Projekt ze “Sztucznej inteligencji”

Sprawozdanie

Klasyfikacja znaków drogowych

Mateusz Stencel	Tomasz Szczęchor	Oskar Ruczyński
188676	188721	188862

1. Wstęp

1.1. Klasyfikacja

Klasyfikacja jest klasycznym przykładem uczenia nadzorowanego. Polega ona na przypisywaniu obiektów do określonych klas na podstawie ich cech lub atrybutów. Celem klasyfikacji jest znalezienie modelu lub funkcji, która może przewidywać przynależność obiektów do określonych klas. Przydzielanie do klasy odbywa się na podstawie dostarczonych wcześniej danych treningowych, których przynależność do danej kategorii jest znana.

1.2. Założenia

Celem projektu było porównanie skuteczności działania wybranych klasyfikatorów na przykładzie klasyfikacji znaków drogowych dla różnego stopnia skomplikowania problemu klasyfikacji rozumianego jako liczba klas, na które rozróżnia się znaki drogowe.

2. Podstawy teoretyczne porównywanych metod

Porównano działanie następujących klasyfikatorów: k–najbliższych sąsiadów, liniowego SVM oraz konwolucyjnej sieci neuronowej.

2.1. Klasyfikacja k–najbliższych sąsiadów

Klasyfikator k-najbliższych sąsiadów (kNN - ang. *k nearest neighbours*) jest jednym z najprostszych modeli klasyfikacyjnych. Polega on na wyznaczeniu odległości nowego obiektu od innych zakwalifikowanych już obiektów, znalezieniu jego k najbliższych sąsiadów, a następnie przydzieleniu go do klasy, do której należy najwięcej sąsiadów. Działanie modelu opiera się na odległościach między obiektami, stąd ważnym aspektem jest dobranie odpowiedniego sposobu obliczania tej odległości. Najczęściej stosowaną metryką jest odległość euklidesowa. Równie ważnym aspektem algorytmu jest odpowiedni dobór hiperparametru k. Jednym z najprostszych i najczęściej stosowanych sposobów tego doboru jest przetestowanie działania modelu dla różnych wartości k, a następnie wybranie tej wartości, dla której algorytm osiąga najlepsze wyniki.

2.2. Klasyfikacja liniowa SVM

Maszyna wektorów wspierających (SVM – ang. *support vector machine*) jest liniowym modelem do przeprowadzania klasyfikacji. Działanie modelu polega na oddzielaniu obiektów, które są liniowo rozdzielne, za pomocą prostej. Proste te stanowią granice decyzyjne dla klasyfikatora. W przypadku wielowymiarowej klasyfikacji występuje hiperpłaszczyzna zamiast prostej. Marginesami nazywa się odległości między najbliższymi obiektami różnych klas. Obiekty te nazywane są wektorami wspierającymi. Celem SVM jest maksymalizacja marginesów między wektorami wspierającymi klas. Proces ten nazywa się klasyfikowaniem maksymalnego marginesu. Funkcja straty oblicza średnią wartość wszystkich naruszeń marginesów. W procesie uczenia ważną rolę pełnią dwa parametry, które są ściśle ze sobą związane. Tymi parametrami są: liczba iteracji oraz współczynnik nauki (*LR*), który określa przyrost nauki modelu.

Jednym ze sposobów minimalizowania funkcji straty jest metoda local search. Jest to metoda optymalizacji dyskretnej, polegająca na przeszukiwaniu zbioru sąsiadów bieżącego rozwiązania i wybraniu takiego, który rokuje najlepiej w danym momencie. Wyróżnia się

prostotą i elastycznością. Ze względu na to, że jest to algorytm zachłanny, jego działanie może się zakończyć w optimum lokalnym.

2.3. Klasyfikacja z użyciem konwolucyjnej sieci neuronowej

Sieć konwolucyjna (ang. CNN – *Convolutional Neural Network*) to rodzaj sieci neuronowej, która jest powszechnie stosowana w zadaniach analizy obrazu. Sieć CNN zbudowana jest z 3 głównych rodzajów warstw – konwolucyjnych, łączących i w pełni połączonych.

Warstwy konwolucyjne (ang. *convolutional layers*) wykonują operację splotu danych wejściowych z odpowiednio skonstruowanym filtrem. Proces ten ma na celu uczenie wykrywania cech i wzorców zawartych w obrazie. Pierwsze warstwy konwolucyjne wydobywają ogólne cechy, którymi mogą być np. krawędzie. Kolejne warstwy uczą się wykrywać szczegółowe informacje. Warstwy łączące (ang. *pooling layers*) zmniejszają rozmiar przestrzeni cech w celu zredukowania kosztów obliczeniowych, rozmiaru wykorzystywanej pamięci, a także zmniejszenia liczby parametrów oraz ograniczenia ryzyka przetrenowania modelu. Warstwy w pełni połączone (ang. *fully-connected layers*), zwane też warstwami gęstymi, przetwarzają wektor cech i uczą się globalnych wzorców z cech uzyskanych od warstw konwolucyjnych i łączących.

