|  |  |
| --- | --- |
| Projekt – sieci neuronowe | Data złożenia projektu: 13.05.2022 |
| Numer grupy projektowej: CWP 4 | Imię i nazwisko I: Szymon Musiał  Imię i nazwisko II: Piotr Wilkosz |

TSR – System rozpoznawania znaków (polskich)

# Opis problemu i danych

W ramach projektu poruszana jest dziedzina klasyfikacji obrazów. Klasyfikowanymi obrazami są polskie znaki drogowe. Celem jest nauczenie sieci poprawnej klasyfikacji zdjęcia znaku do symbolu znaku.

Zdjęcia z kamery przedniej samochodu, skalsyfikowane przez model mogą być pomocą dla kierowcy.

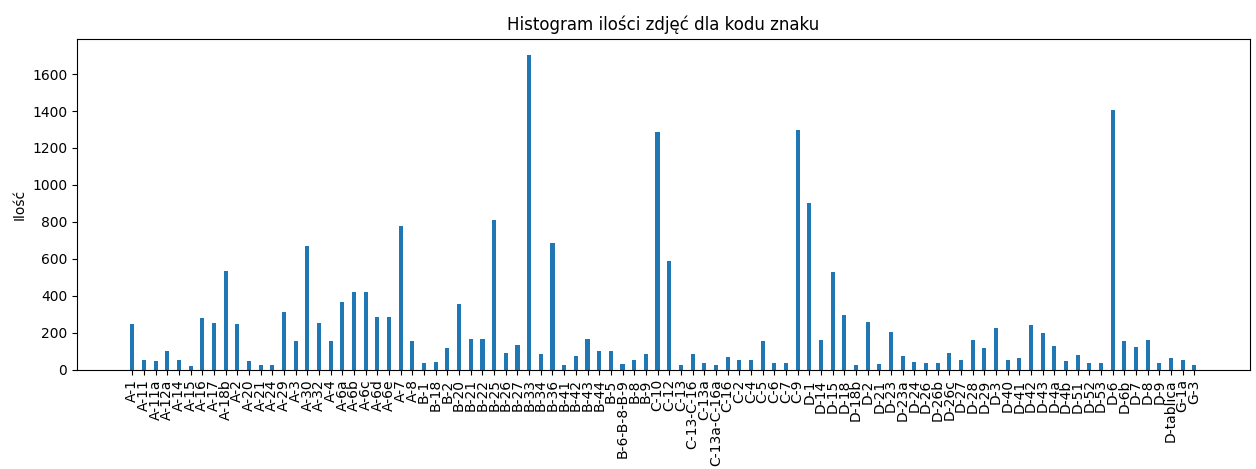
Analizowany zbiór danych składa się z 21\_032 rekordów. Znajdziemy w nim 92 zmienne wejściowe. Liczność wybranych zmiennych zaprezentowana jest na poniższym histogramie. Problematyczne staje się przedstawienie podstawowych statystyk ze względu, iż rozpatrywany jest problem klasyfikacji obrazów.

Pod względem liczności rekordów w klasie możemy zauważyć, że maksymalna ilość obrazów w klasie wynosi 1706 dla klasy 38 o symbolu B-33 – Ograniczenie prędkości.

Przechodząc do analizy klasy o minimalnej ilości rekordów możemy wyróżnić klasę 15 o symbolu A-15 – Śliska jezdnia o liczności równej 20.

Średnia ilość rekordów na klasę wynosi 228.61 rekordy.

Odchylenie standardowe wynosi 319.02.



