Politechnika Wrocławska

Sprawozdanie sieci płytkie i sieci głębokie Sieci neuronowe i neurosterowniki W12N, Automatyka i Robotyka

PROWADZĄCY **Dr inż. Piotr Ciskowski**

GRUPA **Pn 15:15**

AUTOR **Anastasiia Korobka(269460)**

1 Sieci płytkie

Sieci płytkie (ang. shallow neural networks) to modele sieci neuronowych, które składają się zazwyczaj z jednej warstwy ukrytej albo nawet bez żadnej warstwy ukrytej, oprócz warstwy wejściowej i wyjściowej. Są to najprostsze formy sieci neuronowych i są używane głównie w prostszych zadaniach, gdzie złożoność danych nie wymaga głębszych architektur.

Jak działają sieci płytkie? Warstwa Wejściowa: Pierwsza warstwa, przez którą dane są wprowadzane do modelu. Wielkość tej warstwy jest zwykle określona przez wymiary danych wejściowych.

Warstwa Ukryta (opcjonalna): W modelach płytkich, może być tylko jedna warstwa ukryta lub nawet żadna. Neurony w tej warstwie przetwarzają dane wejściowe przez ważoną sumę ich wejść, a następnie stosują funkcję aktywacji (np. ReLU, sigmoidalną, tanh) do wprowadzenia nieliniowości do procesu uczenia.

Warstwa Wyjściowa: Ostatnia warstwa modelu, która zwraca wynik. Funkcja aktywacji w tej warstwie zależy od konkretnego zadania - na przykład softmax dla klasyfikacji wieloklasowej lub sigmoid dla klasyfikacji binarnej.

Proces uczenia Uczenie sieci płytkiej, podobnie jak w bardziej złożonych sieciach, opiera się na następujących krokach:

Propagacja w przód (ang. forward propagation): Dane wejściowe są przekazywane przez sieć od warstwy wejściowej przez ewentualne warstwy ukryte do warstwy wyjściowej, gdzie jest generowane wyjście.

Obliczenie błędu: Różnica między wyjściem sieci a rzeczywistą wartością oczekiwaną (etkietą) jest obliczana za pomocą funkcji straty, np. błędu średniokwadratowego dla zadań regresji lub entropii krzyżowej dla klasyfikacji.

Propagacja wsteczna (ang. backpropagation): Algorytm, który oblicza gradient funkcji straty względem wszystkich wag w modelu, a następnie stosuje algorytm optymalizacji (np. SGD, Adam) do aktualizacji wag w celu minimalizacji straty.

Iteracja: Proces jest powtarzany (propagacja w przód i wstecz) na wielu partiach danych (batchach) przez wiele epok, aż model przestanie się znacząco poprawiać.

1.1 XOR w kontekście sieci neuronowych

XOR jest często używany jako przykład problemu, który nie jest liniowo separowalny, co oznacza, że nie można go rozwiązać przy użyciu jednowarstwowej sieci neuronowej (perceptronu). Potrzebna jest co najmniej jedna warstwa ukryta, aby sieć neuronowa mogła nauczyć się funkcji XOR, co czyni go klasycznym problemem testującym zdolność sieci do rozwiązywania złożonych problemów.

1.2 Implementacja

Inicjalizacja wag Funkcje init2 inicjalizują wagi sieci neuronowej losowymi wartościami bliskimi zeru dla dwóch warstw: warstwy ukrytej i warstwy wyjściowej. Każda warstwa otrzymuje dodatkowy neuron (bias) do poprawy zdolności uczenia się.

Funkcja aktywacji sigmoid jest funkcją aktywacji używaną w sieci, przekształcającą sumę ważoną wejść neuronów na wyjście w zakresie od 0 do 1. Jest to typowa funkcja aktywacji dla problemów klasyfikacji binarnej.

Propagacja w przód działaj2 realizuje propagację w przód dla dwóch warstw sieci:

Oblicza wyjście z pierwszej warstwy, stosując funkcję sigmoidalną. Dodaje bias do wyników pierwszej warstwy. Oblicza wynik końcowy przez kolejne zastosowanie wag i funkcji sigmoidalnej. Uczenie sieci ucz2 wykonuje proces uczenia:

Dla każdej epoki i każdego przykładu w danych: Wykonuje propagację w przód. Oblicza błąd (różnica między przewidywanym a rzeczywistym wynikiem). Wykonuje propagację wsteczną, aby dostosować wagi na podstawie gradientu błędu. Zapisuje błąd średniokwadratowy (MSE) dla monitorowania postępów.

