Raport z zadania 2. - Konwolucyjne sieci neuronowe

Michał Stefanik

16 października 2023

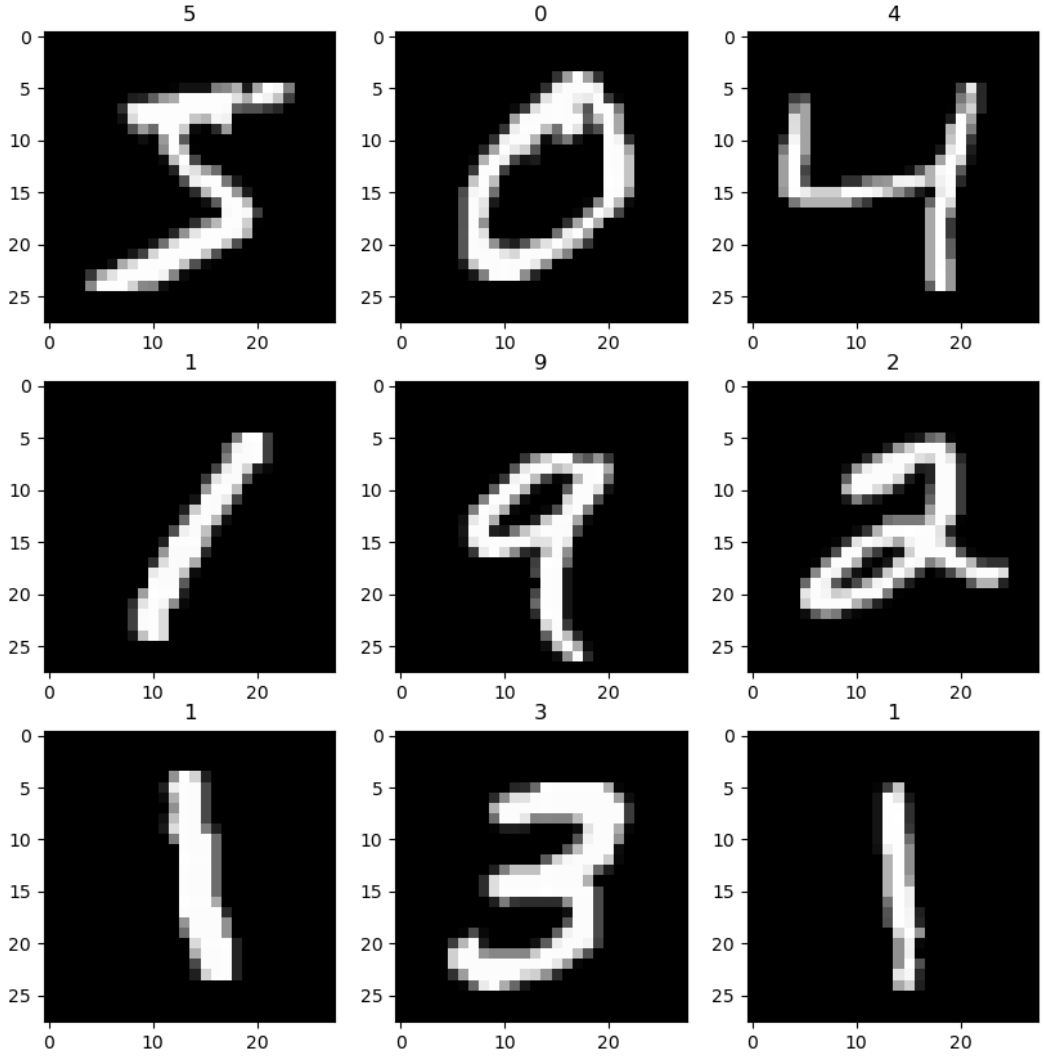
1 Co trzeba było zrobić?

Zapoznanie się z technikami klasyfikacji obrazów sieciami konwolucyjnymi.