Wyróżnia się także warstwy spłaszczające (ang. *flatten layer*) oraz warstwy porzucenia (ang. *dropout layer*). Warstwa spłaszczająca przekształca wielowymiarowe dane wejściowe na jednowymiarowy wektor. Zazwyczaj jest używana jako przejście między warstwą konwolucyjną, a warstwą w pełni połączoną. Warstwa porzucenia losowo wyłącza niektóre neurony podczas treningu. Jest to jeden ze sposobów zapobiegania przeuczeniu się modelu, czyli na tyle dokładnemu dopasowaniu do danych treningowych, że ma to negatywny wpływ na dokładność modelu dla danych testowych. Na końcu sieci stosuje się warstwę wyjściową, która generuje wynik klasyfikacji.

3. Przebieg realizacji

Projekt wykonano przy użyciu języka **`Python`**. Do pomiarów wykorzystano odpowiednio przygotowaną bazę danych niemieckich znaków drogowych. Znaki drogowe podzielono na 3 klasy znaków: ostrzegawcze, nakazu i zakazu. Spośród tych 3 klas wyszczególniono 36 podklas. Wszystkie zdjęcia znaków zostały przeskalowane do rozmiaru 32 x 32 pikseli.

Rys. 3.1. – rodzaje znaków znajdujące się w zbiorach treningowym i testowym

3.1. Klasyfikacja k–najbliższych sąsiadów

Zaimplementowano model, którego działanie opiera się na mierzeniu odległości od wszystkich sąsiadów i wybieraniu najbliższych. Mierzenie oparto na porównywaniu zdjęć poprzez obliczanie różnic w wartościach składowych RGB poszczególnych pikseli, a następnie obliczaniu normy z otrzymanych różnic. Zbadano wpływ użytych norm oraz liczby k najbliższych sąsiadów na dokładność klasyfikacji. Wybrano parametry, dla których model osiąga najlepsze rezultaty. Do prezentacji wyników użyto miary euklidesowej (L2) oraz przyjęto $k = 1$, co czyni model klasyfikatorem według najbliższego sąsiada.

3.2. Klasyfikacja liniowa SVM

Zaimplementowano liniowy model SVM, którego działanie opiera się na macierzy wag. Dla każdej klasy znaków macierz zawiera wagi, przypisane poszczególnym pikselom, które mają wpływ na wynik przydziału do danej klasy. W modelu zastosowano podejście jeden przeciw wszystkim (ang. *one vs. all*), polegające na oznaczeniu jednej klasy jako pozytywnej, a reszty klas jako negatywne. Do minimalizacji funkcji straty użyto metody local search. Do wyznaczenia wag dla klasyfikacji 3 klas znaków model trenowano przez 10 tysięcy iteracji przy współczynniku uczenia LR = 10^{-5}. Natomiast dla klasyfikacji 36 klas model trenowano przez 100 tysięcy iteracji przy współczynniku uczenia LR = 10^{-6}.

3.3. Klasyfikacja z użyciem konwolucyjnej sieci neuronowej

Implementując model skorzystano z funkcjonalności dostępnych w bibliotece `tensorflow`. Do uczenia wykorzystano optymalizator "Adam" oraz entropię krzyżową (ang. *categorical cross-entropy*) jako funkcję kosztu. Dla klasyfikacji obejmującej 3 klasy trenowano model przez 30 epok. Natomiast dla 36 klas przez 80 epok.

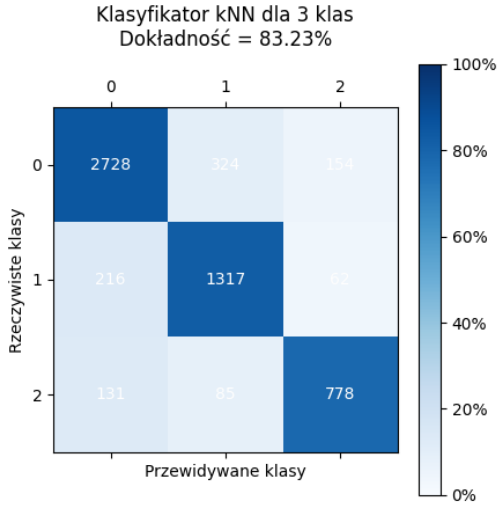
Warstwa	Typ	Mapy	Rozmiar	Rozmiar jądra	Krok	F. aktywacji
In	Wejściowa	3 (RGB)	32x32	—	—	—
Conv2D	Splotowa	32	30x30	3x3	1	ReLU
MaxPool	Maks. łącząca	32	15x15	2x2	2	—
Conv2D	Splotowa	64	13x13	3x3	1	ReLU
MaxPool	Maks. łącząca	64	6x6	2x2	2	—
Conv2D	Splotowa	64	4x4	3x3	1	ReLU
Flatten	Spłaczająca	1024	1x1024	—	—	—
Dense	W pełni połączona	64	1x64	—	—	ReLU
Dropout	Porzucenia	—	1x64	—	—	—
Out	W pełni połączona	36	1x36	—	—	—