Predykcja predict używa nauczonej sieci do przewidywania etykiet na podstawie danych testowych, aplikując obliczenia sieci z propagacji w przód i zwracając klasyfikacje.

Przygotowanie danych Dane są wczytywane, a następnie przetwarzane, aby uzyskać cechy numeryczne i kategoryczne. Dane są normalizowane i przekształcane przy użyciu odpowiednich transformacji. Podziel dane na zestawy treningowe i walidacyjne. Wykresy i analiza Rysuje wykres błędu MSE po każdej epoce. Oblicza dokładność modelu na zestawie walidacyjnym. Wyświetla macierz pomyłek, aby zobaczyć liczby fałszywie i prawdziwie pozytywnych oraz negatywnych wyników. Przygotowuje DataFrame z predykcjami i zapisuje je do pliku CSV.

1.3 Analiza wynikow

PassengerId: Identyfikator pasażera. Liczy się od 1 do 891, co oznacza, że w zbiorze danych jest 891 pasażerów.

Survived: Informuje, czy dana osoba przeżyła (1) czy nie (0). Średnia wartość wynosi około 0.384, co oznacza, że około 38.4

Pclass: Klasa biletu, która jest wskaźnikiem statusu ekonomicznego (1 = pierwsza klasa; 2 = druga klasa; 3 = trzecia klasa). Najwięcej pasażerów (491) podróżowało w trzeciej klasie.

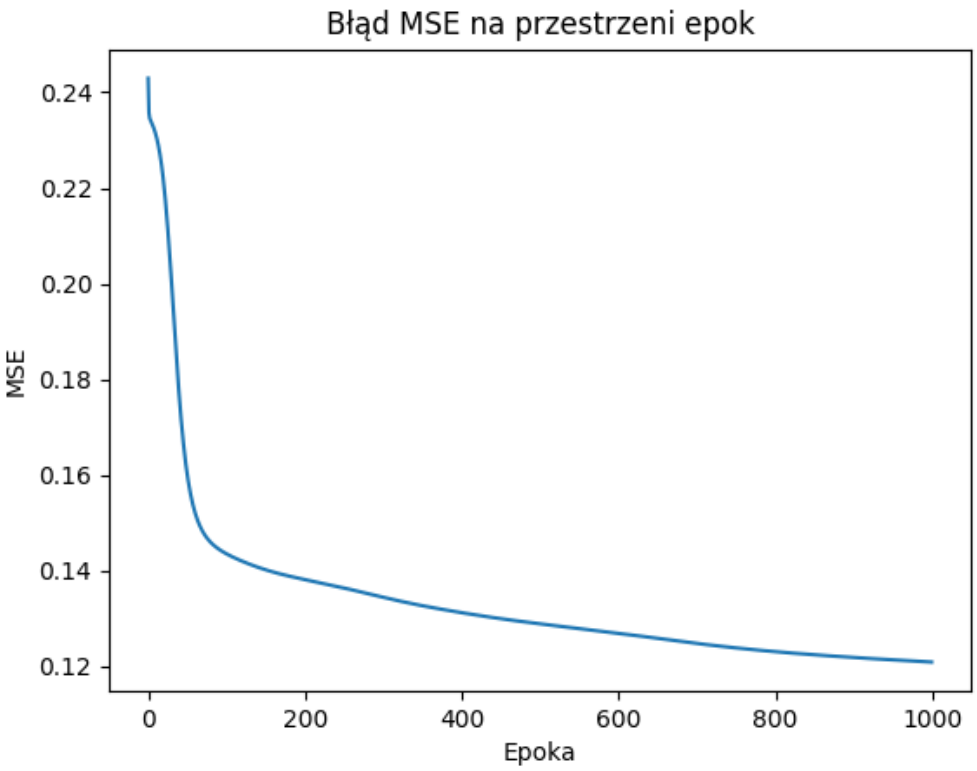
Age: Wiek pasażerów. Średni wiek wynosi około 29.7 lat, z wartościami od niemowląt (0.42 roku) do osób starszych (80 lat). Brakuje danych o wieku dla niektórych pasażerów, co jest wskazane przez liczbę 714 (nie wszyscy pasażerowie mają zarejestrowany wiek).

