights by:

```
model.load_state_dict(torch.load('mymodel.pt'))
```

Saving Predictions

```
def save_predictions(model, device, test_loader, path):
    model.eval()
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = F.softmax(model(data), dim=1)
            with open(path, "a") as out_file:
                np.savetxt(out_file, output)
```

Note: `test_loader's` shuffle has to be false to preserve the correct order!