Dokumentacja konwolucyjnej sieci neuronowej do klasyfikacji obrazów języka migowego.

1. **Dane**

   Zbiór danych zawiera 1200 zdjęć przedstawiających litery w języku migowym podzielone równomiernie na 4 kategorie/ foldery. Folder A zawiera zdjęcia dłoni przedstawiającą literę A. Analogiczna sytuacja występuje również w folderach B i C. W folderze 0 zostały zawarte losowo przemieszane inne znaki języka migowego.

2. **Przygotowanie danych**

   Ścieżka do danych została umieszczona w pliku config.ini. Dzięki temu zmiana ścieżki nie wymaga ingerencji w kod. Ze ścieżki skrypt pobiera nazwy folderów jako nazwy klas do tablicy cls_lables oraz sprawdza ile plików jest w poszczególnych folderach.

   Następnie za pomocą trzech zmiennych train_ratio, valid_ratio oraz test_ratio jest określana proporcja w jakiej dzielimy dane na treningowe, walidacyjne i testowe. Kolejnym krokiem jest stworzenie folderu imgs w katalogu projektu na przetwarzane obrazy. Po stworzeniu uruchamiana jest metoda split_the_data. Ma ona za zadanie skopiować do folder z danymi do folderu imgs, oraz rozdzielić je w odpowiednich proporcjach na dane treningowe, walidacyjne i testowe. Przykładowa strutkura danych w folderze imgs to imgs\test\0.

   Za pomocą instancji klasy ImageDataGenerator dane treningowe zostają poddane metodom augmentacji i normalizacji. Normalizacja powoduje wyskalowanie wartości w macierzy reprezentującej zdjęcie do wartości z zakresu 0-1. Użyte metody augmentacji to rotation_range, width_shift_range, height_shift_range, horizontal_flip_true. Podczas przesunięcia obrazu brakujące piszczele są wypełniane przez najbliższe za pomocą parametru fill_mode = ‘nearest’ Dla danych walidacyjnych zostaje zastosowana tylko normalizacja. Za pomocą metody flow.from.directory dane zostają przygotowane do przekazania do modelu. Rozmiar zdjęc zostaje ustawiony na 100x100 w celu przyspieszenia procesu uczenia, a parametr na ‘categorical’ poniewaz mamy wiecej niz dwie klasy. Metoda została zastosowania dla danych walidacyjnych i testowych.

3. **Budowa modelu**

   Stworzony model składa się z 3 warstw konwolucyjnych Conv2D, 4 warstw MaxPooling2d, warstwy Flatten, 3 warstw Dense oraz warstwy Droput. Warstwy konwolucyjne zawierają kolejna 64,128,254 filtry, funkjci aktywacyjnych ‘relu’. Pierwsza z nich zawiera parametr input_shame określający rozmiar macierzy przesyłanych do modelu. Warstwy MaxPooling posiadają wymiary pola odpowiednio (4,4), (2,2), (2,2) i (2,2). Po warstwie wypłaszczającej Flatten model posiada 3 warstwy gęsto połączone Dense oraz jedną warstwę Dropout. ilość neuronów w

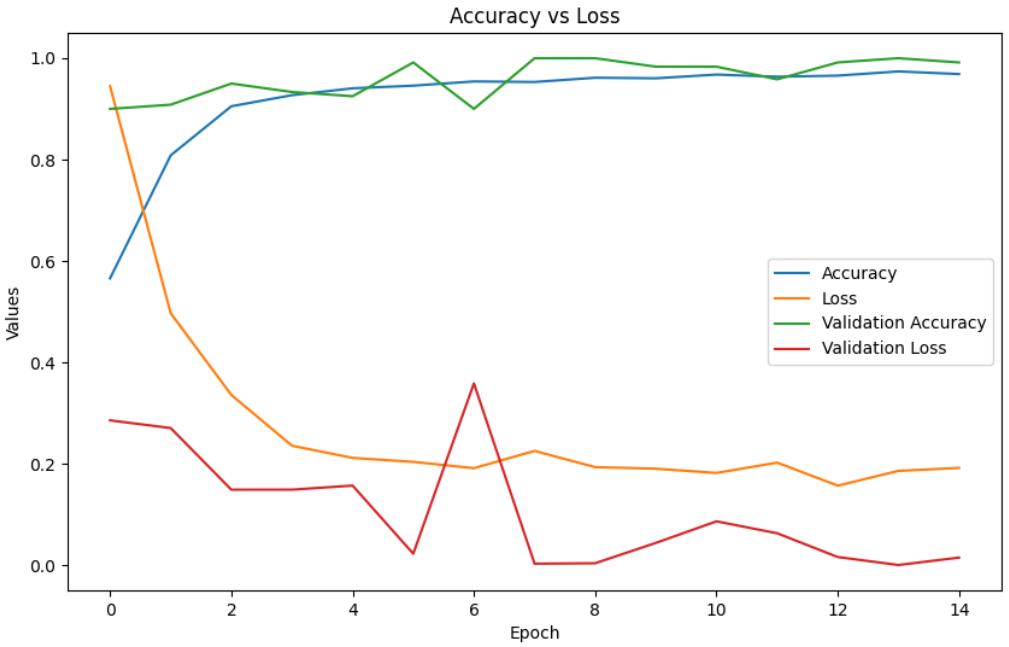
warstwach Dense to kolejno 256, 32 i 4. Funkcje aktywacji w dwóch pierwszych warstwach to relu, natomiast w ostatniej softmax. Warstwa droput wyłącza 30% neuronów w pierwszej warstwie Dense. Model został skompilowany przy użyciu optymalizera RMSprop ze współczynnikiem uczenia na 0.001, funkcja straty to 'categorical_crossentropy' a metryka 'accuracy'.