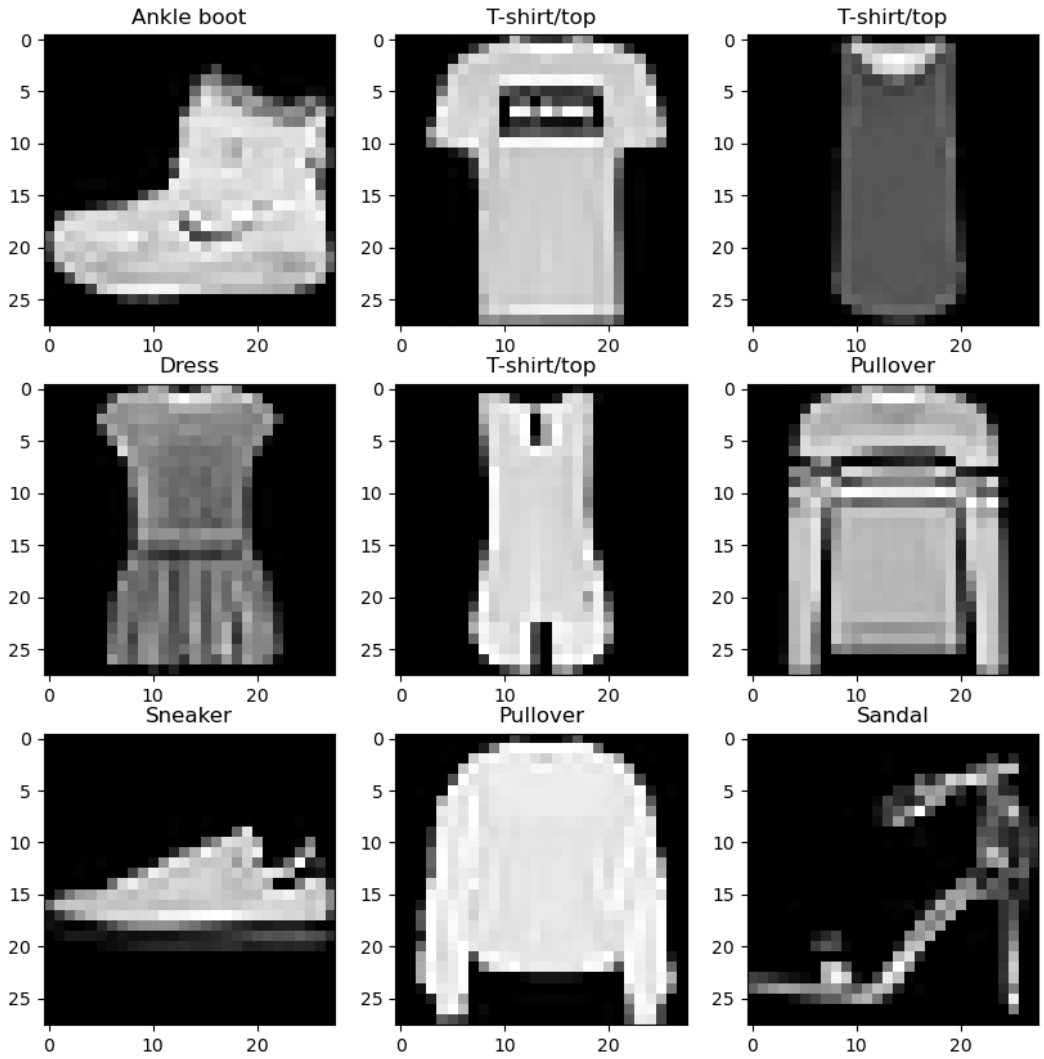
2 Zbiór danych

Zbiór danych na których przeprowadzałem ćwiczenie to połącznie MNIST z FashionMnist. Do każdego obrazka z MNIST zostały dodane dwa obrazki z FashionMnist - jeden z lewej strony, drugi z prawej strony. Lewy obrazek był w kolejności takiej jak w oryginalnym zbiorze, prawy pochodził z tego samego zbioru, ale po losowej permutacji. W zależności od parzystości liczby, jako oznaczenie rekordu została wybierana klasa obrazka lewego (dla parzystych) lub prawego (dla nieparzystych). Kod tworzenia takiego zbioru danych pokazany jest na listingu poniżej.

