Projekt – sieci neuronowe	Data złożenia projektu: 27.05.2022
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.

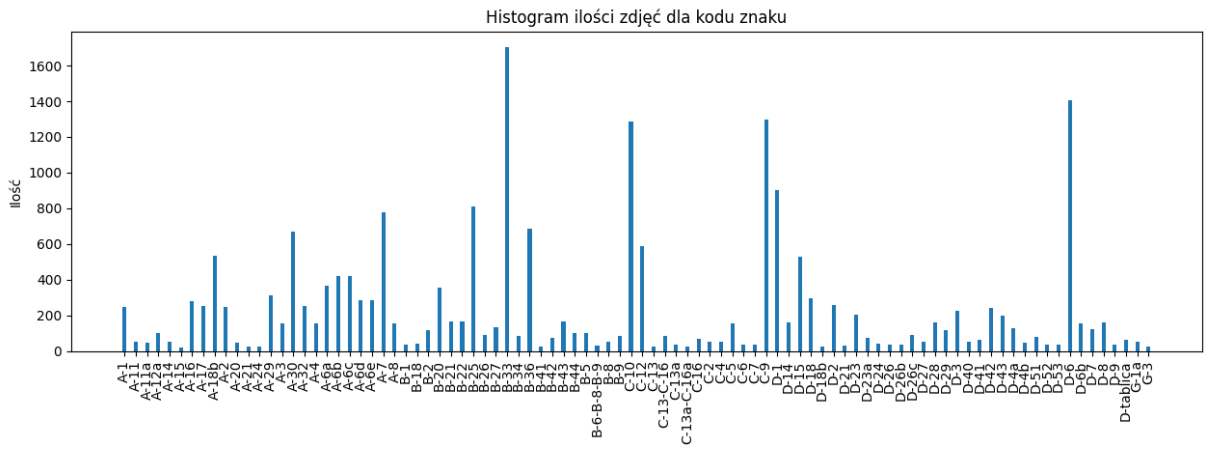
Analizowany zbiór danych składa się z 21032 rekordów. Znajdziemy w nim 92 zmienne wejściowe. Liczność danej klasy dla zmiennej wyjściowej zaprezentowana jest na poniższym histogramie. Problematyczne staje się przedstawienie podstawowych statystyk ze względu, iż rozpatrywany jest problem klasyfikacji 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-– 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