SibSp: Liczba rodzeństwa lub małżonków, z którymi pasażer podróżował na Titanicu. Średnio, pasażerowie mieli mniej niż jednego członka rodziny tego typu na pokładzie.

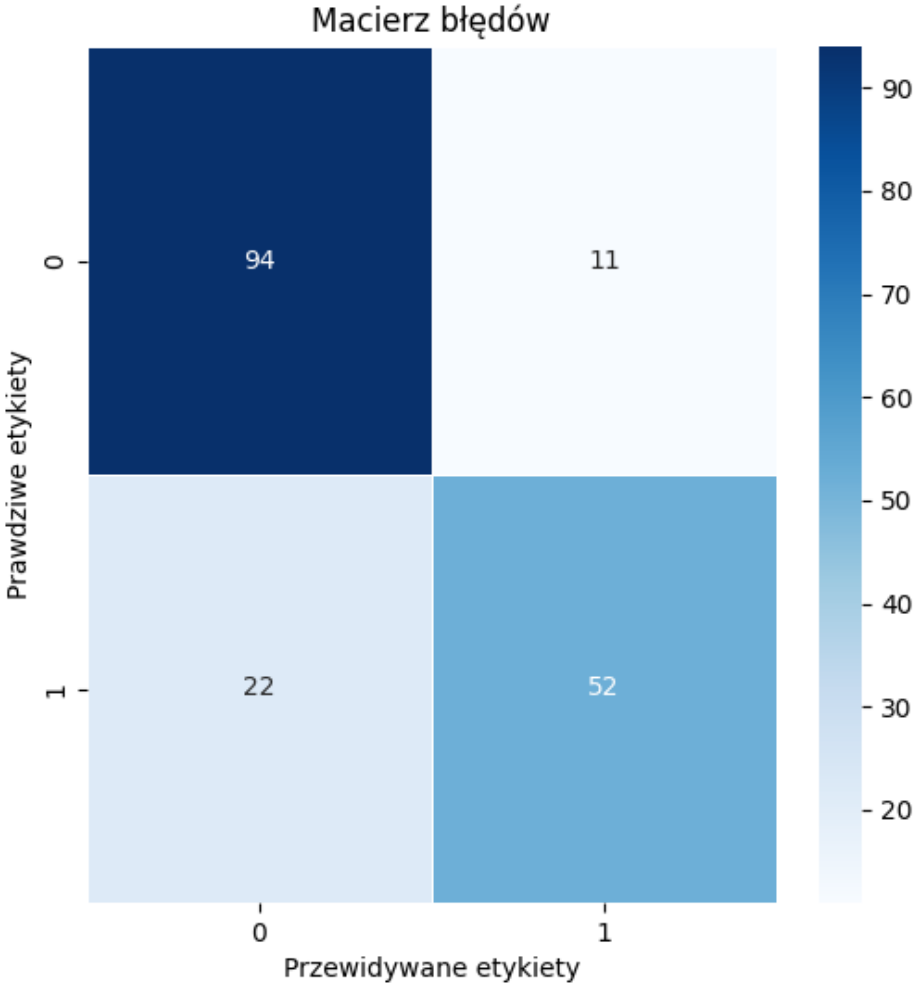
Parch: Liczba rodziców lub dzieci, z którymi pasażer podróżował. Większość pasażerów podróżowała bez rodziców lub dzieci.

Fare: Opłata za bilet. Średnia cena biletu wynosiła około 32.20, ale występowały znaczne różnice w cenach, co jest wskazane przez wysoki odchylenie standardowe oraz maksymalną opłatę 512.33.

Analiza wyników modelu Dokładność modelu: Model przewidujący, kto przeżyje, osiągnął dokładność 82Wnioski Pclass jest istotnym atrybutem, ponieważ większość pasażerów podróżowała trzecią klasą, która mogła oferować gorsze warunki przeżycia. Mniejszy procent przeżywalności (około 38Wiek i Płatność są zróżnicowane, co sugeruje, że różne demografie i ekonomiczne warstwy społeczne były obecne na statku, co może być użyteczne do dalszej analizy i modelowania. Model: Uzyskana dokładność sugeruje, że model ma potencjał do dalszych optymalizacji i być może lepszego zrozumienia czynników wpływających na przeżywalność.