Wykres 1. Ilość zdjęć dla danego kodu znaku

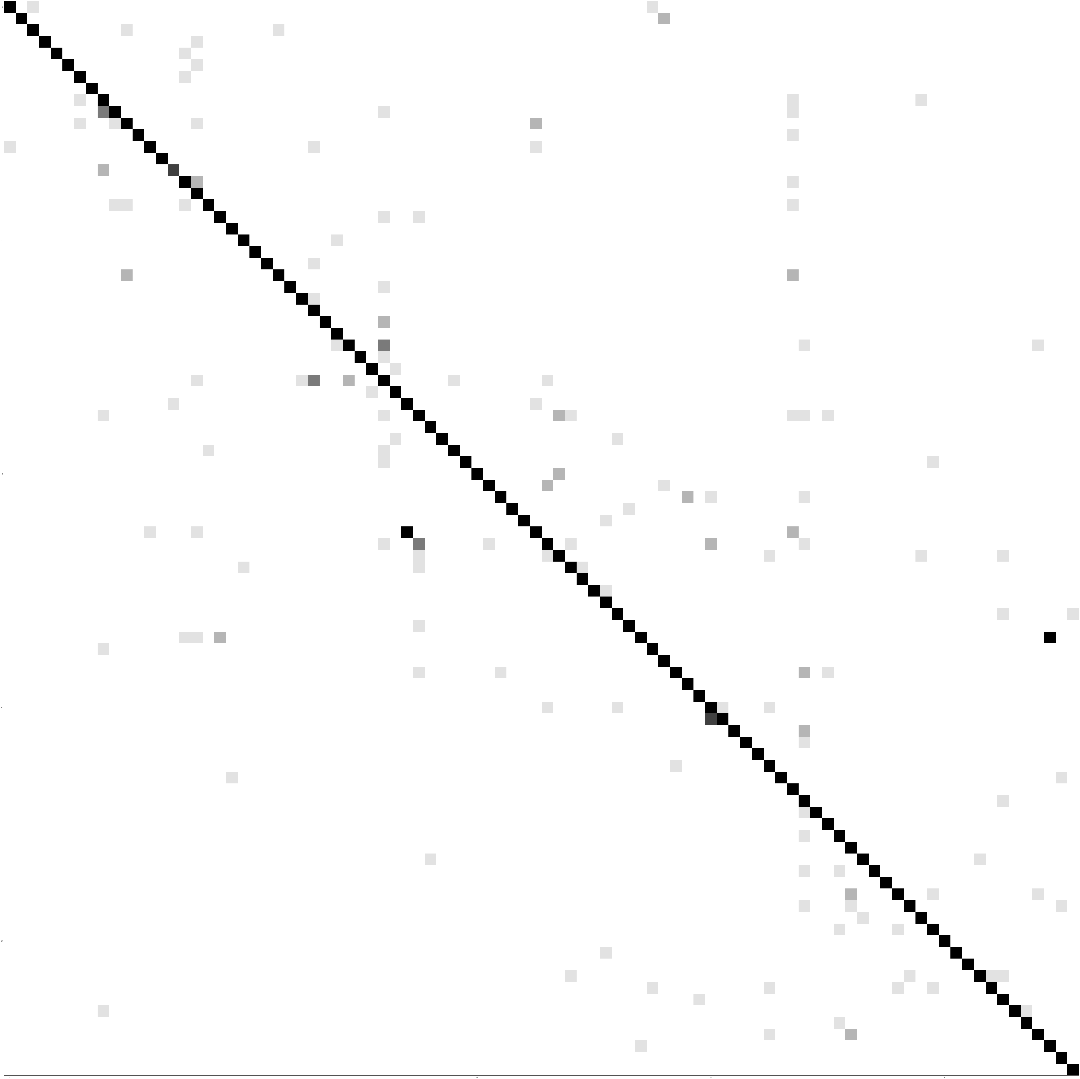
# Obróbka danych

Ilość brakujących rekordów i sposób rozwiązania tego problemu, normalizacja/standaryzacja danych wejściowych, transformaty wykonane na danych (zależnie od problemu, np. dla szeregów czasowych średnia z ostatnich 10 rekordów, dla rozpoznawania obrazów przetworzenie przez filtry graficzne). Sposób podziału na zbiór uczący oraz zbiór testujący.

Otrzymany zbiór posiadał już podział na testujący oraz treningowy. Ilość rekordów w poszczególnych zbiorach rozkładał się w proporcji 80:20. Ze zbioru treningowego 10% zdjęć zostało przeniesione do zbioru walidującego używanego przy trenowaniu.

Ponadto zbiór testujący po wytrenowaniu sieci został użyty w celi walidacji dokładności.

Poniżej zaprezentowano macierz po odfiltrowaniu zdjęć złej jakości, na której to osi X znajduje się klasa przypisana do zdjęcia testującego a do osi Y klasyfikacja modelu. Odchyłki od głównej przekątnej stanowią błędy dopasowania. Kolor elementów macierzy jest szary dla 5 elementów



Rys 1. Graficzna macierz dopasowania modelu

Macierz ta była decyzyjną w procesie filtrowania, a zaprezentowana powyżej jest już wynikową.

Ponadto zastosowany ImageDataGenerator od razu dokonuje powiększenia zbioru stosując np. przesunięcie, skalowanie, rotacje i tym podobne przekształcenia.

