Analiza Obrazów

Dokumentacja projektu

Aplikacja do rozpoznawania amerykańskich znaków języka migowego

Marcin Filipek, Michał Ćwierz, Agnieszka Lasek

Link do kodu źródłowego: https://github.com/Fillipek/Sign-Language-Recongition.

Opis i założenia projektu

Celem naszego projektu było napisanie aplikacji, która na podstawie obrazu przekazywanego do niej z kamerki, jest w stanie wykryć litery z alfabetu amerykańskiego języka migowego. Założeniem wykorzystanym w projekcie było wykorzystanie sieci neuronowej w celu rozpoznawania odpowiednich kształtów. Dzięki tej informacji, dany obiekt można zakwalifikować jako jeden ze znanych znaków migowych.

Narzędzia wykorzystane do realizacji projektu

Program został napisany w całości w języku Python. Do realizacji najważniejszych elementów projektu posłużyła nam biblioteka Tensorflow, dzięki której możliwe było nauczenie sieci neuronowej rozpoznawania poszczególnych liter z języka migowego. Biblioteki Pillow i Opencv pomogły nam w obsługiwaniu elementów graficznych projektu oraz pozwoliły uzyskać zrozumiały graficzny interfejs użytkownika. Innymi bibliotekami niezbędnymi do działania programu są: pytesseract, numpy, matplotlib oraz pandas.

Do wytrenowania sieci użyliśmy zbioru danych “MNIST Sign Language” ze strony https://www.kaggle.com/datamunge/sign-language-mnist.