Rysunek 1: Blad MSE



Rysunek 2: Macierz bledow

1.4 Wersja z wykorzystaniem Keras

Wczytywanie danych: Dane są wczytywane z pliku CSV input/train.csv, który zawiera różne atrybuty pasażerów (klasa, płeć, wiek itp.).

Definiowanie cech: Określono, które kolumny w danych będą używane jako cechy wejściowe dla modelu, w tym zarówno cechy numeryczne, jak i kategoryczne.

Przetwarzanie wstępne danych:

Dane numeryczne są przetwarzane poprzez wypełnienie brakujących wartości medianą i następnie skalowanie standardowe (odjęcie średniej i podzielenie przez odchylenie standardowe). Dane kategoryczne są przetwarzane przez wypełnienie brakujących wartości stałą wartością 'missing' i zastosowanie kodowania one-hot, co przekształca kategorie na wektor binarny. Tworzenie modelu:

Model to prosta sieć neuronowa zbudowana z warstw gęstych (Dense layers). Pierwsza warstwa ma 64 neurony i funkcję aktywacji ReLU, a druga warstwa to pojedynczy neuron z funkcją aktywacji sigmoidalną, przewidujący przeżycie (0 lub 1). Jako funkcję straty używa się binary_crossentropy, odpowiednią dla problemów klasyfikacji binarnej. Optymalizator to SGD (stochastyczny spadek gradientu). Trenowanie modelu: Model jest trenowany na danych uczących przez 100 epok, gdzie w każdej epoce aktualizowane są wagi sieci na podstawie błędu prognoz

względem rzeczywistych danych. Używane są również dane walidacyjne do monitorowania wydajności modelu na danych, których nie widział podczas treningu.

Ewaluacja modelu: Po trenowaniu modelu jego dokładność jest oceniana na danych walidacyjnych.

Predykcja i zapis do pliku CSV: Model dokonuje predykcji na danych walidacyjnych, a wyniki (razem z dodatkowymi informacjami o pasażerach, takimi jak imię, wiek, płeć i klasa) są zapisywane do pliku CSV.

Wizualizacja wyników: Na wykresach pokazane są dokładność i wartość funkcji straty modelu na danych uczących i walidacyjnych w zależności od epoki, co pozwala zobaczyć, jak model poprawiał się (lub nie) w czasie.

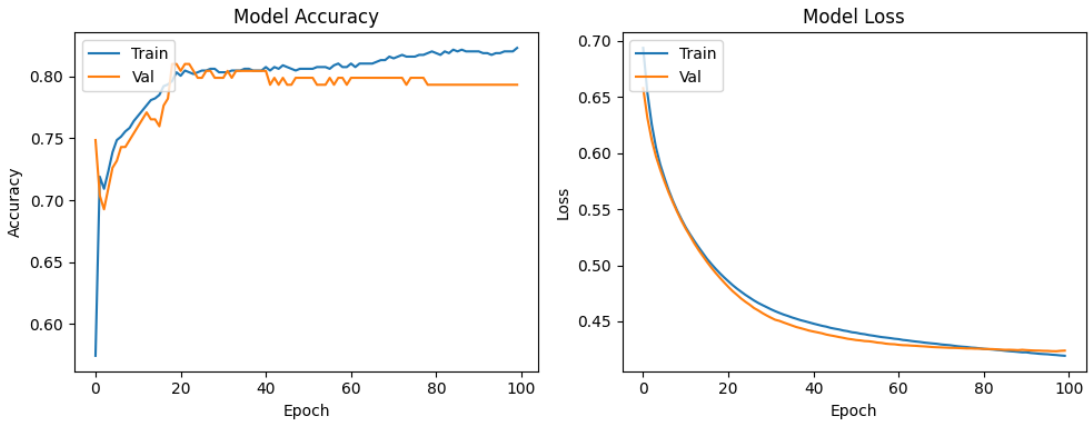
Każdy krok procesu od przygotowania danych, przez trenowanie modelu, po wizualizację i zapis wyników jest zautomatyzowany, co umożliwia łatwe przetestowanie różnych konfiguracji modelu i przetwarzania danych.