W celu modyfikacji zdjęć podjęto próby wykorzystać to VGG16 oraz VGG19, przekazany jako argument ImageDataGenerator reprocessing\_function=tf.keras.applications.vgg16.preprocess\_inpu

Jednakże spowodowało to spadek dokładności modelu, dlatego zrezygnowano z niego. Gdyby użyto przetrenowanego wcześniej modelu skorzystanie z VGG lub podobnych modelów było by wręcz koniecznością ze względu na dostępność danych w tych modelach

Obraz zawierający tekst, wewnątrz, sprzęt elektroniczny, wyświetlanie

Opis wygenerowany automatycznie

Rys 2. Zastosowany filtr VGG16

Dodatkowo sprawdzono filtr szum Gaussa który również nie przyczynił się do poprawy wyników.

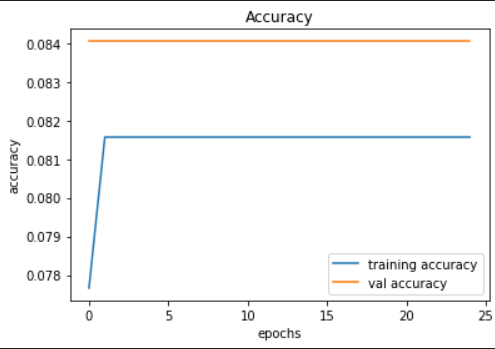


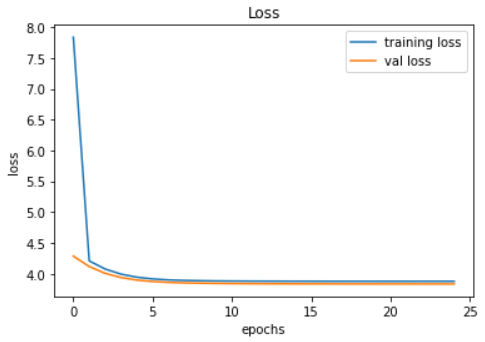
Rys 3. Dane – obrazy wejściowe

# Opis zastosowanych sieci neuronowych

Typ i architektura sieci, ilość epok uczenia, sposób uczenia sieci. Informacja o własnej implementacji/ użytym frameworku. Co najmniej 2-3 różne architektury sieci.

Pojedyncze Conv2D





W projekcie zastosowano konwolucyjną sieć neuronową

Ilość epok uczenia została empirycznie dobrana na 25, mniejsza liczba powodowała niską dokładność a większa powodowała przetrenowanie co skutkowało rozbieżnością na poziomie kilkudziesięciu procent w testach dokładności na danych testujących i walidujących podczas uczenia modelu

Do uczenia użyto biblioteki Keras który jako backend wykorzystuje bibliotekę Tensorflow w wersji 2.5, który natomiast wykorzystuje kartę graficzną w celu przyspieszenia uczenia. Oprócz tego w celu wizualizacji uzyskanych wyników używamy biblioteki matplotlib. Do konsrukcji macierzy [confusion(konfuzji)?] używamy możliwości oferowanych przez bibliotekę sklearn. Do operacji matematycznych wykorzystujemy bibliotekę numpy.

#TODO 2 dodatkowe architektury sieci

# Dyskusja wyników oraz wnioski

Zebrane w tabeli wyniki sieci z podziałem na zbiór uczący i testujący. Co najmniej 2 miary, wraz z komentarzem, dlaczego zostały wybrane takie, a nie inne. Określenie (subiektywne), czy wyniki są satysfakcjonujące wraz z uzasadnieniem. Wnioski, dalsze propozycje rozwoju projektu.

#TODO wyniki w tabeli, skad te wyniki?

#TODO 2 miary

Uważamy, że uzyskane wyniki są satysfakcjonujące. Specyfika rozpatrywanego przez nas problemu jest problemem skomplikowanym. Jakość używanych zdjęć różnie nie jest rewelacyjna. Licznośc poszczególnych klas, ilość znaków drogowych w kategoriach bardzo się różni. Czasem zdjęć jednego znaku jest 20 a czasem 1200. Wynika to z czystej specyfiki rozpatrywanego problemu. Naturalnym zjawiskiem pod względem kraju w którym się znajdujemy jest, że na drodze częściej znajdziemy znaki ograniczające maksymalną dozwoloną prędkość niż znaki na przykład ostrzegające o przeprawie promowej lub o zakazie wjazdów motorowerów. Z tego też powodu napotykamy się problemem wysokiego odchylenia standardowego wynoszącego 319. W związku z tym uważamy, że uzyskany przez nas współczynnik dokładności $accuracy jest satysfakcjonujący biorąc pod uwagę specyfikę problemu.

Dalszym etapem rozwoju projekty mogłoby być zastosowanie mechanizmu detekcji znaków drogowych w czasie rzeczywistym dzięki czemu możliwy byłoby zbudowanie systemu informującego kierowcę o drodze.