Wstęp do sztucznej inteligencji

Lista 1

Aleksandra Czarniecka (272385)

marzec 2025

Wstęp

Celem laboratorium było stworzenie i wytrenowanie sieci neuronowej do rozpoznawania cyfr ze zbioru EMNIST MNIST, a także analiza jej skuteczności na zbiorze testowym oraz własnym zestawie próbek pisma. Dodatkowo przewidziano implementację klasyfikatora opartego o Random Forest.

1 Zadanie 1: Sieć neuronowa

1.1 Implementacja

Sieć neuronowa została zaimplementowana w języku Python z wykorzystaniem biblioteki PyTorch. Model składa się z dwóch warstw splotowych, warstw w pełni połączonych oraz warstwy dropout. Proces trenowania obejmował 5 epok.

1.2 Wyniki na zbiorze testowym

Klasa	Precyzja	Czułość	F1-score	Wsparcie
0	0.9939	0.9969	0.9954	980
1	0.9947	0.9974	0.9960	1135
2	0.9932	0.9835	0.9883	1032
3	0.9863	0.9960	0.9911	1010
4	0.9959	0.9817	0.9887	982
5	0.9899	0.9843	0.9871	892
6	0.9906	0.9906	0.9906	958
7	0.9798	0.9912	0.9855	1028
8	0.9918	0.9918	0.9918	974
9	0.9861	0.9871	0.9866	1009
Dokładność			0.9902	10000
Średnia makro	0.9902	0.9901	0.9901	10000
Średnia ważona	0.9902	0.9902	0.9902	10000

Tabela 1: Wyniki klasyfikacji na zbiorze testowym

2 Zadanie 2: Własny zbiór testowy

2.1 Opis zbioru

Stworzono własny zbiór testowy zawierający po trzy egzemplarze każdej cyfry, zapisane odręcznie (białe na czarnym tle).

2.2 Wyniki klasyfikacji

Klasa	Precyzja	Czułość	F1-score	Wsparcie
0	1.0000	0.3333	0.5000	3
1	1.0000	1.0000	1.0000	3
2	1.0000	1.0000	1.0000	3
3	0.7500	1.0000	0.8571	3
4	1.0000	1.0000	1.0000	3
5	0.5000	0.6667	0.5714	3
6	0.5000	0.3333	0.4000	3
7	1.0000	0.6667	0.8000	3
8	1.0000	1.0000	1.0000	3
9	0.6000	1.0000	0.7500	3
Dokładność			0.8000	30
Średnia makro	0.8350	0.8000	0.7879	30
Średnia ważona	0.8350	0.8000	0.7879	30

Tabela 2: Wyniki klasyfikacji na własnym zbiorze testowym

Dla lepiej napisanych cyfr (bardziej szablonowych) dokładność może wzrosnąć nawet do 0.9314/0.9750.

3 Interpretacja wyników

3.1 Wyjaśnienie metryk

- **Dokładność (accuracy)** – procent poprawnie sklasyfikowanych przykładów w całym zbiorze testowym.
- **Czułość (recall)** – zdolność modelu do wykrywania danej klasy (ile spośród wszystkich rzeczywistych przykładów danej klasy zostało poprawnie sklasyfikowanych).
- **Precyzja (precision)** – procent poprawnie sklasyfikowanych przykładów danej klasy spośród wszystkich przewidzianych jako ta klasa.
- **F1-score** – średnia harmoniczna precyzji i czułości, bardziej zrównoważona metryka skuteczności klasyfikacji.

3.2 Analiza wyników dla MNIST

Model osiągnął bardzo wysoką dokładność (99,02%) na zbiorze MNIST, co oznacza, że model prawidłowo klasyfikuje niemal wszystkie cyfry. Wysoka precyzja i czułość dla większości klas wskazują, że model dobrze rozpoznaje cyfry w standardowych warunkach. Dla MNIST wartości F1-score są również bardzo wysokie, co sugeruje, że model jest dobrze wytrenowany.

3.3 Analiza wyników na własnym zbiorze

Dokładność (accuracy) wynosi 80%, co oznacza, że model poprawnie sklasyfikował 24 z 30 próbek. Wysoka precyzja dla większości klas wskazuje, że jeśli model przewidział daną cyfrę, to zazwyczaj była to dobra predykcja. Natomiast niższa czułość dla cyfr 0, 5 i 6 sugeruje, że model często je pomijał lub mylił z innymi. Wynik F1-score dla cyfry 6 wynosi tylko 0.4, co wskazuje, że model ma trudności z jej rozpoznawaniem.

Możliwe przyczyny błędów:

- **Mała liczba próbek** – tylko 3 przykłady na cyfrę, co może nie być wystarczające do poprawnej klasyfikacji.
- **Styl pisma** – model trenował się na standardowym zbiorze MNIST, natomiast dużo zależy od tego jak zostały napisane odręczne cyfry.
- **Nieczytelność cyfr** – cyfry mogą być zapisane w sposób niestandardowy, przez co sieć je źle rozpoznaje.
- **Podobieństwo cyfr** – np. 5 może być mylone z 3 lub 6, co może tłumaczyć jej niższą precyzję i czułość.

4 Wnioski

Model działa bardzo dobrze na standardowym zbiorze MNIST, jednak jego skuteczność spadła na własnym zbiorze odręcznych cyfr.

Możliwe sposoby poprawy wyników:

- Zwiększenie liczby próbek w zbiorze testowym.
- Dodatkowe dostrojenie modelu (fine-tuning) na własnych danych.
- Zastosowanie augmentacji danych (np. rotacje, zmiany kontrastu).
- Eksperymentowanie z innymi modelami, np. bardziej zaawansowanymi sieciami neuronowymi.

5 Zadanie 3: Klasyfikator Random Forest

5.1 Implementacja

Klasyfikator Random Forest został zaimplementowany przy użyciu biblioteki scikit-learn.

5.2 Wyniki na zbiorze testowym

Klasa	Precyzja	Czułość	F1-score	Wsparcie
0	0.9681	0.9908	0.9793	980
1	0.9877	0.9930	0.9903	1135
2	0.9616	0.9709	0.9662	1032
3	0.9633	0.9624	0.9629	1010
4	0.9745	0.9725	0.9735	982
5	0.9762	0.9641	0.9701	892
6	0.9760	0.9781	0.9771	958
7	0.9715	0.9630	0.9673	1028
8	0.9617	0.9548	0.9583	974
9	0.9629	0.9524	0.9576	1009
Dokładność			0.9705	10000
Średnia makro	0.9704	0.9702	0.9703	10000
Średnia ważona	0.9705	0.9705	0.9705	10000

Tabela 3: Wyniki klasyfikacji na zbiorze testowym przy użyciu klasyfikatora Random Forest

6 Wnioski

Klasyfikator Random Forest osiągnął wysoką dokładność (0.9705) na zbiorze MNIST, skutecznie rozpoznając cyfry. Szczególnie dobrze wypadły klasy 1, 4, 5 i 6, z F1-score powyżej 0.97. Nieco niższe wyniki uzyskano dla cyfr 3, 8 i 9, ale nadal pozostają one wysokie (≥ 0.9576). Średnie F1-score (0.9703 makro, 0.9705 ważone) potwierdzają stabilność modelu. Random Forest okazał się skuteczny, choć trochę gorszy od sieci neuronowych, jest prostszy i bardziej odporny na przeuczenie (overfitting).