Tabela 3.3.1. – architektura sieci konwolucyjnej dla 36 klas

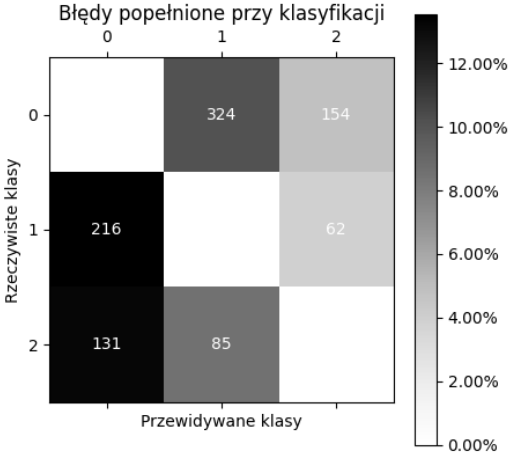
4. Wyniki

4.1. Klasyfikacja k–najbliższych sąsiadów

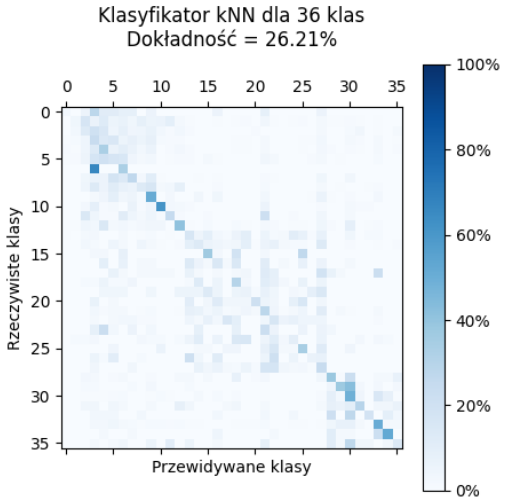
Poniżej przedstawiono wyniki klasyfikacji dla 3 i 36 klas za pomocą macierzy konfuzji i błędów. Wykreślono również zależność dokładności od liczby najbliższych sąsiadów *k* przy zastosowaniu normy miejskiej (L1) oraz normy euklidesowej (L2).



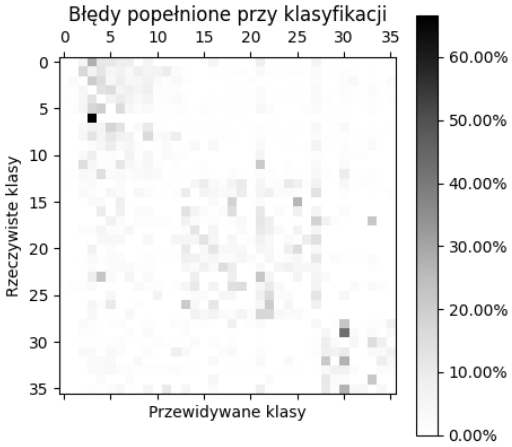
Rys. 4.1.1. – Macierz konfuzji dla modelu kNN, podział na 3 klasy, k=1, norma euklidesowa



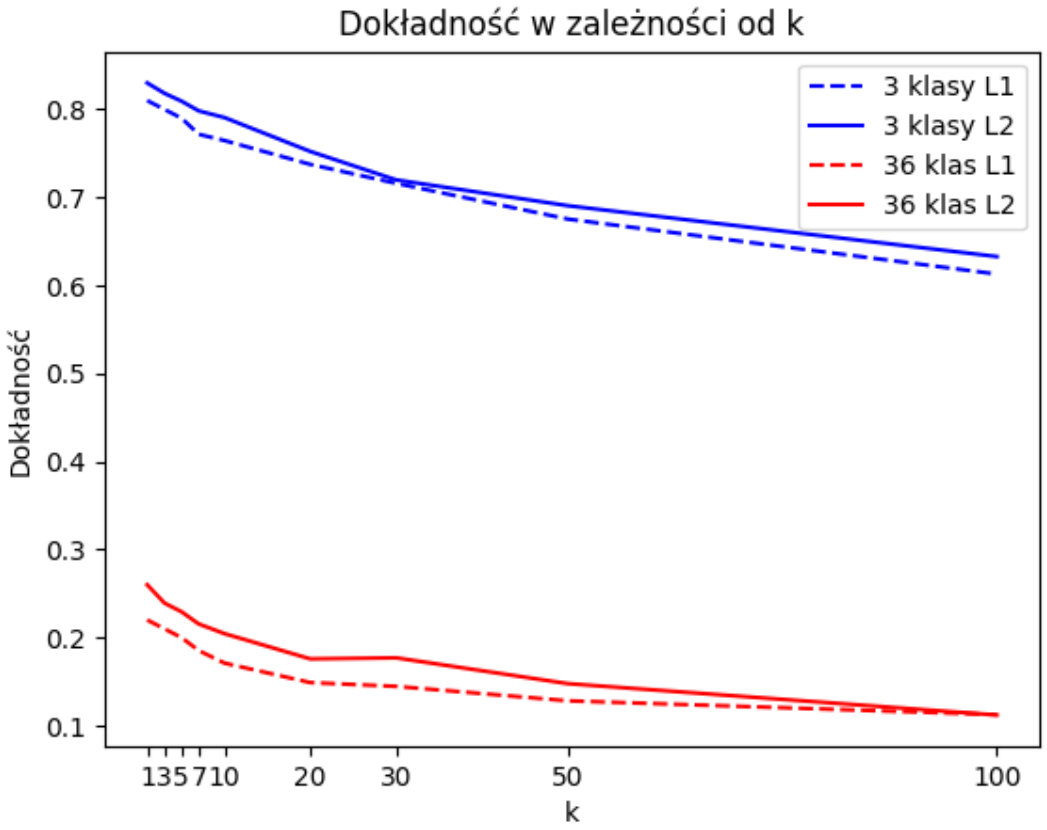
Rys. 4.1.2. – Macierz błędu dla modelu kNN, podział na 3 klasy, k=1, norma euklidesowa