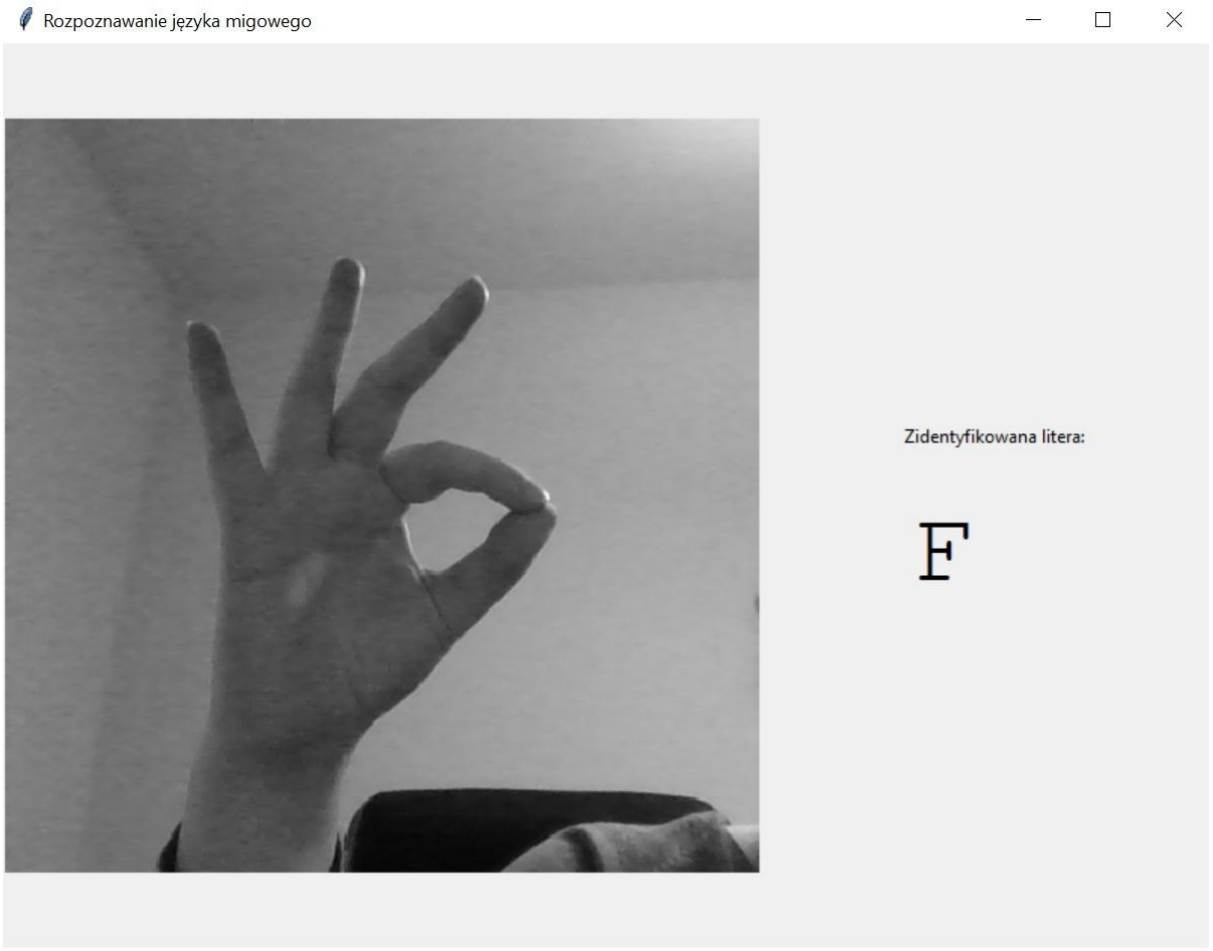
Podział pracy w zespole

- Marcin Filipek - algorytm
- Michał Ćwierz - interfejs użytkownika
- Agnieszka Lasek - dokumentacja

Należy zaznaczyć, że powyższe przypisanie zadań studentom to jedynie określenie osoby nadzorującej dany segment projektu. Sama realizacja zadań była przeprowadzana grupowo.

Interfejs użytkownika

Poniżej przedstawiony został wygląd interfejsu użytkownika, pojawiający się po poprawnym uruchomieniu programu. Należy pamiętać o tym, że program przetwarza obrazy przekazywane za pomocą kamerki, a więc musi być ona w pełni funkcjonalna po uruchomieniu projektu przez użytkownika.



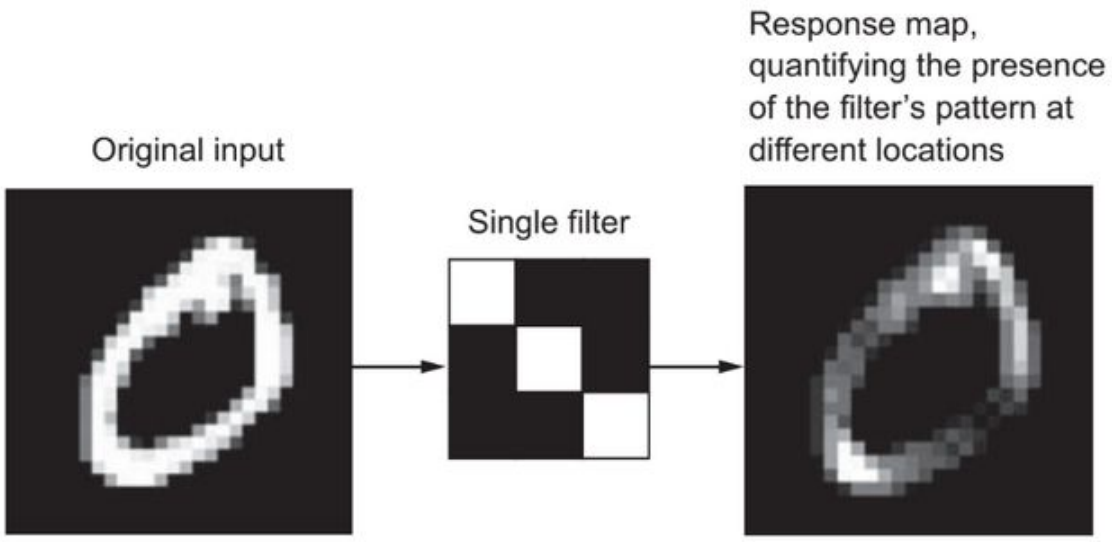
Obraz 1. Wygląd interfejsu użytkownika po poprawnym uruchomieniu programu.Instrukcja uruchomienia znajduje się w README.md w repozytorium.