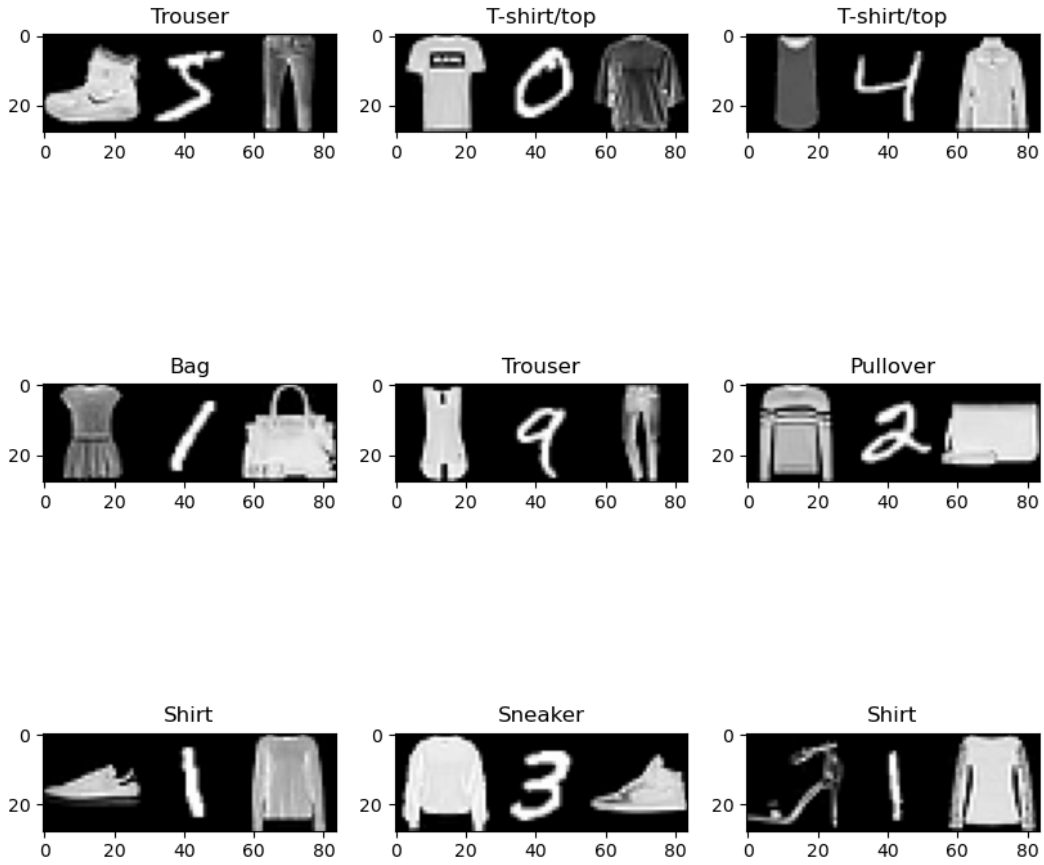
```
def shuffle_dataset(dataset):
    random_permutation = torch.randperm(len(dataset))
    return dataset.data[random_permutation], dataset.targets[random_permutation]


def generate_dataset(dataset_1, dataset_2):
    left_data = dataset_1.data
    left_labels = dataset_1.targets

    right_data, right_labels = shuffle_dataset(dataset_1)

    center_data = dataset_2.data
    center_labels = dataset_2.targets

    data = (
        torch.cat((left_data, center_data, right_data), 2)
        .float()
        .unsqueeze(1)
        .to(device)
    )
    data = data / 255
    labels = torch.where(center_labels % 2 == 0, left_labels, right_labels).to(device)
    return torch.utils.data.TensorDataset(data, labels)
```