Rys. 4.1.3. – Macierz konfuzji dla modelu kNN, podział na 36 klas, k=1, norma euklidesowa



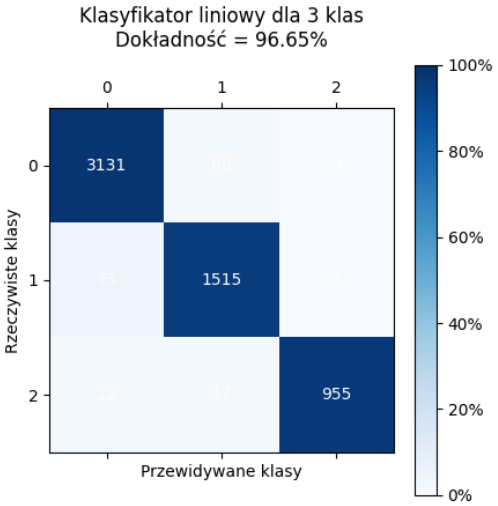
Rys. 4.1.4. – Macierz błędu dla modelu kNN, podział na 36 klas, k=1, norma euklidesowa



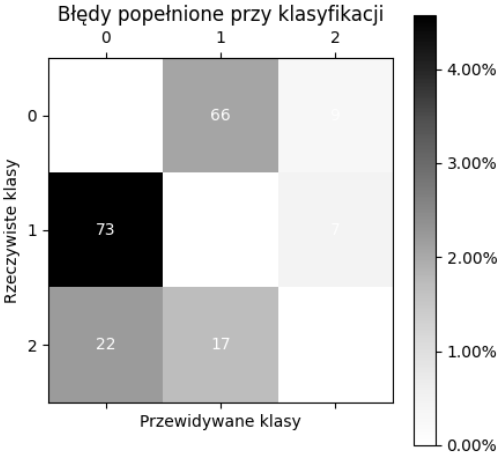
Rys. 4.1.5. – Dokładność klasyfikacji w funkcji liczby najbliższych sąsiadów k, model kNN. Zbadane wartości k: 1, 3, 5, 7, 10, 20, 30, 50, 100. Granatowe krzywe przedstawiają wyniki uzyskane dla 3 klas, czerwone krzywe przedstawiają wyniki uzyskane dla 36 klas. Krzywe wykreślone przerywaną linią przedstawiają wyniki uzyskane dla zastosowania normy miejskiej. Krzywe wykreślone ciągłą linią przedstawiają wyniki uzyskane dla zastosowania normy euklidesowej.

4.2. Klasyfikacja liniowa SVM

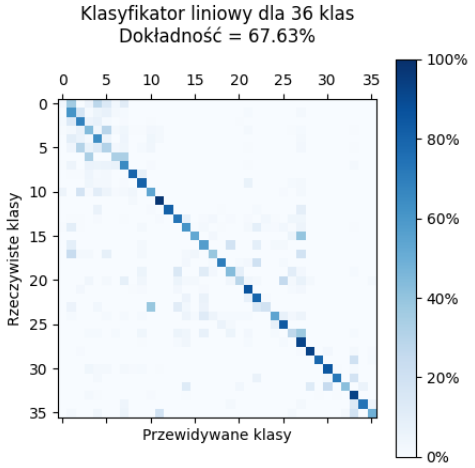
Na wykresach poniżej widoczne są wyniki klasyfikacji dla 3 klas oraz 36 klas. Wyniki przedstawiono w formie wizualizacji macierzy konfuzji. Dla obu klasyfikacji przedstawiono również macierz popełnionych błędów. Wykreślono wartości funkcji straty oraz dokładność w kolejnych iteracjach.



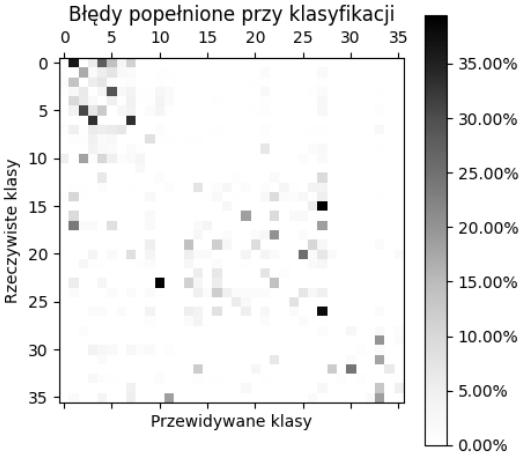
Rys. 4.2.1. – Macierz konfuzji dla modelu liniowego SVM, podział na 3 klasy, za metodę minimalizacji funkcji straty przyjęto local search



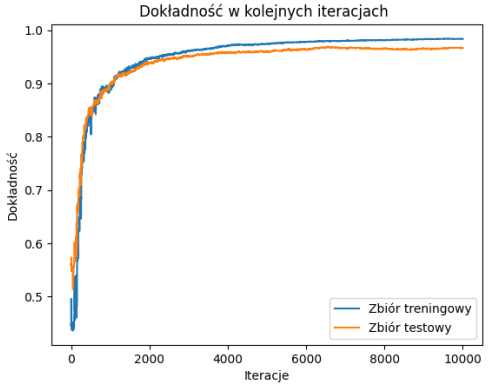
Rys. 4.2.2. – Macierz błędu dla modelu liniowego SVM, podział na 3 klasy, za metodę minimalizacji funkcji straty przyjęto local search



