Kacper Koźmin 268571

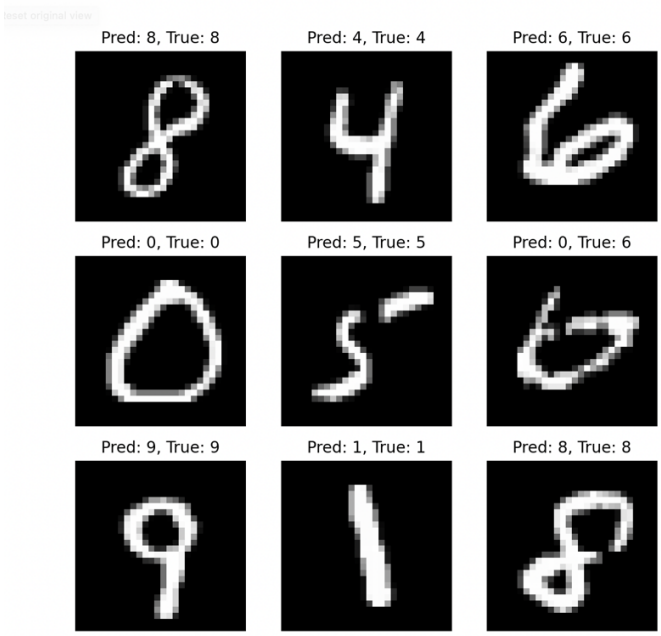
Języki programowania do zastosowań biomedycznych 11:15 – 13:00

Projekt dotyczy klasyfikacji obrazów cyfr ręcznie pisanych ze zbioru MNIST.

Opis projektu

Projekt dotyczy klasyfikacji obrazów cyfr ręcznie pisanych ze zbioru MNIST. Bazowym kodem był tutorial PyTorch dotyczący sieci neuronowych typu MLP (Multi-Layer Perceptron). Moje ulepszenia obejmowały:

- Zmianę architektury sieci neuronowej z MLP na CNN (Convolutional Neural Network), co pozwala na lepsze rozpoznawanie cech obrazu.
- Zastosowanie optymalizatora Adam zamiast domyślnego SGD.
- Modyfikację parametru weight decay dla optymalizatora, aby zwiększyć regularyzację.
- Edycję liczby oraz rozmiaru filtrów w warstwach konwolucyjnych w celu poprawy dokładności modelu.
- Dodanie wykresów prezentujących loss i accuracy w zależności od liczby epok
- Przedstawienie 9 losowych liczb z wartością przewidzianą i prawdziwą

