Klasyfikacja gatunku muzycznego na podstawie właściwości utworu

Dominika Bocheńczyk, Dorota Meszka, Jolanta Śliwa

8 kwietnia 2022

1 Cel obliczeń

Utworzenie prostej sieci neuronowej rozpoznającej gatunek muzyczny podanego utworu na podstawie jego cech.

2 Dane

Dane użyte w projekcie pochodzą ze zbioru GTZAN biblioteki Tensorflow. Zbiór ten składa się z 1000 ścieżek audio o długości 30 sekund. Każde 100 ścieżek należy do jednego z 10 gatunków muzycznych:

- blues
- muzyka klasyczna
- country
- disco
- hip hop
- jazz
- metal
- pop
- reggae
- rock

Każda ścieżka jest 16-bitowym plikiem audio 22050Hz Mono w formacie .wav. Dla każdego utworu wygenerowano jego reprezentację MFCC.

MFCC (Mel-frequency cepstral coefficients) reprezentują aspekty takie jak barwa, brzmienie oraz kompozycja w taki sposób, aby mogły później zostać odczytane przez komputer jako wektor wartości liczbowych. Reprezentacja przybliża sposób w jaki ludzki układ słuchowy odbiera dźwięki.

Aby nieco zwiększyć zbiór danych, zdecydowałyśmy się na podział każdej ścieżki muzycznej na krótsze fragmenty - początkowo 6-sekundowe, po modyfikacjach zdecydowałyśmy się na 3-sekundowe. W ostatecznym modelu użyłyśmy zatem 10 000 ścieżek muzycznych.

3 Oprogramowanie

Do wykonania zadania został użyty język Python z bibliotekami Tensorflow (w tym Keras), Scikit-learn, NumPy oraz pakiet Librosa.

4 Topologia sieci

Jednokierunkowa sieć neuronowa składa się z warstwy wejściowej, wyjściowej zawierającej 10 neuronów - tylu ile gatunków muzycznych rozróżniamy, oraz trzech warstw ukrytych, złożonych z odpowiednio 512, 256 i 64 neuronów. Dla każdej z warstw ukrytych zostosowana została funkcja aktywacji ReLu (Rectified Linear Activation Function), a dla wyściowej funkcja Softmax.

Dzięki zastosowaniu ReLu zmniejszamy prawdopodobieństwo pojawienia się problemu znikającego gradientu, a ponieważ Softmax zmienia uzyskany wektor liczb na wektor prawdopodobieństw (normalizuje wynik, w taki sposób, że całkowita suma kolejnych wartości wektora wynosi 1) jest odpowiednia do zadania klasyfikacji - każda pozycja obrazuje prawdopodobieństwo przynależności utworu do danego gatunku muzycznego.

Po otrzymaniu wyników pierwotnych, które wskazywały na przeuczenie sieci, dodałyśmy również dodatkowe warstwy - dwie warstwy konwolucyjne, oraz dwie warstwy łączące.

Łączna liczba optymalizacji, jaką musiała wykonać sieć wynosiła 1 211 594.

5 Symulacja

5.1 Parametry

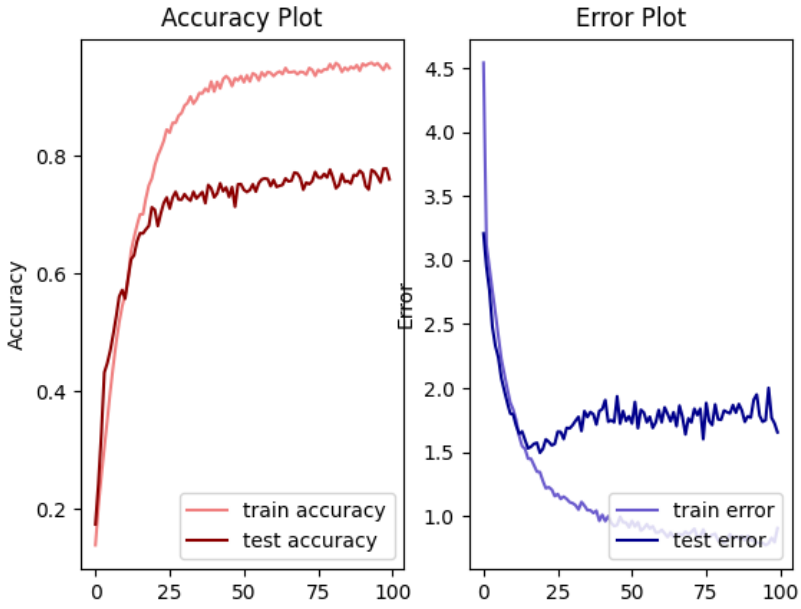
Zbiór danych został podzielony na parametry i oczekiwane wyniki dla zbioru uczącego i zbioru testowego. Rozmiar zbioru testowego został ustawiony na 30% a rozmiar zbioru treningowego stanowi pozostałe 70% danych. Trenowanie sieci neuronowej trwało 100 epok. Podczas trenowania sieci wykorzystano optymalizator Adam, który charakteryzuje się tym, że nie przeprowadza optymalizacji funkcji dla wszystkich danych treningowych tylko dla kolejnych partii (batch) danych, oraz funkcji straty categorical crossentropy.

5.2 Wyniki

Przy pierwszej próbie pomimo bardzo wysokiego poziomu trafności dla zestawu testowego - 97% prawidłowość w rozpoznawaniu zestawu testującego jest

znacznie mniejsza - 54%.

Po dokonaniu modyfikacji, wyniki zbioru treningowego nie uległy znacznemu obniżeniu - nadal utrzymują się w okolicach 95%, natomiast udało się zwiększyć poziom trafności zbioru testowego do około 75%.



Rysunek 1: Wykresy uczenia i straty dla 100 epok

6 Wnioski

Wynik otrzymany w pierwszej próbie jest spowodowany zjawiskiem przeuczenia sieci. Sieć zbytnio dopasowała się do zestawu treningowego w przeciwieństwie do testowego. Podczas drugiej próby dokonano modyfikacji, aby przeciwdziałać powyższemu zjawisku.

Pierwszą z nich było zastosowanie metody "dropout" polegającej na odrzuceniu losowo pewnej części neuronów w trakcie treningu. Po drugie, dodano regulator - element interfejsu API biblioteki Keras, służącego do regulacji wag, który umożliwia dodanie "kary" za wielkość wagi do funkcji straty, w naszym przypadku została użyta kategoria l2 - suma kwadratów wag. Postanowiłyśmy też zwiększyć zbiór danych z początkowego, wynoszącego 5000 ścieżek, do dwukrotnie większego. Ostatnią modyfikacją było dodanie dwóch warstw konwolucyjnych i łączących. Wszystkie te zmiany poskutkowały dość istotnym zwiększeniem skuteczności dla zbioru testowego.

Model okazał się być wysoce skuteczny, ale i dość szybki - analiza 100 epok trwała około 6 minut.