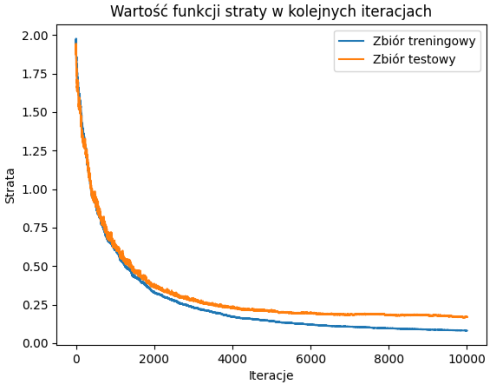
Rys. 4.2.3. – Macierz konfuzji dla modelu liniowego SVM, podział na 36 klas, za metodę minimalizacji funkcji straty przyjęto local search



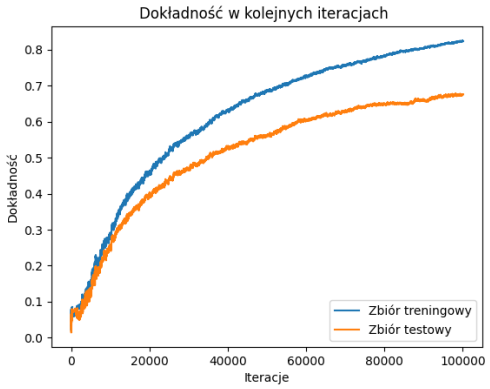
Rys. 4.2.4. – Macierz błędu dla modelu liniowego SVM, podział na 36 klas, za metodę minimalizacji funkcji straty przyjęto local search



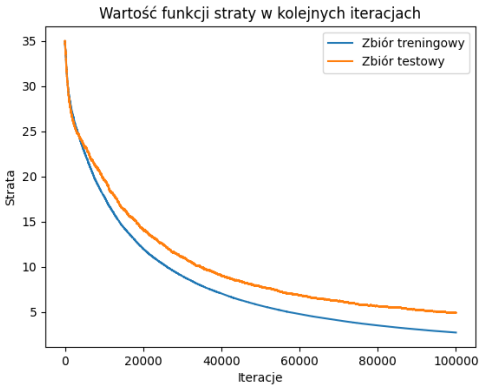
Rys. 4.2.5. – Dokładność modelu liniowego SVM w kolejnych iteracjach, podział na 3 klasy, za metodę minimalizacji funkcji straty przyjęto local search



Rys. 4.2.6. – Wartość funkcji straty modelu liniowego SVM w kolejnych iteracjach, podział na 3 klasy, za metodę minimalizacji funkcji straty przyjęto local search



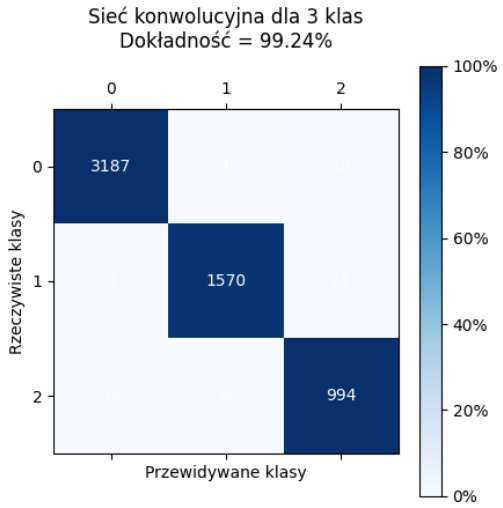
Rys. 4.2.7. – Dokładność modelu liniowego SVM w kolejnych iteracjach, podział na 36 klas, za metodę minimalizacji funkcji straty przyjęto local search



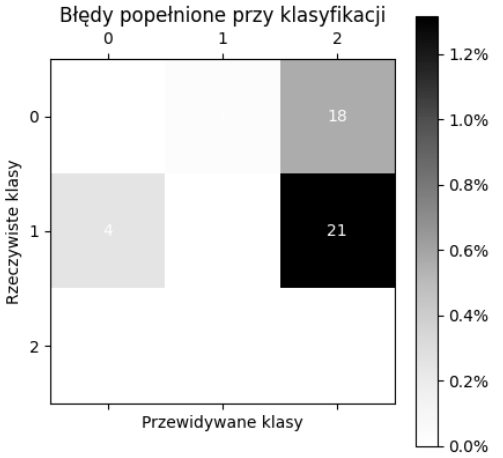
Rys. 4.2.8. – Wartość funkcji straty modelu liniowego SVM w kolejnych iteracjach, podział na 36 klas, za metodę minimalizacji funkcji straty przyjęto local search

4.3. Klasyfikacja z użyciem konwolucyjnej sieci neuronowej

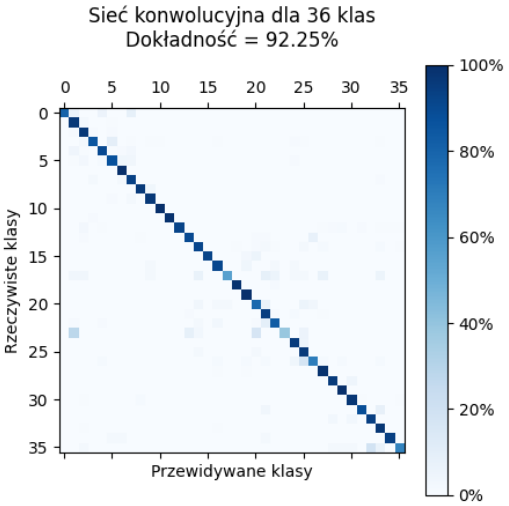
Na wykresach poniżej widoczne są wyniki klasyfikacji dla 3 klas oraz 36 klas. Wyniki przedstawiono w formie wizualizacji macierzy konfuzji. Dla obu klasyfikacji przedstawiono również macierz popełnionych błędów. Wykreślono wartości funkcji straty oraz dokładność w kolejnych epokach.



