Program do wykrywania dojrzałości owoców

Jakub Matuszek
Przemysław Marciniec

Na czym bazuje nasz program?

Konwolucyjne sieci neuronowe (CNN)

CNN to algorytm, który może pobrać obraz wejściowy i sklasyfikować go wedle predefiniowanych kategorii (np. rasy psa). Sieci konwolucyjne poprzez trening są w stanie nauczyć się, jakie cechy szczególne obrazu pomagają w jego klasyfikacji. Ich przewagą nad standardowymi sieciami głębokimi jest większa skuteczność w wykrywaniu zawiłych zależności w obrazach. Jest to możliwe dzięki zastosowaniu filtrów badających zależności pomiędzy sąsiednimi pikselami.

TensorFlow

TensorFlow to otwarta biblioteka programistyczna stworzona przez zespół Google Brain w celu ułatwienia tworzenia modeli uczenia maszynowego. Jest jednym z najpopularniejszych narzędzi wykorzystywanych w dziedzinie uczenia maszynowego i deep learningu. Ułatwia tworzenie zaawansowanych modeli uczenia maszynowego. Dzięki swojej elastyczności, skalowalności i wsparciu społeczności, jest szeroko stosowana w różnych dziedzinach i branżach, przyczyniając się do rozwoju sztucznej inteligencji i innowacji technologicznych.

Keras

Keras to popularna biblioteka służąca do deep learning'u napisana w języku Python. Została stworzona w celu ułatwienia tworzenia i eksperymentowania z sieciami neuronowymi. Keras został stworzony jako interfejs wysokiego poziomu do biblioteki TensorFlow, choć od wersji TensorFlow 2.0 stał się również częścią oficjalnego API TensorFlow. Najważniejsze cechy tej biblioteki są takie, że jest ona prosta w użyciu, modularność czyli składowanie warstw (takich jak konwolucyjne, rekurencyjne, gęste itd...), wieloplatformowość, wsparcie dla wielu zastosowań oraz rozszerzalność która umożliwia dostosowywanie własnych warstw, loss functions, metrics i innych komponentów.

Klasyfikacja obrazów

Czym ona tak naprawdę jest?

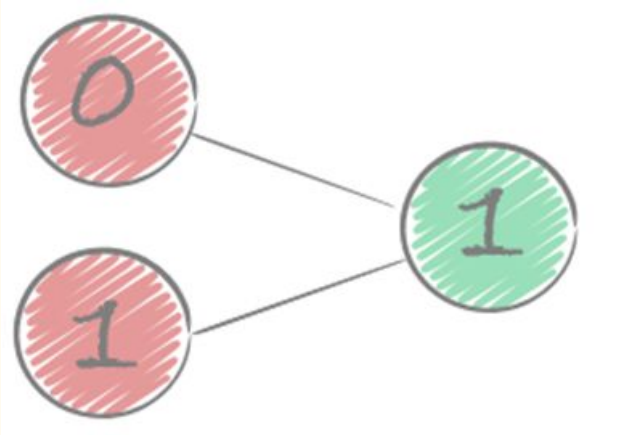
Klasyfikacja obrazów to proces przyporządkowywania obrazów do określonych **kategorii** lub **klas** na podstawie ich **cech** i **wzorców**. Wykorzystuje się różne techniki i algorytmy, w tym uczenie maszynowe, aby nauczyć system rozpoznawać i odróżniać różne obiekty, wzory i cechy na obrazach, w naszym przypadku owoce.

Przykładowe wykorzystania to np. szerokie zastosowanie w medycynie podczas wykrywania raka. Algorytmy uczą się rozpoznawać cechy charakterystyczne guzów na obrazach medycznych, takich jak mammografia czy tomografia komputerowa. Dzięki temu można skutecznie wspomagać diagnozę i wczesne wykrywanie chorób.

Neurony i tensory

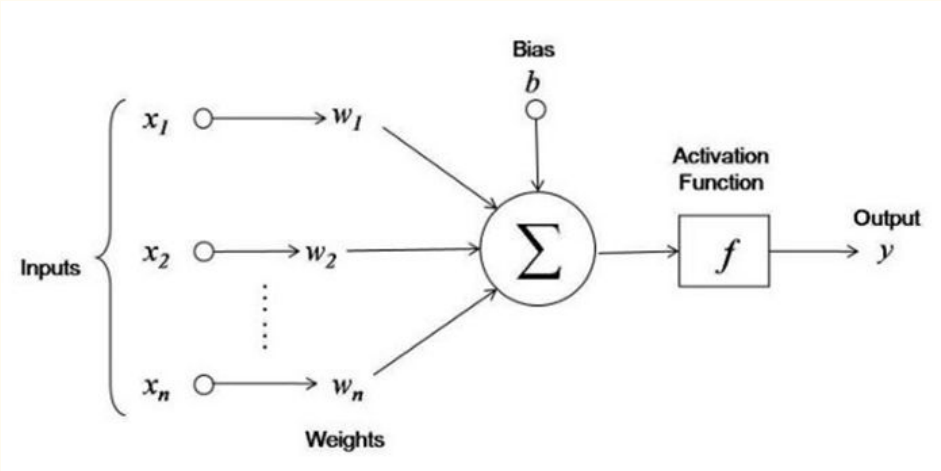
Neuron

Uczeni zaproponowali bardzo prosty model matematyczny, który nazwano sztucznym neuronem (ang.neuron). Miał on co najmniej jedno wejście binarne (0/1 albo prawda/fałsz) i tylko jedno binarne wyjście. Wyjście zostaje uaktywnione jeśli osiągnięta została określona ilość wejść. Taki układ jest niewiele lepszy od zwykłych bramek logicznych. Twórcy pokazali, że nawet dla tak uproszczonego modelu możliwe jest dla sieci neuronowej (mnóstwo neuronów połączonych ze sobą) rozwiązanie dowolnego zagadnienie logicznego.