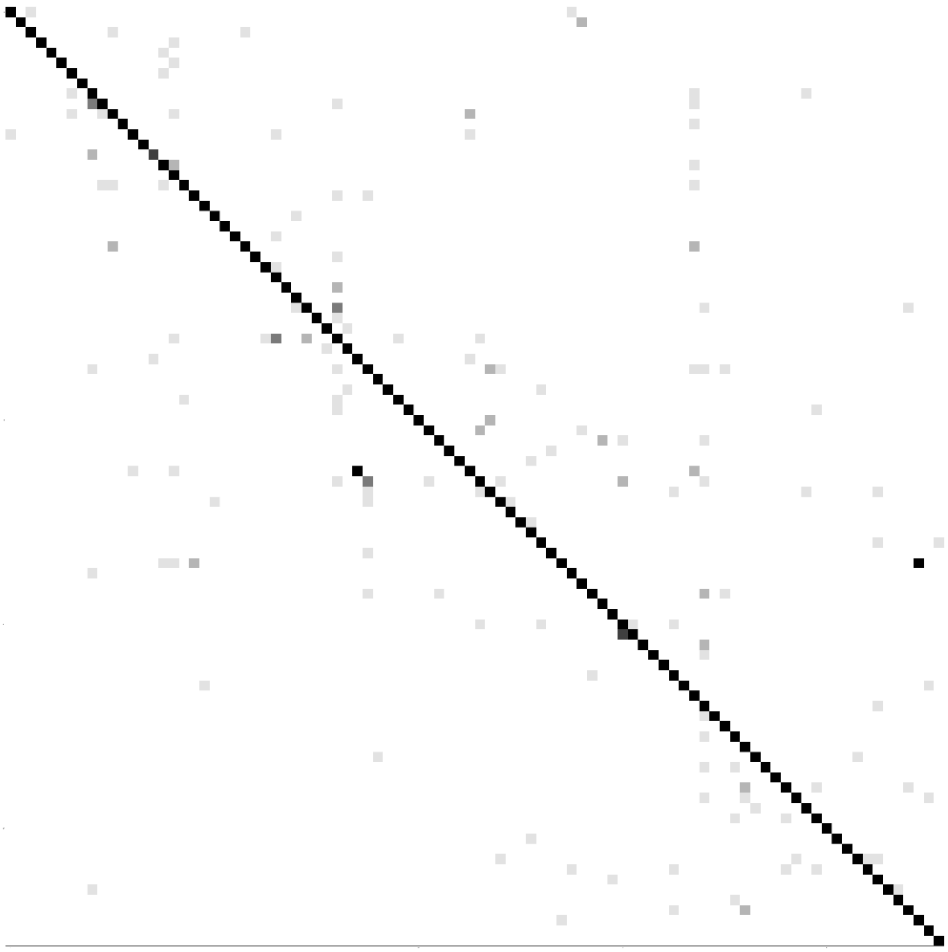
Rys 1. Przedstawienie klas

Obróbka danych

Otrzymany zbiór posiadał już podział na testujący oraz treningowy. Ilość rekordów w poszczególnych zbiorach rozkładała się w proporcji 90:10. 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 dla wybranego najlepszego modelu 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



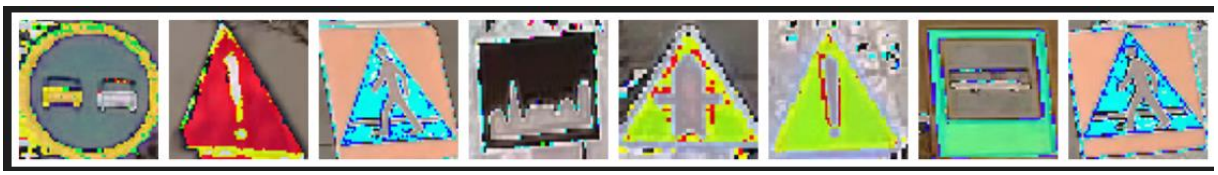
Rys 2. Graficzna macierz dopasowania modelu

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 byłoby wręcz koniecznością ze względu na dostępność danych w tych modelach



Rys 3. Zastosowany filtr VGG16

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

Rys 4. Dane – obrazy wejściowe

Opis zastosowanych sieci neuronowych

W projekcie zastosowano konwolucyjną sieć neuronową

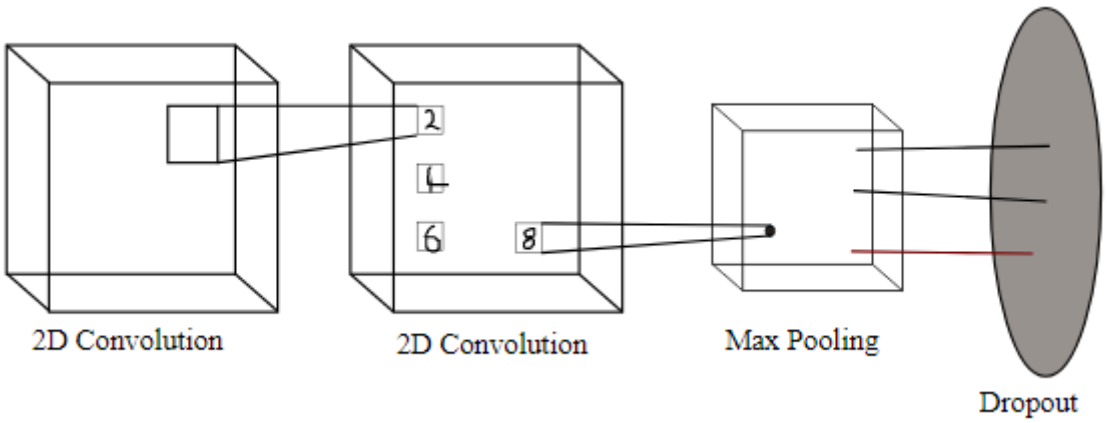
Ilość epok uczenia została empirycznie dobrana na 25, mniejsza liczba powodowała niską dokładność a większa powodowała przetrenowanie co skutkowało rozbieżnością na poziomie kilkudziesięciu procent w testach dokładności na danych testujących i walidujących podczas uczenia modelu