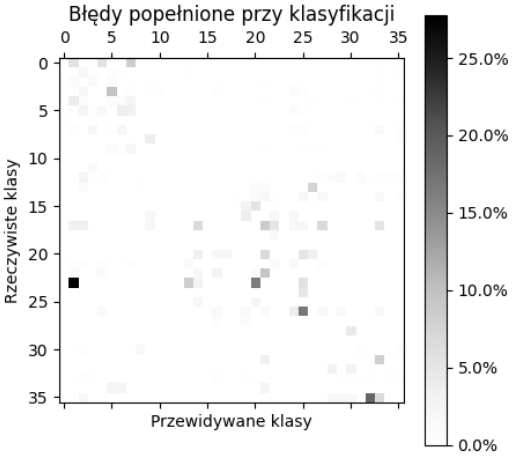
Rys. 4.3.1. – macierz konfuzji dla modelu CNN, podział na 3 klasy



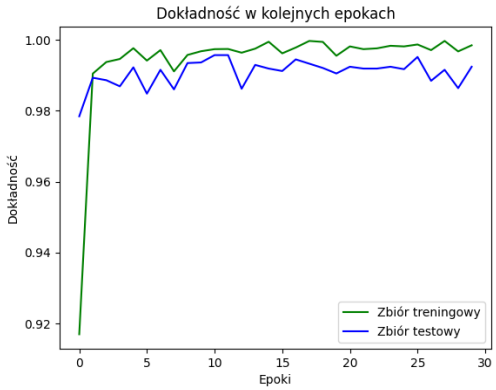
Rys. 4.3.2. – macierz błędu dla modelu CNN, podział na 3 klasy



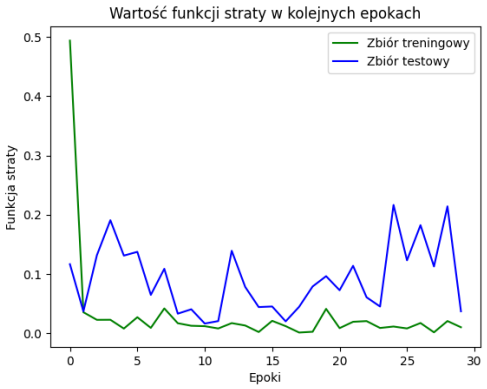
Rys. 4.3.3. – macierz konfuzji dla modelu CNN, podział na 36 klas



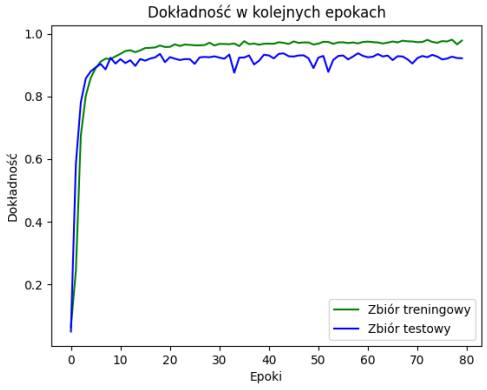
Rys. 4.3.4. – macierz błędu dla modelu CNN, podział na 36 klas



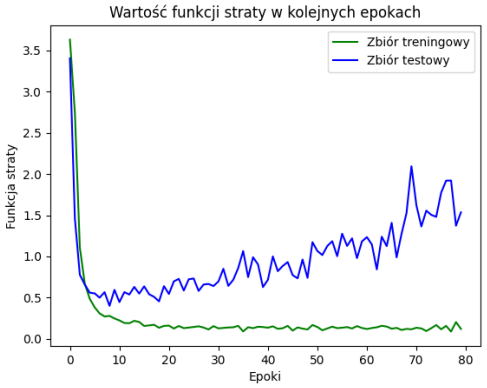
Rys. 4.3.5. – Dokładność modelu CNN w kolejnych iteracjach, podział na 3 klasy



Rys. 4.3.6. – Wartość funkcji straty modelu CNN w kolejnych iteracjach, podział na 3 klasy



Rys. 4.3.7. – Dokładność modelu CNN w kolejnych iteracjach, podział na 36 klas



Rys. 4.3.8. – Wartość funkcji straty modelu CNN w kolejnych iteracjach, podział na 36 klas

5. Wnioski

5.1. Klasyfikacja k–najbliższych sąsiadów

Na podstawie wyników dla kNN można stwierdzić, że klasyfikator osiąga przyzwoite wyniki w przypadku klasyfikacji 3 klas. Jest to spowodowane tym, że klasy te różnią się ogólnym kształtem i kolorem.