Rysunek 1: Przykładowe obrazki z MNIST



Rysunek 2: Przykładowe obrazki z FashionMNIST



Rysunek 3: Przykładowe obrazki z połączonego zbioru danych

3 Model

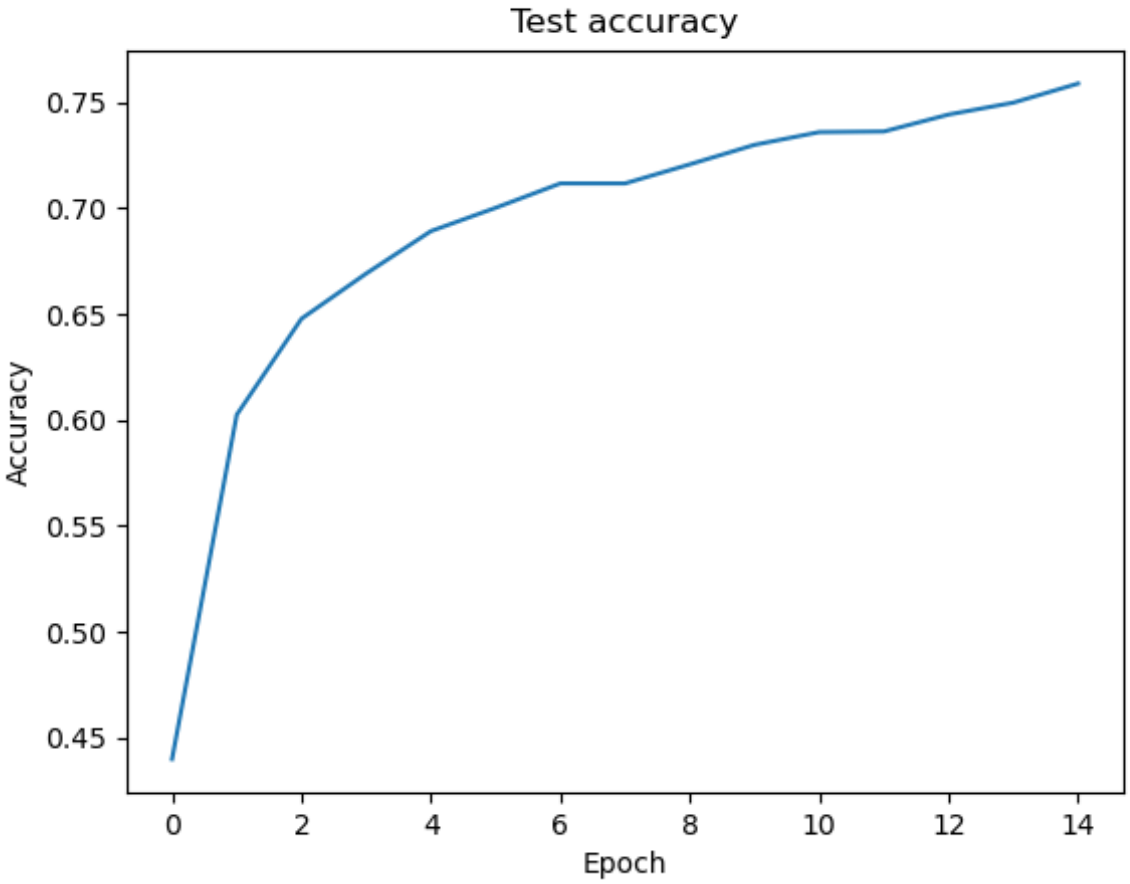
Model składa się z trzech warstw konwolucyjnych oraz dwóch warstw w pełni połączonych. Jako funkcji aktywacji użyto ReLU. Jak funkcji straty użyto CrossEntropyLoss. Optimizerem był Adam. Dokładne wartości parametrów modelu pokazane są na listingu poniżej.

```
model = torch.nn.Sequential(
torch.nn.Conv2d(1, 32, 3),
torch.nn.ReLU(),
torch.nn.MaxPool2d(2),
torch.nn.Conv2d(32, 64, 3),
torch.nn.ReLU(),
torch.nn.MaxPool2d(2),
torch.nn.Conv2d(64, 64, 3),
torch.nn.ReLU(),
torch.nn.MaxPool2d(2),
torch.nn.Flatten(),
```