1.5 Analiza wynikow

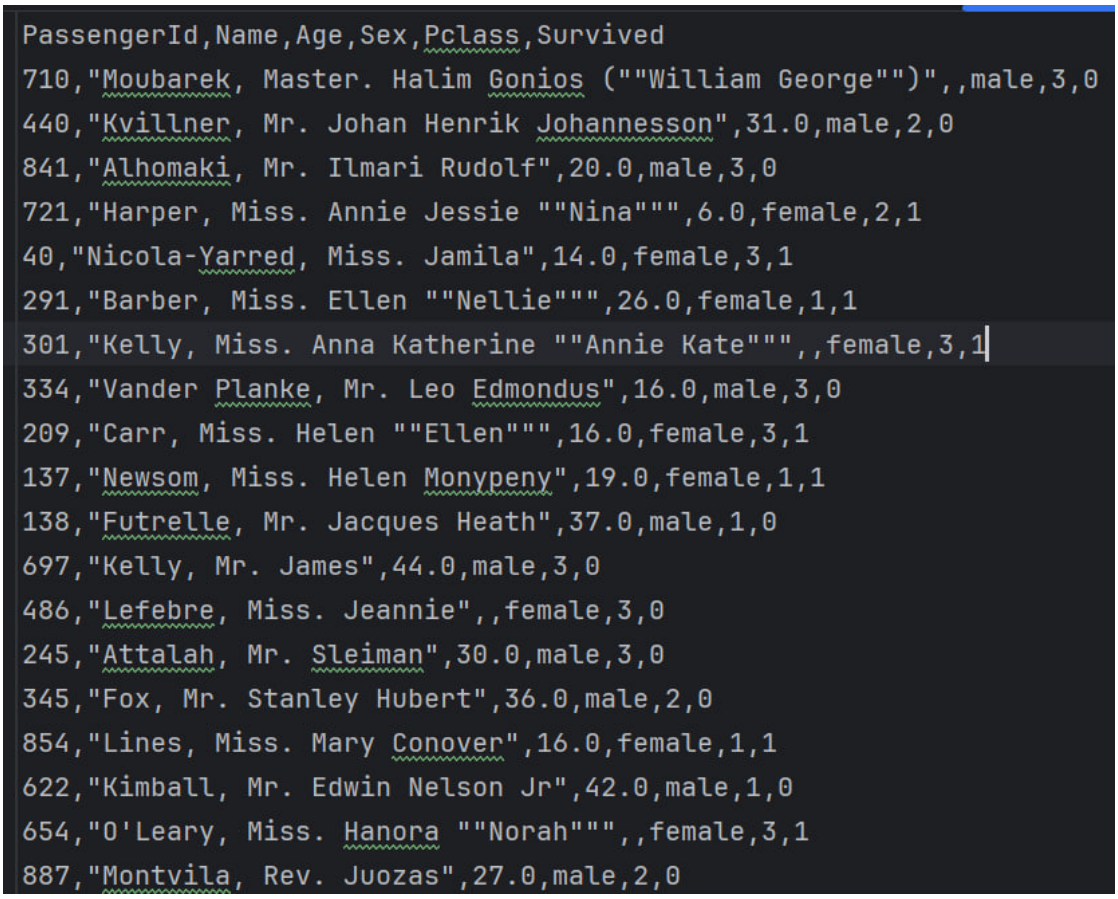
Proces trenowania:

W każdej epoce (iteracji) model jest trenowany na danych treningowych (X_train, y_train) i jego wyniki są oceniane na danych walidacyjnych (X_val, y_val). Metryki takie jak dokładność (accuracy) i strata (loss) są rejestrowane. Model poprawia swoje wagi, aby zminimalizować stratę, co prowadzi do lepszej dokładności w kolejnych epokach. Dokładność (accuracy) pokazuje, jaki procent predykcji modelu był poprawny. Strata (loss) mierzy, jak daleko przewidywania modelu są od faktycznych wartości; niższa strata wskazuje na lepsze dopasowanie modelu. Wyniki:

Validation accuracy: 0.8: Pokazuje, że po nauczeniu modelu, dokładność na zbiorze walidacyjnym wynosi około 80Wyniki trenowania są dodatkowo wizualizowane przez wykresy pokazujące zmiany dokładności i straty w trakcie kolejnych epok, co pozwala na wizualną ocenę postępów modelu i potencjalnego overfittingu czy underfittingu. Wyniki te wskazują na generalną skuteczność modelu w przewidywaniu przeżycia na podstawie danych z Titanic, chociaż zawsze istnieje przestrzeń do dalszego optymalizowania modelu, na przykład przez regularyzację, zmianę architektury sieci czy dostosowanie parametrów trenowania.