W przypadku klasyfikacji dla 36 wyszczególnionych klas, których obiekty różnią się od siebie w niewielkim stopniu, kNN poradził sobie znacznie gorzej. Na podstawie macierzy błędów dla 36 klas (rys. 4.1.4.) zauważono, że błędna etykieta, którą klasyfikator przypisywał do obiektu testowego najczęściej należała do tej samej nadrzędnej klasy co poprawny rezultat.

Z rys. 4.1.5. wynika, że zwiększenie liczby sąsiadów miało negatywny wpływ na dokładność modelu. Pokazuje on także, że wykorzystanie normy euklidesowej pozwoliło uzyskać lepszy rezultat niż wykorzystanie normy miejskiej.

5.2. Klasyfikacja liniowa SVM

Na podstawie wyników uzyskanych dla SVM stwierdzono, że klasyfikator osiąga bardzo dobre wyniki w przypadku klasyfikacji 3 klas. Z macierzy popełnionych błędów dla 3 klas (rys. 4.2.2.) wynika, że klasyfikator najczęściej mylił znak ostrzegawczy ze znakiem zakazu. Analizując wykresy dokładności i funkcji straty w kolejnych iteracjach dla 3 klas, można stwierdzić, że dobrze dobrano hiperparametry.

W przypadku klasyfikacji dla 36 wyszczególnionych klas, których obiekty różnią się od siebie w niewielkim stopniu, SVM poradził sobie gorzej. Z macierzy konfuzji (rys. 4.2.3.) i błędów (rys. 4.2.4.) dla 36 klas wynika, że klasyfikator najwięcej błędów popełniał przy klasyfikacji znaków należących do znaków zakazu. Może być to spowodowane małymi różnicami między klasami reprezentującymi znaki zakazu. Przykładem tego są wszystkie znaki ograniczające prędkość, które różnią się jedynie liczbą w środku znaku. Analizując wykresy dokładności w kolejnych iteracjach dla 36 klas (rys. 4.2.7.), można zauważyć, że dokładność dla zbioru testowego nie podąża ściśle za dokładnością zbioru treningowego. Zjawisko to sygnalizuje niewielkie przeuczenie modelu.

5.3. Klasyfikacja z użyciem konwolucyjnej sieci neuronowej

Na podstawie analizy wykresu dokładności predykcji modelu (rys. 4.3.5.) oraz wykresu funkcji straty (rys. 4.3.6.) dla kolejnych epok stwierdzono, że model oparty na konwolucyjnej sieci neuronowej dla problemu rozróżniania 3 ogólnych rodzajów znaków drogowych (ostrzegawcze, zakazu, nakazu) osiągnął wysoką dokładność już po pierwszej epoce. Dla kolejnych epok dokładność predykcji modelu raz nieco wzrasta, a raz nieco spada. Jednakże nie zaobserwowano ani znaczącego polepszenia się, ani znaczącego pogorszenia się skuteczności modelu.

Dla klasyfikacji 36 klas model osiąga bardzo dobre wyniki i popełnia jedynie błędy na pojedynczych klasach. Przykładem tego jest błędna klasyfikacja znaku ostrzegawczego "przejście dla pieszych" jako znak zakazu "ograniczenie prędkości do 30 km/h". Analizując wykresy funkcji straty w kolejnych epokach dla 36 klas (rys. 4.3.8.), zauważono zjawisko nadmiernego dopasowania (ang. *overfitting*) w modelu. Widoczne jest ono na podstawie tego, że krzywa straty dla danych treningowych maleje oraz krzywa straty dla danych testowych rośnie.

6. Podsumowanie

	3 klasy				*36 klas*			
Klasyfikator	*dokładność*	*precyzja*	*zwrot*	*f1 miara*	*dokładność*	*precyzja*	*zwrot*	*f1 miara*
$kNN_{k=1,n=1}$	0,8109	0,7972	0,7724	0,7846	0,2202	0,2100	0,2415	0,2247
$kNN_{k=1,n=2}$	0,8323	0,8198	0,8110	0,8154	0,2621	0,2467	0,2419	0,2443
Liniowy SVM	0,9665	0,9624	0,9674	0,9649	0,6763	0,6165	0,6725	0,6433
CNN	0,9924	0,9928	0,9868	0,9898	0,9225	0,8942	0,8924	0,8933

Rys. 6.1. – podsumowanie wyników pomiarów, przeprowadzonych w celu zbadania jakości klasyfikatorów w zależności od liczby zadanych klas do rozróżnienia.

Klasyfikator kNN jako model nieuczący porównuje dużą ilość wzorców w ramach klasyfikacji. Zaletą takiego podejścia jest brak treningu, który w przypadku modeli uczących na ogół jest procesem czasochłonnym. Klasyfikowanie wykorzystujące bezpośrednio bazę danych powoduje, że wraz ze zwiększaniem liczby próbek, kNN coraz lepiej przewiduje wynik. Jednakże coraz większa ilość danych w bazie powoduje coraz dłuższy czas trwania klasyfikacji. Model kNN uzyskał najgorsze wyniki spośród testowanych klasyfikatorów. Klasyfikator k–najbliższych sąsiadów nadaje się do klasyfikacji obiektów wyraźnie różniących się od siebie, czyli na przykład rozróżniania znaków drogowych na ostrzegawcze, zakazu i nakazu. Dla bardziej złożonej klasyfikacji, jaką jest rozróżnianie znaków na ich poszczególne rodzaje, kNN osiąga na tyle niską dokładność, że odradza się stosowanie go do omawianego problemu.

