Nasz projekt skupia się na klasyfikacji ubrań ze względu na ich kategorie. W celu realizacji tego zadania wykorzystałyśmy konwolucyjne sieci neuronowe. Charakteryzują się one zdolnością do automatycznego ekstraktowania cech z obrazów poprzez warstwy konwolucyjne, co czyni je idealnymi do analizy danych obrazowych.

W naszym projekcie przedstawiamy krok po kroku proces budowy modeli CNN do klasyfikacji ubrań oraz wypróbujemy trzy różne architektury w celu otrzymania jak najlepszych wyników.

Do naszego projektu wykorzystałyśmy zbiór Fashion MNIST, który składa się z 70,000 obrazów, nie posiadający brakujących wartości.

Zbiór danych zawiera 10 równomiernie rozłożonych klas ubrań: T-shirt/top, Spodnie, Sweter, Sukienka, Płaszcz, Sandały, Koszula, Buty sportowe, Torba, Botki. Każdy obrazek ma wymiary 28x28 pikseli i jest w skali szarości. Dane te zostały opracowane przez firmę Zalando w celu stworzenia bardziej złożonego zadania klasyfikacji obrazów niż klasyczny zbiór MNIST, który zawierał ręcznie pisane cyfry.

# przygotowanie danych

Importujemy zbiór danych FashionMnist

Dzielimy dane na zbiór walidacyjny i treningowy (10 000 probek używamy jako zbior walidacyjny a pozostała część jako zbiór treningowy)

W celu zapobiegnięcia przeuczeniu modelu wykonujemy augmentacje na danych, w naszym przypadku wykonałyśmy flipy horyzontalne.

Teraz tworzymy data loadery,

Data loader będzie tworzyl 64 probki w każdej iteracji. w celu uniknięcia systematycznych wzorców w zbiorze treningowym dane będą tasowane na początku każdej epoki.

#modele

Modele zostały zaimplementowane przy użyciu PyTorch, popularnej biblioteki do uczenia maszynowego. Wszystkie treningi przeprowadziłyśmy z użyciem GPU w celu przyspieszenia uczenia.

Architektura 1 ma dwie warstwy konwolucyjne, pierwsza z kanalem wejściowym rownym 1 (obrazki są czarnobiałe) i wyjściowym 64 oraz druga z kanałem wyjściowym rownym 128

Rozmiar jądra ustawiamy na 5, padding będzie równy 2 ( czyli dodajemy 2 piksele zerowe wokół obrazu w celu zaxhowania jego wymiarow). Uzyjemy funkcji aktywacji ReLU. W celu zmniejszenia przetrenowania ustawiłyśmy dropout na 0.05.

Dodajemy warstwę maxpooling z kernelsize=2.

Następnie dane z wielu kanałów ulegają spłaszczeniu do jednowymiarowego wektora o rozmiarze 6272. W dalszej części znajdują się warstwy gęsto połączone (MLP) o liczbie neuronów kolejno 1024 i 10, gdyż taka jest liczba rozpoznawanych klas

Dodajemy funkcje aktywacji relu

Przenosimy model

Uzywamy optymalizatora adamw z regularyzacja wag rowna 0.05 i learning ratem 0.01

Trenujemy model, ustawiamy losowe ziarno, model trenujemy przez 20 epok. Otrzymujemy wynikokolo 96,29 % dokladnosci.

Dla osiagniecia lepszego wyniku wytrenujemy ten sam model jednak z zmniejszonym lr do 0.0005. Otrzymujemy lepszy wynik rowny 96,94 %.