Jak widać na załączonym powyżej obrazie, interfejs użytkownika składa się z dwóch głównych elementów. Po lewej stronie pokazywany jest przetwarzany aktualnie obraz, odczytywany za pomocą kamerki, natomiast po prawej stronie wyświetlana jest informacja o tym, jaka litera została zidentyfikowana przez program.

Rys. 2: Symbole rozopoznawane przez sieć i odpowiadająca im litera alfabetu.

Opis działania programu

Do rozpoznawania symboli najlepiej nadaje się konwolucyjna sieć neuronowa, dlatego taka została zimplementowana. Poniżej znajduje się tabela przedstawiająca budowę warstwową zaimplementowanej sieci. Zaletą tej sieci jest rozpoznawanie kształtów niezależnie od ich położenia, obrotu i skali na obrazie wejściowym.



Rys. 3: Schemat działania sieci konwolucyjnej. Każda wartwa konwolucyjn*a używa filtrów (kerneli) do wyodrębniania z obrazu coraz prostszych cech. Warstwy "Pooling" służą do wyciągania wartości min, max lub średnich z wyodrębionych cech.*

Numer i nazwa warstwy	**Ilość filtrów**	**Funkcja aktywacji**	**Dodatkowe parametry**
1 - Conv2D	32, kernel = (3x3)	relu	format danych wejściowych = (28 x 28)
2 - MaxPooling2D	(2 x 2)		
3 - Conv2D	64, kernel = (3x3)	relu	
4 - MaxPooling2D	(2 x 2)		
5 - Conv2D	64, kernel = (3x3)	relu	
6 - Flatten	Spłaszczanie struktury poprzeniej wartwy do 1-wymiarowego wektora		
7 - Dense(64)	Zagęszczanie poprzedniej warstwy, funkcja aktywacji = relu		
8 - Dense(25)	Warstwa wynikowa o rozmiarze takim, jak ilość rozpoznawanych symboli (ilość klas).		

Użyta funkcja optymalizacji (optimizer) nosi nazwę "adam", a funkcja straty mierząca postęp uczenia się to funkcja z biblioteki Keras o nazwie SparseCategoricalCrossentrophy.

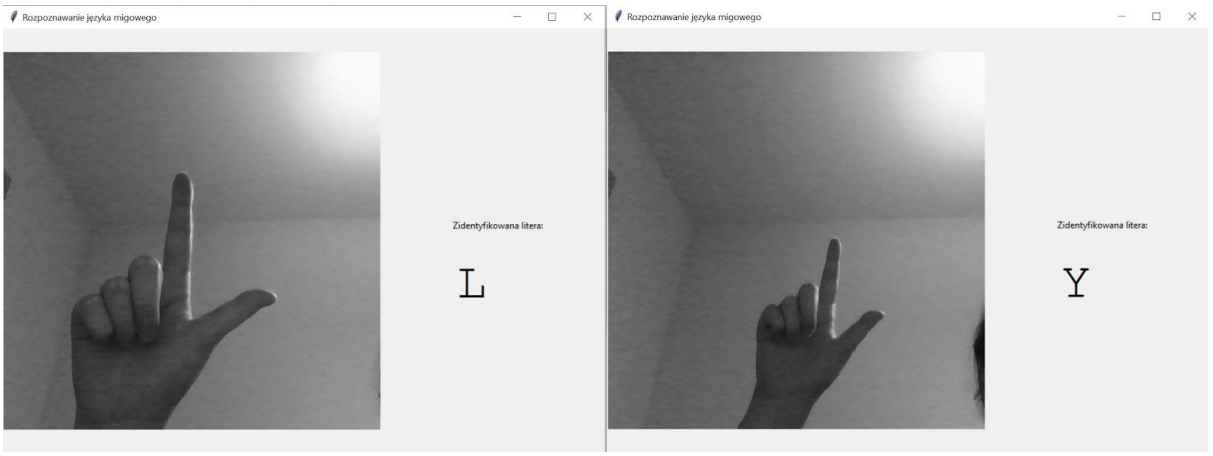
Ilość danych treningowych w zbiorze to nieco ponad 25 tysięcy obrazów, co nie jest bardzo dużym zbiorem danych. Z tego powodu program używa generatora danych do zwiększenia liczby obrazów treningowych. Generator przekształca obrazy używając symetrii, obrotu, przesunięcia i skalowania o losowe wartości do wygenerowania większej ilości danych dla sieci.