Rysunek 3



Rysunek 4: Predykcje modelu

2 sieci głębokie

Sieci głębokie, znane również jako głębokie sieci neuronowe (DNN - Deep Neural Networks), są szczególnym rodzajem sieci neuronowych, które zawierają wiele warstw między warstwą wejściową a wyjściową. Te dodatkowe warstwy, zwane ukrytymi warstwami, umożliwiają sieciom głębokim uczenie się bardziej złożonych wzorców i reprezentacji danych, co sprawia, że są one szczególnie efektywne w rozwiązywaniu trudnych problemów związanych z przetwarzaniem obrazów, dźwięku, wideo czy języka naturalnego.

Jak działają sieci głębokie? Warstwy: Sieci głębokie składają się z wielu warstw, z których każda przetwarza wejście od poprzedniej warstwy, dokonuje pewnych obliczeń, a następnie przekazuje wynik do następnej warstwy. Typowe warstwy w sieciach głębokich to:

Warstwy konwolucyjne (Convolutional Layers): Przetwarzają obrazy za pomocą filtrów, które automatycznie wykrywają cechy takie jak krawędzie, kolory czy tekstury. Warstwy rekurencyjne (Recurrent Layers): Są stosowane w przetwarzaniu sekwencji, np. w analizie języka naturalnego czy danych czasowych. Przechowują informacje o poprzednich stanach, co pozwala na analizę zależności w czasie. Warstwy ukryte (Hidden Layers): Te warstwy nie mają bezpośredniego kontaktu z warstwami wejściowymi lub wyjściowymi i przyczyniają się do głębszego przetwarzania cech. Aktywacja: Funkcje aktywacji, takie jak ReLU (Rectified Linear Unit) lub sigmoid, są stosowane w każdej warstwie, aby wprowadzić nieliniowość do procesu uczenia, co pozwala modelom na uczenie się bardziej skomplikowanych wzorców.

Strata i Optymalizacja: Sieci uczą się poprzez minimalizację funkcji straty, która mierzy różnicę między przewidywaniami a rzeczywistymi wartościami. Proces ten wykorzystuje algorytmy optymalizacyjne, takie jak SGD (Stochastic Gradient Descent) czy Adam, które dostosowują wagi w sieci, aby zminimalizować stratę.

Wsteczna propagacja (Backpropagation): To kluczowa technika w procesie uczenia, która pozwala na efektywne obliczanie gradientów funkcji straty względem każdej wagi w sieci, a następnie odpowiednie dostosowanie wag.

Ładowanie danych: Dane MNIST są ładowane z tensorflow.keras.datasets.mnist, gdzie są one podzielone na zestawy treningowe i testowe.

2.1 Dzialanie programu

Przygotowanie danych:

Obrazy są przeskalowane do zakresu [0,1] przez podzielenie przez 255.0 (normalizacja). Etykiety są konwertowane na one-hot encoding, co oznacza, że każda etykieta zostaje przekształcona w 10-elementowy wektor binarny. W danych treningowych 50% etykiet zostaje zastąpionych losowymi etykietami, co jest eksperymentem badającym odporność sieci na nieprawidłowe dane. Definicja modeli:

create_mlp_model: Tworzy prosty model perceptronu wielowarstwowego (MLP) z jedną warstwą ukrytą. create_cnn_model: Tworzy bardziej złożony model sieci konwolucyjnej (CNN), który jest bardziej odpowiedni dla przetwarzania obrazów. Kompilacja modeli: Modele są kompilowane z optymalizatorem Adam i funkcją straty categorical_crossentropy, co jest typowe dla zadań klasyfikacyjnych z wieloma klasami.

Trenowanie modeli: Modele są trenowane na danych treningowych, z wykorzystaniem danych walidacyjnych do monitorowania postępów. Użyto mechanizmów takich jak EarlyStopping i ModelCheckpoint do zatrzymywania trenowania, gdy model przestaje się poprawiać.

Wyświetlanie predykcji: Funkcja display_predictions pokazuje, jak model radzi sobie z przewidywaniem na przykładowych obrazach z zestawu testowego. Wyświetlane są obrazy wraz z przewidywanymi i rzeczywistymi etykietami.

