Projektowania i zastosowania sieci neuronowych

Temat projektu: Implementacja autoenkodera do rekonstrukcji obrazów z wykorzystaniem konwolucji.

Lista wykonawców:

- Julia Rojek 272529
- Emilia kowalczyk 272539

Cel projektu:

Celem projektu jest stworzenie modelu autoenkodera do kompresji i rekonstrukcji obrazów. Model ten będzie wykorzystywał warstwy konwolucyjne do ekstrakcji cech z obrazów oraz transponowane konwolucje (dekonwolucje) do ich rekonstrukcji. Projekt ma na celu:

1. Zrozumienie i implementację struktury autoenkodera od podstaw w Pythonie.
2. Opracowanie algorytmów do kompresji obrazów i ich rekonstrukcji.
3. Eksperymentowanie z różnymi hiperparametrami w celu optymalizacji modelu.

Harmonogram:

1. Przygotowanie danych (wczytanie, normalizacja, transformacje).

   Wczytanie, normalizacja, transformacje (zmiana rozmiaru, obrót itp.),podział danych na zbiór treningowy i testowy, utworzenie DataLoadera

2. Implementacja struktury autoenkodera (enkoder, dekoder).
   a. Warstwy konwolucyjne (dla enkodera).
   b. Warstwy transponowanej konwolucji (dekonwolucji) (dla dekodera).
   c. Funkcje aktywacji (ReLU, Sigmoid).
   d. Funkcję forward, która łączy enkoder i dekoder.
   e. Propagację wsteczną, aby model mógł uczyć się z danych.
3. Trenowanie modelu (funkcja kosztu, optymalizacja, pętla treningowa).
   Implementacja:
   a. Funkcji kosztu (MSE) do oceny jakości rekonstrukcji.
   b. Optymalizacji (Gradient Descent), aby zaktualizować wagi modelu na podstawie gradientów obliczonych w czasie propagacji wstecznej.
   c. Pętlli treningowej, która wykonuje kilka epok, w których model jest uczony na mini-batchach danych.
   d. Zapis wytrenowanego modelu, aby przechować wagi po zakończeniu treningu.
   e. Ewaluację modelu na danych testowych, aby sprawdzić jakość rekonstrukcji na niewidzianych danych

4. Optymalizacja modelu, testowanie, zapis modelu, wizualizacja wyników.
    a. Optymalizacja modelu, w tym eksperymentowanie z hiperparametrami (np. zmiana liczby warstw, funkcji aktywacji, algorytmu optymalizacji).
    b. Testowanie modelu na danych testowych, aby ocenić jego zdolność do generalizowania na nowych danych.
    c. Zapis modelu po treningu, aby móc załadować model i wykorzystać go do dalszej pracy lub predykcji.
    d. Wizualizacja wyników, aby ocenić jakość rekonstrukcji i zobaczyć, jak dobrze model radzi sobie z obrazami.

Kamienie milowe:

1. Przygotowanie danych (wczytanie, normalizacja, transformacje).
2. Implementacja struktury autoenkodera (enkoder, dekoder).
3. Trenowanie modelu (funkcja kosztu, optymalizacja, pętla treningowa).
   Implementacja:
4. Optymalizacja modelu, testowanie, zapis modelu, wizualizacja wyników.

Proponowanie narzędzia:

- Język programowania: Python

- Dane do testowania modelu: Dataset z Kaggle

Bibliografia:

https://www.mygreatlearning.com/blog/autoencoder/