Sprawozdanie 6

Krzysztof Maciejewski 260449

Wstęp

Do przeprowadzenia doświadczenia został użyty zbiór Fashion MNIST posiadający obrazy czarnobiałe o wielkości 28x28.

```
class CNNModel(nn.Module):
    def __init__(self, num_channels, kernel_size, pool_size):
        super(CNNModel, self).__init__()
        self.conv1 = nn.Conv2d(1, num_channels, kernel_size)
        self.relu1 = nn.ReLU()

        self.conv2 = nn.Conv2d(num_channels, num_channels, kernel_size)
        self.relu2 = nn.ReLU()

        self.pool = nn.MaxPool2d(pool_size)
        self.flatten = nn.Flatten()
        self.lazy_linear = nn.LazyLinear(10)

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu1(x)
        x = self.conv2(x)
        x = self.relu2(x)
        x = self.pool(x)
        x = self.flatten(x)
        x = self.lazy_linear(x)
        return x
```

Sieć posiada dwie warstwy konwolucyjne i jedną warstwę liniową. Możemy zdefiniować liczbę kanałów wyjściowych, rozmiar filtra oraz okna poolingu. Jako optymalizator został użyty Adam.

Badanie wpływu paramterów

• Liczba kanałów wyjściowych warstwy konwolucyjnej

Liczba kanałów	Accuracy
8	88.36%
16	90.19%
32	90.62%
64	91.92%
128	89.16%

Najlepiej poradziła sobie sieć z 64 kanałami wyjściowymi. Ten rozmiar okazał się optymalny i pozwolił na wyekstraktowanie dużej liczby cech obrazów za pomocą wielu kanałów, co przełożyło się na lepszą dokładność.

• Rozmiar filtra warstwy konwolucyjnej

Rozmiar filtra	Accuracy
2	88.97%
3	90.42%
5	87.97%
7	88.50%
9	88.17%

Rozmiar filtra 3x3 okazał się być najlepiej dostosowany do rozmiaru obrazów 28x28.

• Rozmiar okna poolingu

Rozmiar okna	Accuracy
2	90.33%
3	90.13%
5	86.92%

W przypadku najmniejszego okna poolingu model poradził sobie nieznacznie lepiej od okna o wielkości 3. Okno o wielkości 5 było już zdecydowanie zbyt duże do tych obrazów i dlatego dokładność była dużo mniejsza.

• Zaburzenia danych: dane można zaburzyć dodając do wejściowego batcha batch o tych samych wymiarach, wygenerowany jako szum gaussowski o różnych odchyleniach. Przebadać scenariusze: szum dodany w danych testowych vs szum dodany zarówno w testowych, jak i treningowych.

Tryb	Accuracy
Brak szumów	90.19%
Szum w danych testowych	87.84%
Szum w danych testowych i treningowych	89.63%

Model, który uczył się i testował na danych bez szumów był nieznacznie lepszy od pozostałych. Zgodnie z moimi przewidywaniami, dodanie szumu tylko do danych testowych pogorszyło wyniki modelu. W przypadku szumu w danych testowych i treningowych model skorzystał z informacji zawartych w szumach danych treningowych i stał się bardziej odporny na szumy w nowych danych.

Podsumowanie

Modele sieci konwolucyjnych są skuteczne w rozpoznawaniu cech obrazów, nawet dla bardzo prostych danych takich jak zbiór MNIST. Fashion MNIST dobrze sprawdził się podczas przeprowadzania tego doświadczenia. Był wystarczająco skomplikowany by można było wyciągnąć wartościowe wnioski, ale na tyle prosty, że obliczenia trwały stosunkowo krótko.