Wykresy dokładności i strat: Na koniec, kod generuje wykresy pokazujące dokładność i stratę modelu na danych treningowych i walidacyjnych w trakcie każdej epoki trenowania.

2.2 Analiza wynikow

Porównując efektywność dwóch modeli sieci neuronowych na przedstawionych wykresach dokładności i strat, widzimy następujące różnice między modelem wielowarstwowego perceptronu (MLP) a modelem sieci konwolucyjnej (CNN):

Dokładność (Accuracy):

MLP: Dokładność treningowa i walidacyjna wydaje się być względnie stabilna, gdzie dokładność walidacyjna jest konsekwentnie niższa niż treningowa. Widać, że model MLP osiągnął stosunkowo wysoką dokładność, ale różnica między dokładnością treningową a walidacyjną wskazuje na pewien poziom przetrenowania (overfitting).

CNN: Znaczna poprawa dokładności treningowej przez cały proces uczenia, przy jednoczesnym zwiększeniu dokładności walidacyjnej. CNN wykazuje lepsze ogólne wyniki, osiągając wyższą dokładność walidacyjną niż MLP. To wskazuje na lepsze uogólnienie modelu na nieznane dane. Strata (Loss):

MLP: Strata na zbiorze treningowym i walidacyjnym stopniowo spada, ale tempo zmniejszania się straty zwalnia w miarę postępów w epokach. Istnieje wyraźna różnica między stratą treningową a walidacyjną, co ponownie wskazuje na overfitting.

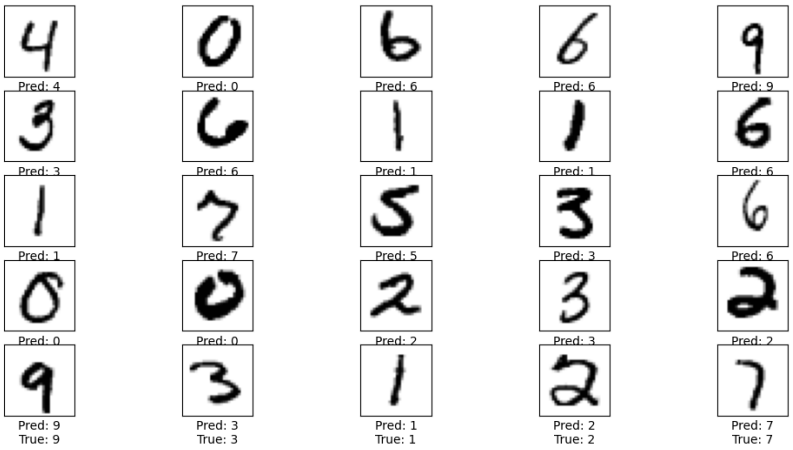
CNN: Strata zarówno na treningu, jak i walidacji spada w sposób bardziej regularny i spójny. Mniejsza luka między stratą treningową a walidacyjną sugeruje, że CNN jest bardziej odporny na przetrenowanie w porównaniu do MLP i lepiej generalizuje na nieznane dane. Wizualizacja Predykcji:

Predykcje: Przykłady predykcji dla obu modeli pokazują, jak oba modele radzą sobie z klasyfikacją cyfr ręcznie pisanych. CNN zazwyczaj osiąga lepszą dokładność w rozpoznawaniu bardziej złożonych wzorców i cyfr, co jest widoczne na ostatnich obrazach.

