Zadanie: Analiza Danych Temperaturowych z Użyciem TensorFlow i Warstw Konwolucyjnych

Cel:

Celem zadania jest stworzenie modelu sieci neuronowej w TensorFlow, który będzie analizował dane temperatury, wykorzystując warstwy konwolucyjne oraz regularyzację L2, i dokonywał prognoz temperatury na przyszłość. Model będzie trenowany na danych historycznych, a następnie wykorzystany do przewidywania przyszłych wartości temperatury. Na końcu zostanie przeprowadzona analiza wyników i wizualizacja danych.

Krok 1: Przygotowanie danych

Na początku należy załadować dane dotyczące temperatury. Zakładając, że dane są w formacie CSV, możemy je załadować za pomocą biblioteki pandas. Dane powinny zawierać przynajmniej jedną kolumnę z wartościami temperatury w czasie. Dla uproszczenia, możemy wygenerować dane na podstawie funkcji sinusoidalnej, aby symulować zmiany temperatury w czasie.

Dalszym krokiem jest przygotowanie odpowiednich danych wejściowych do sieci neuronowej. W tym przypadku wykorzystamy dane z ostatnich 30 dni do przewidywania temperatury w kolejnym dniu. Można to zrealizować za pomocą funkcji create_dataset(), która przygotowuje dane w formie odpowiedniej dla modelu.

Krok 2: Budowa modelu sieci neuronowej

W tym kroku stworzymy model sieci neuronowej z warstwami konwolucyjnymi. Model powinien zawierać:

1. **Warstwy konwolucyjne**:
    - Pierwsza warstwa konwolucyjna z 64 filtrami o rozmiarze 3 i funkcją aktywacji ReLU.
    - Druga warstwa konwolucyjna z 32 filtrami o rozmiarze 3 i funkcją aktywacji ReLU.

W obu warstwach zastosujemy regularyzację L2, aby uniknąć przeuczenia modelu.

2. **Warstwa spłaszczająca**:
    - Po warstwach konwolucyjnych dane zostaną spłaszczone do jednego wektora.
3. **Warstwa gęsta (Dense)**:
    - Dodajemy warstwę gęstą z 50 neuronami i funkcją aktywacji ReLU.
4. **Warstwa wyjściowa**:
    - Warstwa wyjściowa z jednym neuronem, który będzie przewidywał temperaturę.

Krok 3: Kompilacja modelu

Model zostanie skompilowany z użyciem optymalizatora Adam oraz funkcji straty mean_squared_error, ponieważ przewidujemy wartości ciągłe. Model zostanie także zweryfikowany na danych walidacyjnych podczas treningu.

Krok 4: Trening modelu

Model będzie trenowany na danych temperatury przez określoną liczbę epok (np. 10) i batch size (np. 32). Podczas treningu zostaną wyświetlone dane dotyczące strat (loss) na danych treningowych i walidacyjnych.

Krok 5: Zapis modelu

Po zakończeniu treningu model zostanie zapisany w pliku .h5 za pomocą funkcji model.save(). Taki plik może być później załadowany i używany do dalszych prognoz.

Krok 6: Predykcja i wizualizacja wyników

Po zakończeniu treningu modelu, będziemy mogli dokonać prognoz na podstawie danych testowych. Wyniki te zostaną porównane z rzeczywistymi wartościami temperatury, a także wizualizowane na wykresie.

- Na wykresie zostaną przedstawione zarówno rzeczywiste dane temperatury, jak i przewidywane przez model.
- Analiza wyników będzie polegała na obliczeniu błędu średniokwadratowego (MSE) przy użyciu funkcji mean_squared_error z biblioteki sklearn.metrics.

Krok 7: Prognozowanie temperatury na przyszłość

Na końcu będziemy mogli przewidywać temperaturę na przyszłe okresy czasowe. W tym celu będziemy korzystać z ostatnich dostępnych danych (np. ostatnich 30 dni), a następnie generować prognozy na kolejne dni. Dane wejściowe będą przesuwane, aby na każdym etapie wprowadzać najnowszą przewidywaną temperaturę jako część danych wejściowych.

Krok 8: Wizualizacja prognoz na przyszłość

Na zakończenie wygenerujemy wykres, który przedstawi rzeczywiste temperatury, a także przewidywane temperatury na przyszłe okresy czasowe. Dzięki temu będzie można ocenić, jak dobrze model przewiduje zmiany temperatury w kolejnych dniach.