**Sztuczna Inteligencja - Projekt**

**Spoken langugage identification**

**Jakub Lewkowicz, Bartłomiej Kocot**

**03-06-2019**

**Wstęp**

Celem naszego projektu było stworzenia aplikacji klasyfikującej język mówiony na podstawie nagrania przez mikrofon lub przesłania pliku z nagraniem. W naszym projekcie uwzględniliśmy trzy języki tj. angielski, niemiecki i hiszpański i stworzyliśmy model, który określa jedną z wymienionych klas dla danego nagrania. Nasze rozwiązanie opiera się na przetworzeniu 10 sekundowych nagrań dźwiękowych w formacie .flac na spektrogramy, które następnie przetwarzane są przez konwolucyjną sieć neuronową, która zwraca prawdopodobieństwo przynależności dla danej klasy. W celu implementacji sieci posłużyliśmy się biblioteką Keras. Data set, którego użyliśmy pobrany został z platformy kaggle <https://www.kaggle.com/toponowicz/spoken-language-identification>.

**Podział projektu na zadania:**

Znalezienie odpowiedniego datasetu przetworzenie plików dźwiękowych wybór odpowiedniego modelu wytrenowanie modelu testowanie modelu ewentualne porównanie rezultatów uzyskanych w innych modelach stworzenie aplikacji z wykorzystaniem już wytrenowanego modelu .

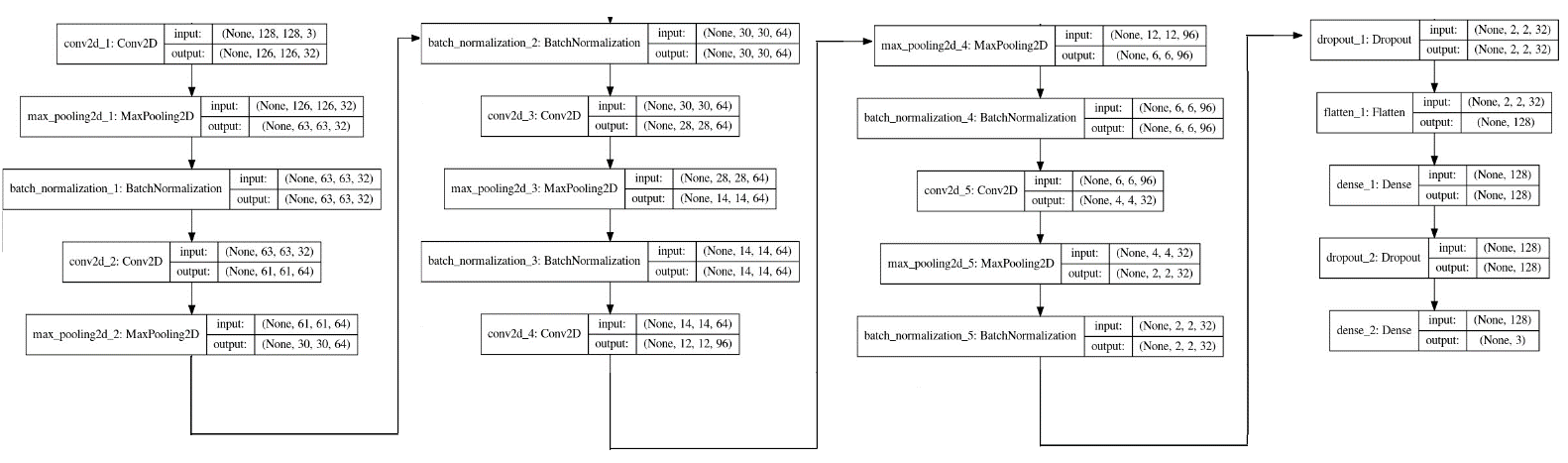
**Algorytm KNN**

Pierwszym podejściem do projektu było użycie algorytmu knn z wcześniej wygenerowanych cech dźwięku algorytmem MFCC. Stworzyliśmy własny algorytm knn, a następnie ze względu na dużą ilość cech na plik dźwiękowy (około 2000) musieliśmy odpowiednio je zredukować. Niestety algorytm dawał efekty do 48%. Można wyciągnąć wnioski, że algorytm KNN nie nadaję się do dużej ilości danych, a ich redukcja może powodować utratę ważnych informacji.

**Data set**

Zbiór danych składa się z 73080 plików treningowych i 540 plików walidacyjnych. Nagrania są w formacie .flac i każde jest długości 10 sekund. W zbiorze próbki są równo rozłożone jeśli chodzi o język, płeć, zakłócenia dźwięku jak i akcent mówcy.

**Model**

W naszym projekcie postanowiliśmy użyć konwolucyjną sieć neuronową o poniższej architekturze.

Rozwiązanie to powstało na podstawie innych przykładowych rozwiązań dla podobnych problemów znalezionych w internecie. Początkowym pomysłem była klasyfikacja języka na podstawie algorytmu KNN, aczkolwiek po czasie doszliśmy do wniosku, że nie ma to najmniejszego zastosowania dla tak dużych danych(73080 plików). Implementacja algorytmy KNN znajduje się w pliku knnAlgorithm.py. Użycie CNN okazało się zdecydowanie bardziej skuteczne, lecz też nie jest rozwiązaniem optymalnym. Kolejnym rozwiązaniem wartym uwagi jest zastosowanie algorytmu MFCC i użycie np. sieci typu LSTM.

**Trening**

Zanim zaczęliśmy trening naszej sieci musieliśmy przetworzyć pliki dźwiękowe na spektrogramy (algorytm w pliku spectrograms.py). Wygenerowaliśmy spektrogramy w jakości 50 dpi, a następnie przed podaniem plików na sieć zmieniliśmy ich rozdzielczość na 128x128 px.

**Parametry treningowe:**

optimizer: Adam,

training\_batch\_size: 128,

validation\_batch\_size: 8,

loss='categorical\_crossentropy',

steps\_per\_epoch = 200,

validation\_steps = 50 ,

epochs = 30

Dla takich parametrów uzyskaliśmy skuteczność **65%** dla danych walidacyjnych i 85% dla danych treningowych. Wcześniej używając podobnych parametrów, lecz generując spektrogramy o jakości 10 dpi (przetwarzając je na 40x40 px) uzyskaliśmy skuteczność około 50%, możemy więc założyć, że do pewnego stopnia dla lepszej jakości można uzyskać znacznie lepsze wyniki. Jednak z lepszą jakością wiąże się znacznie dłuższy czas przetwarzania zarówno generowania spektrogramów jak i treningu sieci.

**Interfejs graficzny**

W celu zwizualizowania naszej pracy utworzyliśmy graficzny interfejs aplikacji. Umożliwia ona zidentyfikowania języka na podstawie pliku dźwiękowego lub nagrania „na żywo”. W przypadku nagrania plik z dźwiękiem zapisany jest w folderze tymczasowym. W obu metodach tworzony jest następnie, odpowiedni spektogram i na podstawie jego oraz stworzonego modelu aplikacja zwraca odpowiedni język. Do stworzenia interfejsu graficznego wykorzystaliśmy Qt dla Pythona (biblioteka PySide2).