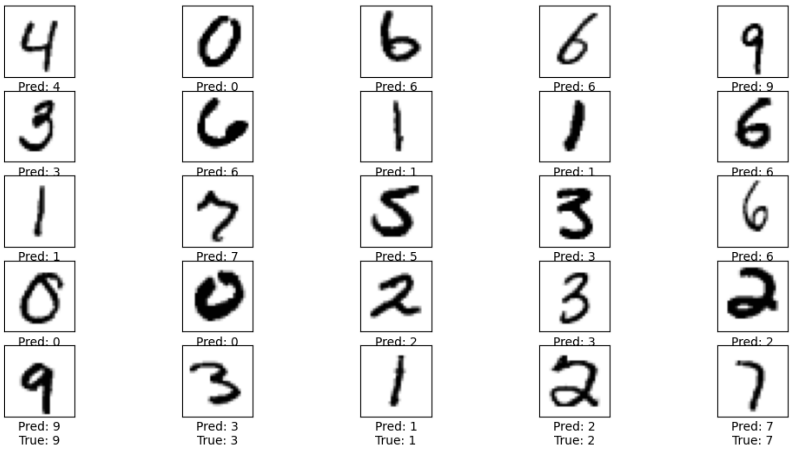
Podsumowanie

CNN generalnie wypada lepiej niż MLP w zadaniach związanych z przetwarzaniem obrazów, co wynika z ich zdolności do efektywnego wyłapywania lokalnych wzorców i cech na obrazach, co jest kluczowe w przypadku zadania takiego jak MNIST. MLP, mimo że jest prostszy do

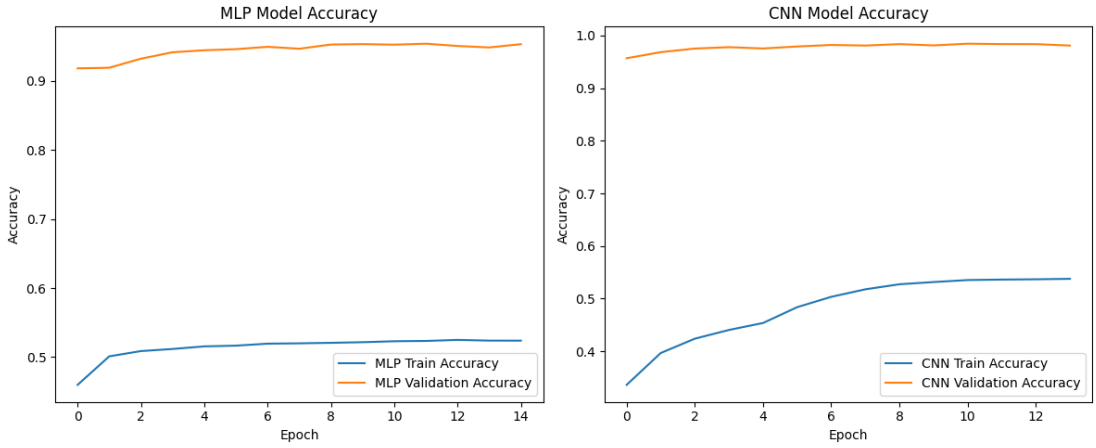
implementacji i trenowania, może nie być wystarczająco skuteczny w przypadku bardziej złożonych wzorców obrazowych, co jest widoczne zarówno na wykresach dokładności i strat, jak i na przykładach predykcji.



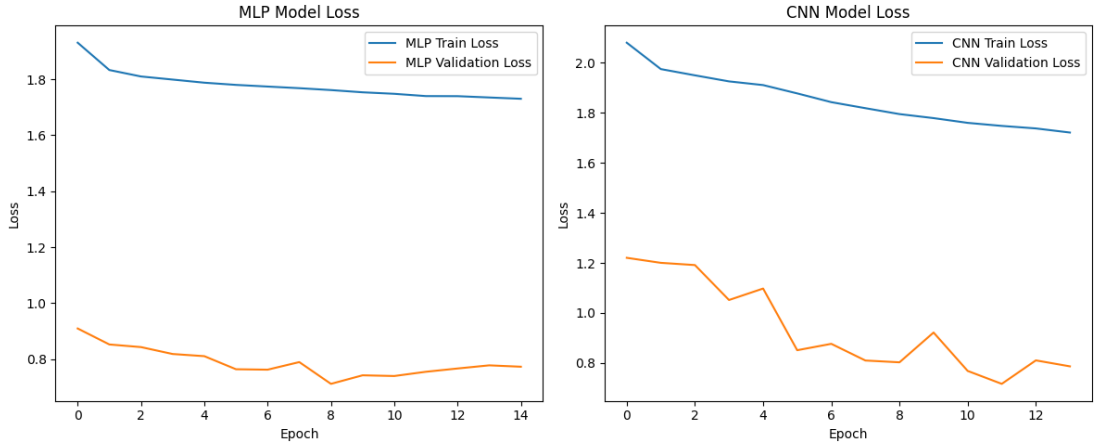
Rysunek 5



Rysunek 6



Rysunek 7



Rysunek 8