Potencjalne błędy

Program nie rozpoznaje poprawnie wszystkich symboli. Jest to spowodowane kilkoma czynnikami:

- Dane wejściowe sieci to obraz o rozmiarach 28 x 28 pikseli. Jeżeli dłoń jest oddalona od kamery za bardzo to sieć widzi niekształtny zbiór pikseli, z którego nie jest w stanie rozpoznać niektórych kształtów. Dobrym przykładem są tu litery A, S, M i N, które przedstawia się za pomocą złożonej w pięść dłoni z różnym ułożeniem kciuka. Rozwiązaniem mogłoby być przerobienie sieci na większy wymiar, bądź wycinanie samej dłoni z obrazu kamery i powiększenie go.
- Sieć, pomimo użycia generatora otrzymała niewielką ilość danych. Zwiększenie zbioru treningowego mogłoby dać lepszą skuteczność.
- Tło w obrazie kamery rozprasza sieć, próbując zinterpretować wyróżniające się obiekty jako symbole.
- Sieć nie rozpoznaje liter J oraz Z. Litery te przedstawia się za pomocą dynamicznego ruchu, podczas gdy nasz program rozpoznaje statyczne obrazy.

Rys. 4: Kilka losowych obrazów ze zbioru treningowego. Palce w niektórych obrazach są zamazane i trudno je rozróżnić.



Obraz 2. Efekty działania programu dla przedstawienia litery “L” z różnych odległości od kamery.

Jak widać na powyższym obrazie, na poprawność rozpoznawania litery alfabetu języka migowego może wpływać wiele czynników związanych z przekazywanym obrazem. Zaprezentowana powyżej litera “L” może być rozpoznana poprawnie w określonych okolicznościach, jednak często będzie też rozpoznawana błędnie.

Błędy w rozpoznawaniu liter mogą być również w dużej mierze spowodowane podobieństwem pomiędzy poszczególnymi znakami migowymi i brakiem możliwości rozpoznania mało widocznych różnic między nimi.

Potencjał rozwoju

Aby poprawić efekty działania programu i częściej otrzymywać poprawne wyniki można w przyszłości zastosować ulepszenia takie jak:

- użycie pretrenowanej sieci neuronowej, np. MobileNet V2 z kilkoma własnymi warstwami, aby dostosować do własnej potrzeby (uzycie tej sieci wymaga obrazów treningowych o rozmiarze minimum 32x32)
- zastosowanie zbioru obrazów kolorowych o większej rozdzielczości
- algorytm wykrywający rękę na obrazie wysokiej rozdzielczości z kamery i wycinający tylko ten fragment obrazu, w którym się znajduje
- usuwanie tła
- dysponując odpowiednim sprzętem możliwe byłoby wprowadzenie na wejściu większej ilości danych, wykorzystywanych do uczenia sieci neuronowej

W celu poprawienia działania programu możliwe jest wprowadzenie wielu rodzajów zmian i ulepszeń, jednak ciężko jest jednoznacznie określić, jaki sposób rozwiązania analizowanego problemu byłby najbardziej optymalny.