Czym jest perceptron?

Aktualnie stosuje się inny rodzaj sztucznych neuronów zwany perceptronem, który daje odpowiedź w formie liczby rzeczywistej. Pojedynczy perceptron jest jedną z najprostszych sieci neuronowych zaproponowaną przez Franka Rosenblatta w 1957 roku. Frank wziął prosty sztuczny neuron binarny i go odrobinę zmodyfikował:



Multi Layer Perceptron (MLP)

MLP to rodzaj sztucznej sieci neuronowej, który jest powszechnie używany w dziedzinie uczenia maszynowego i deep learning'u. Jest to najprostszy rodzaj sieci neuronowej o strukturze **warstwowej**.

MLP składa się z co najmniej trzech warstw: warstwy **wejściowej**, warstw **ukrytych** i warstwy **wyjściowej**. Warstwa wejściowa przyjmuje dane wejściowe, które są przekazywane przez sieć neuronową. Warstwy ukryte są odpowiedzialne za przetwarzanie danych wejściowych poprzez stosowanie operacji matematycznych na wagach połączeń między neuronami. Każdy neuron w danej warstwie ukrytej otrzymuje wejścia od wszystkich neuronów w poprzedniej warstwie ukrytej lub warstwie wejściowej. Warstwa wyjściowa generuje końcowe wyniki lub predykcje na podstawie przetworzonych informacji.

Tensor

Tensor w deep learning'u to po prostu wielowymiarowa tablica liczb. Można go sobie wyobrazić jako kontener, który przechowuje dane numeryczne. Tensory są podstawowym elementem, na którym operuje się podczas trenowania sieci neuronowych. Tensory są ważne, ponieważ pozwalają na przechowywanie i przetwarzanie danych w wielu wymiarach jednocześnie. Sieci neuronowe operują na tensorach, wykonując na nich różne operacje matematyczne, takie jak dodawanie, mnożenie i kombinacje tych operacji. Dzięki temu sieci neuronowe są w stanie modelować złożone zależności między danymi i wykonywać różne zadania, takie jak rozpoznawanie obrazów czy tłumaczenie języka naturalnego.

Konwolucja, RGB

Konwolucja (obrazowo)

Konwolucja jest matematyczną operacją, która polega na przetwarzaniu obrazów za pomocą tzw. **filtrów konwolucyjnych**. Filtry te są małymi macierzami, które przesuwane są po obrazie i dokonują operacji mnożenia i sumowania pikseli.

Proces konwolucji polega na przemnożeniu wartości pikseli obrazu wejściowego przez odpowiednie wartości filtru konwolucyjnego, a następnie zsumowaniu wyników. Ta operacja jest powtarzana dla każdej lokalnej sekcji obrazu, przesuwając filtr na całą powierzchnię obrazu.

Po co stosuje się operację konwolucji?

W efekcie konwolucji otrzymuje się nowy obraz, zwany **mapą cech** lub **mapą konwolucji**. Ta mapa cech zawiera informacje o występowaniu różnych cech, takich jak **krawędzie**, **tekstury**, czy **wzorce** w analizowanym obrazie. W deep learningu, konwolucje są stosowane w warstwach konwolucyjnych sieci neuronowych, które uczą się wykrywać lub wyodrębniać istotne cechy z obrazów.

RGB, tablica NumPy

Komputery przetwarzają obrazy w formacie RGB (Red, Green, Blue) przy użyciu tablicy Numpy, która jest popularnym narzędziem do manipulacji danymi numerycznymi w języku Python. Tablica Numpy może przechowywać dane obrazu w postaci wielowymiarowej.

W przypadku obrazu RGB, tablica Numpy będzie miała trzy wymiary: **wysokość**, **szerokość** i **głębokość kanałów kolorów**. Wysokość i szerokość odnoszą się do rozmiarów obrazu, a głębokość określa liczbę kanałów kolorów, która wynosi 3 dla obrazów RGB.

Jak to działa w praktyce?

Każdy element tablicy Numpy reprezentuje wartość piksela obrazu. Dla obrazu RGB, każdy piksel zawiera trzy wartości, odpowiednio dla składowych Red, Green i Blue. Wartości te są zazwyczaj liczbami całkowitymi lub liczbami zmiennoprzecinkowymi z zakresu 0-255, gdzie 0 reprezentuje brak danego koloru, a 255 oznacza pełne nasycenie danego koloru. Przykładowa tablica Numpy dla obrazu o rozmiarach 2x2 piksele wyglądałaby tak:

```
import numpy as np

image = np.array([
    [[255, 0, 0], [0, 255, 0]],
    [[0, 0, 255], [255, 255, 255]]
])
```