4. Trenowanie modelu

Podczas trenowania modelu i regulacji jego za pomocą zmiany:

- zmiany wartości dropout
- zmianie ilości epok
- zmianie ilości batch_size
- zmianie ilości warstw gęsto połączonych

Udało się uzyskać takie parametry:



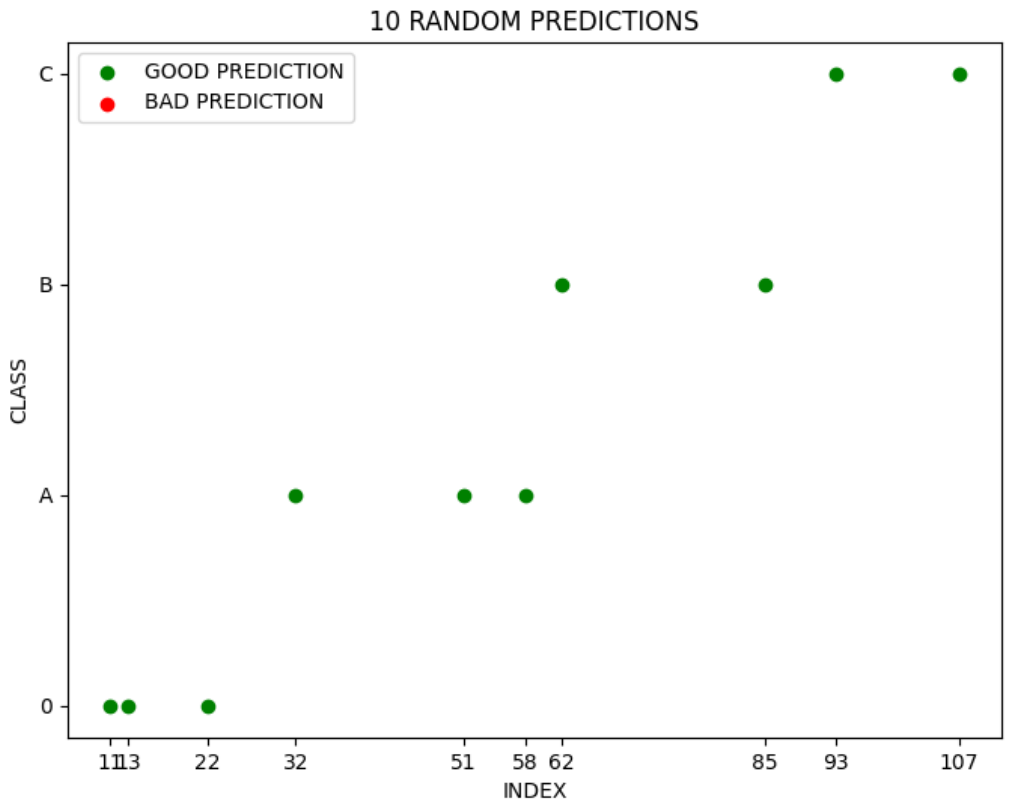
Dokładność i starty na zbiorze walidacyjnym uważam na zadowalającą. Ilość epok wynosi 15.

Model zostaje zapisany do pliku model.h5 znajdującym się w katalogu projektu.

5. Testowanie modelu

Został stworzona instancja klasy ImageDataGenerator w którym dane zostały znormalizowane tak samo jak na poprzednich zbiorach. Batch_size w metodzie flow_from_directory zostaje ustawiony tym razem na 1, jak jest to zalecane podczas testowania sieci. Wyniki predykcji zostają zapisane do numpy array, następnie zostały

wyciągnięte najwyższe wartości z każdego obiektu funkcja argmax i zapisane w Pandas DataFame na podstawie którego został stworzony wykres obrazujący trafność predykcji na podstawie 10 losowych indeksów.



Na wykresie widać, że model uzyskał wynik 10 na 10.

Raport klasyfikacyjny prezentuje się w ten sposób:

	precision	recall	f1-score	support
0	1.00	0.97	0.98	30
A	1.00	1.00	1.00	30
B	0.97	1.00	0.98	30
C	1.00	1.00	1.00	30
accuracy			0.99	120
macro avg	0.99	0.99	0.99	120
weighted avg	0.99	0.99	0.99	120

Z raportu wynika, że sieć poradziła sobie bardzo dobrze z danymi testowymi.