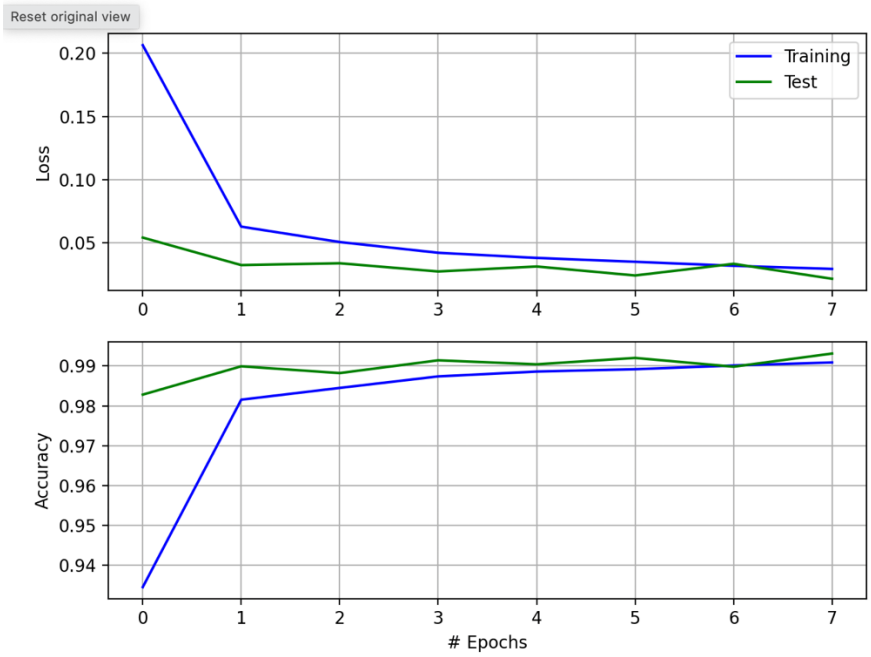


Rys. 1 Przedstawienie 9 losowych liczb z wartością przewidzianą i prawdziwą

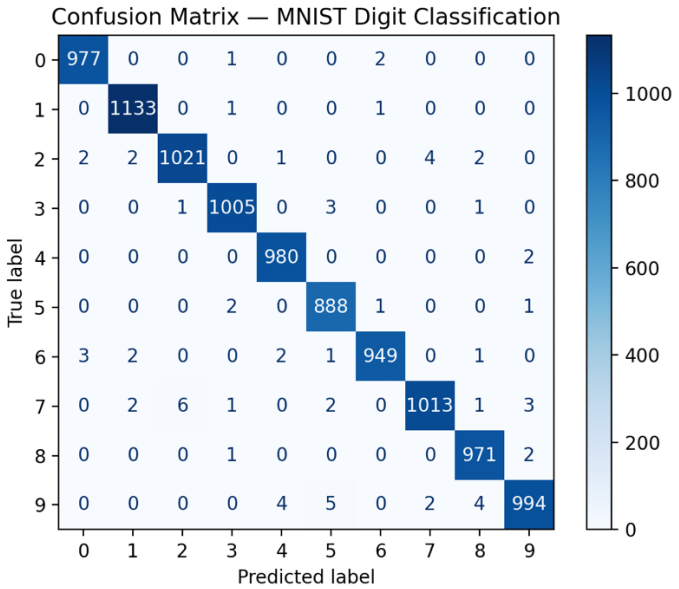
Jak widać na [Rys. 1] liczba 6 nie została poprawnie rozpoznana przez model mimo 99% skuteczności na zbiorze testowym [Rys. 2.].

```
Test Error:
 Accuracy: 99.3%, Avg loss: 0.021722
```

Rys. 2 Średnia wartość dokładności na zbiorze testowym



Rys. 3. Wykres przedstawiający stratę i dokładność modelu na zbiorze treningowym i testowym



Rys. 4. Confusion Matrix na zbiorze 10 tysięcy cyfr testowych

Wnioski

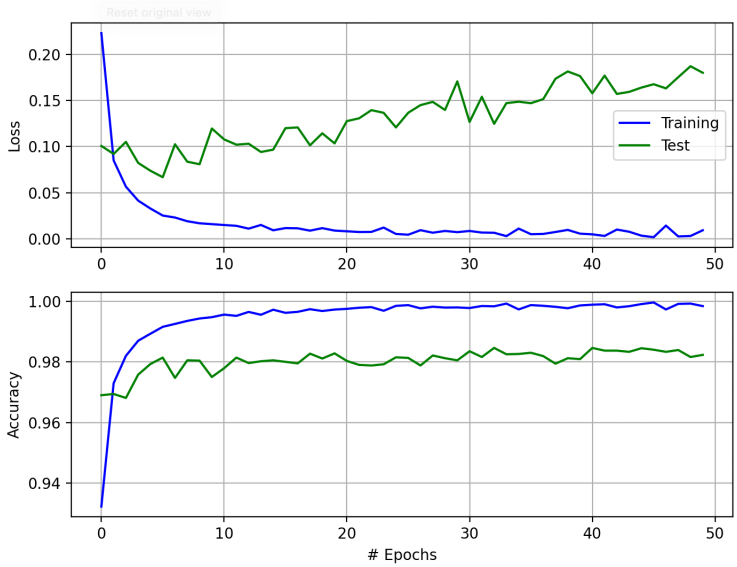
Zastosowanie architektury CNN zamiast pierwotnie zastosowanego MLP znacząco poprawiło skuteczność klasyfikacji obrazów cyfr ze zbioru MNIST, co potwierdzają wysokie wskaźniki dokładności. Optymalizator Adam oraz regularyzacja z parametrem weight decay pozytywnie wpłynęły na efektywność treningu oraz jakość generalizacji modelu.

Wykres [Rys. 3.] pokazuje, że model szybko się uczy, strata gwałtownie spada, a dokładność rośnie i osiąga około 99%. Brak dużej różnicy między zbiorami treningowym i testowym wskazuje na dobrą generalizację i brak przeuczenia. Po kilku epokach zmiany są minimalne, co sugeruje, że model osiągnął optymalną wydajność dla 8 epok. Ilość epok została dobrana metodą eksperymentalną.

Jak widać na „Macierzy pomyłek" [Rys. 4] najczęściej mylonymi liczbami były liczby „7" i „2". Najprawdopodobniej wynika to z podobieństwa w sposobie ich notacji.

W tym przypadku niższa strata walidacyjna niż treningowa wynika z zastosowania regularizacji. Używam dropoutu 30% oraz weight decay. Regularizacja działa tylko podczas treningu, wprowadzając dodatkowy szum i ograniczając możliwość nadmiernego dopasowania modelu do danych treningowych. Powoduje to wyższą stratę na zbiorze treningowym w porównaniu do testowego, gdzie model działa w trybie pełnym (bez dropoutu).

Na przykładzie widać wykres [Rys. 5.], który pokazuje przeuczenie (overfitting). Model osiąga niemal perfekcyjne wyniki na zbiorze treningowym, ale na testowym strata rośnie, a dokładność jest niższa i niestabilna. Wynika to z użycia **SGD** (Stochastic Gradient Descent) zamiast **Adam** (Adaptive Moment Estimation), **MLP** (Multilayer Perceptron) zamiast **CNN** (Convolutional Neural Network) oraz braku **dropoutu** (technika zapobiegająca przeuczeniu polegająca na losowym wyłączaniu neuronów podczas treningu) i **batch normalization** (normalizacja partii – technika przyspieszająca trenowanie i poprawiająca stabilność sieci).



Rys. 5. Wykres modelu z SGD, MLP oraz dropout i batch normalization off