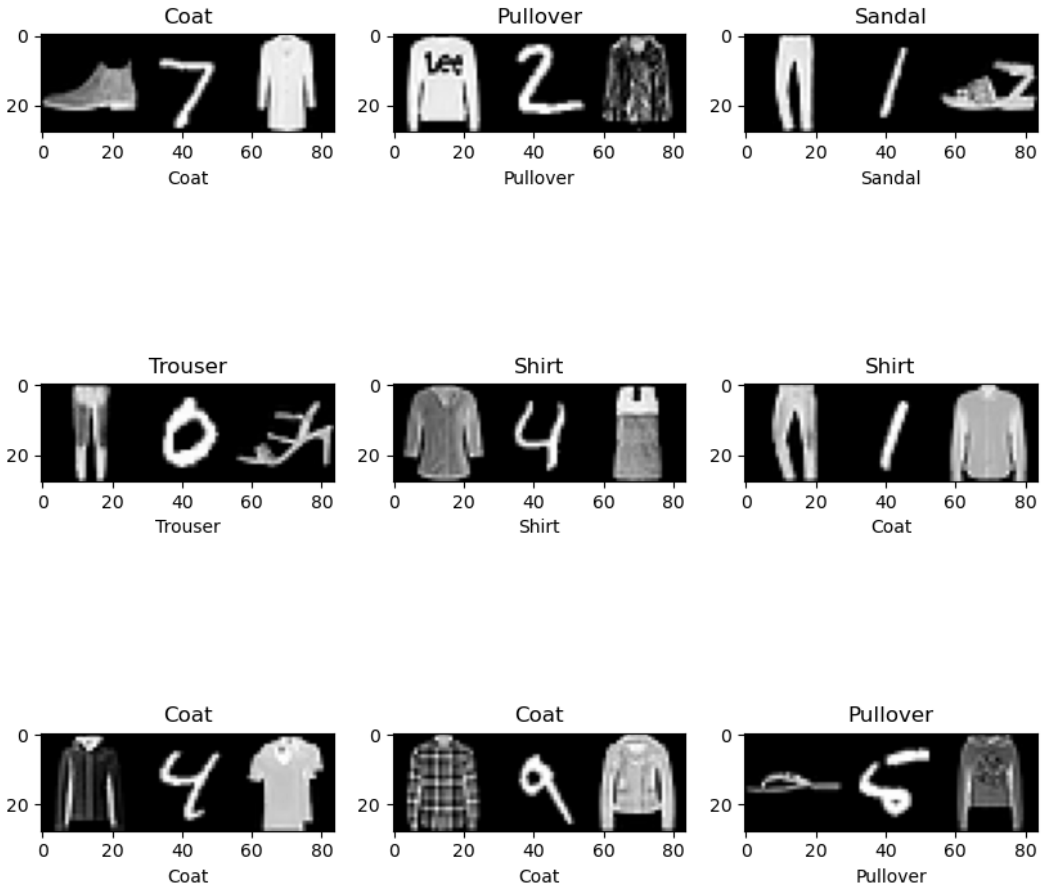
```
        torch.nn.Dropout(0.5),
        torch.nn.Linear(512, 32),
        torch.nn.ReLU(),
        torch.nn.Linear(32, 10),
)
```

4 Wyniki

Po 15 epokach z $learning\_rate = 0.001$ oraz $batch = 128$ osiągnięto dokładność na zbiorze testowym 78.33%.



Rysunek 4: Wykres dokładności na zbiorze testowym



Rysunek 5: Obrazki z przewidzianymi klasami