Klasyfikator liniowy SVM w porównaniu do kNN osiągnął znacznie lepsze wyniki. Klasyfikacja prosta z podziałem na 3 klasy osiągnęła dokładność równą 96,65%. Ponadto zastosowanie metody local search poskutkowało względnie małą liczbą iteracji do minimalizacji funkcji straty. Natomiast dla klasyfikacji 36 klas SVM, pomimo 100 tysięcy iteracji, nie osiągnął zadowalającej dokładności (67,63%).

Klasyfikator oparty na konwolucyjnej sieci neuronowej poradził sobie najlepiej ze wszystkich zastosowanych klasyfikatorów, zarówno dla klasyfikacji znaków drogowych na 3 klasy, jak i 36 klas. CNN dla podziału na 3 klasy osiągnęła prawie stuprocentową dokładność. Dla podziału na 36 klas sieć osiągnęła dokładność klasyfikacji równą 92,25%.

Podsumowując, spośród badanych klasyfikatorów, konwolucyjna sieć neuronowa dla klasyfikacji znaków drogowych osiągnęła najlepsze wyniki.

7. Źródła

[1] A. Géron, [tłum.] K. Sawka: Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow. Wydanie II. Helion SA, 2020.

[2] Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network. https://www.upgrad.com/blog/basic-cnn-architecture/

[3] CS231n: Deep Learning for Computer Vision. https://cs231n.github.io/classification/

[4] Flatten. https://tensorspace.org/html/docs/layerFlatten.html

[5] Konwolucyjna sieć neuronowa (CNN). https://www.tensorflow.org/tutorials/images/cnn?hl=pl

[6] Konwolucyjne sieci neuronowe (CNN), głębokie uczenie i komputerowe rozpoznawanie obrazu. https://www.intel.pl/content/www/pl/pl/internet-of-things/computer-vision/convolutional-neural-networks.html

[7] K. Sopyła: Precision, recall i F1 – miary oceny klasyfikatora. https://ksopyla.com/data-science/precision-recall-f1-miary-oceny-klasyfikatora/

[8] Materiały udostępnione na kursie “Sztuczna Inteligencja - 2023”. udhttps://enauczanie.pg.edu.pl/moodle/course/view.php?id=29579

[9] M. Mamczur: Jak działają konwolucyjne sieci neuronowe (CNN)? https://miroslawmamczur.pl/jak-dzialaja-konwolucyjne-sieci-neuronowe-cnn/

[10] Multiclass Classification Using Support Vector Machines. https://www.baeldung.com/cs/svm-multiclass-classification

[11] Support vector machine. https://en.wikipedia.org/wiki/Support_vector_machine

8. Numeracja klas znaków drogowych

8.1. Podział na 3 klasy

0 - znak zakazu
1 - znak ostrzegawczy
2 - znak nakazu

8.2. Podział na 36 klas

0 - znak zakazu: Ograniczenie prędkości do 20 km/h
1 - znak zakazu: Ograniczenie prędkości do 30 km/h
2 - znak zakazu: Ograniczenie prędkości do 50 km/h
3 - znak zakazu: Ograniczenie prędkości do 60 km/h
4 - znak zakazu: Ograniczenie prędkości do 70 km/h
5 - znak zakazu: Ograniczenie prędkości do 80 km/h
6 - znak zakazu: Ograniczenie prędkości do 100 km/h
7 - znak zakazu: Ograniczenie prędkości do 120 km/h
8 - znak zakazu: Zakaz wyprzedzania
9 - znak zakazu: Zakaz wyprzedzania przez samochody ciężarowe
10 - znak zakazu: Zakaz ruchu w obu kierunkach
11 - znak zakazu: Zakaz wjazdu samochodów ciężarowych
12 - znak zakazu: Zakaz wjazdu
13 - znak ostrzegawczy: Skrzyżowanie z drogą podporządkowaną
14 - znak ostrzegawczy: Inne niebezpieczeństwo
15 - znak ostrzegawczy: Niebezpieczny zakręt w lewo
16 - znak ostrzegawczy: Niebezpieczny zakręt w prawo
17 - znak ostrzegawczy: Niebezpieczne zakręty - pierwszy w lewo
18 - znak ostrzegawczy: Nierówna droga
19 - znak ostrzegawczy: Śliska jezdnia
20 - znak ostrzegawczy: Zwężenie jezdni - prawostronne
21 - znak ostrzegawczy: Roboty na drodze
22 - znak ostrzegawczy: Sygnały świetlne
23 - znak ostrzegawczy: Przejście dla pieszych
24 - znak ostrzegawczy: Dzieci
25 - znak ostrzegawczy: Rowerzyści
26 - znak ostrzegawczy: Oszronienie jezdni
27 - znak ostrzegawczy: Dzikie zwierzęta
28 - znak nakazu: Nakaz jazdy w prawo za znakiem
29 - znak nakazu: Nakaz jazdy w lewo za znakiem
30 - znak nakazu: Nakaz jazdy prosto
31 - znak nakazu: Nakaz jazdy prosto lub w prawo
32 - znak nakazu: Nakaz jazdy prosto lub w lewo
33 - znak nakazu: Nakaz jazdy z prawej strony znaku
34 - znak nakazu: Nakaz jazdy z lewej strony znaku
35 - znak nakazu: Ruch okrężny