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 konstrukcji macierzy konfuzji używamy możliwości oferowanych przez bibliotekę sklearn. Do operacji matematycznych wykorzystujemy bibliotekę numpy.

Schemat sieci podzielono na dwie części: część ekstrakcji cech oraz klasyfikacji. Część wyodrębniająca cechy jest zapętlona dwa razy zanim trafi do części klasyfikacji.

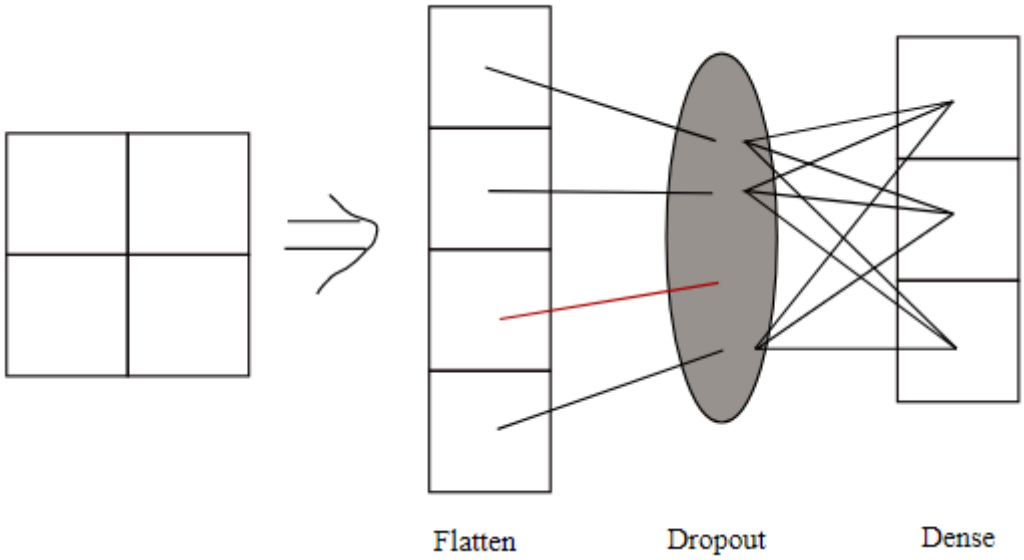
W modelowym rozwiązaniu dwie pierwsze połączone warstwy konwolucji mają 32 filtry oraz rozmiar jądra 5x5. Natomiast druga iteracja posiada tych filtrów 64 a jądro ma rozmiar 3x3. Warstwy każdorazowo mają funkcję aktywacji ReLU (GeLU przyniosło gorsze wyniki).

Warstwa max pooling ma rozmiar 2x2, a współczynnik losowo usuwanych połączeń wynosi 25% zapobiegając przetrenowaniu



Rys 4. Wyodrębnianie cech

Następnie dokonywana jest klasyfikacja



Rys 5. Klasyfikacja

Wyjściowa warstwa w poprzednim procesie jest poddawana transformacji na 1 wymiar (Flatten). Następnie taki sam procent połączeń, jak poprzednio, jest losowo usuwana (Dropout) i ostatecznie łączona każdy z każdym w warstwie Dense.

Modele analizowanych architektur

Architektura „1”:

Conv2D	32 filtry, 5x5 kernel, ReLU f. akt
Conv2D	32 filtry, 5x5 kernel, ReLU f. akt
MaxPooling	2x2 pool size
Dropout	0.25 rate
Flatten	
Dense	256, ReLU f.akt
Dropout	0.5 rate
Dense	SoftMAX

Architektura „2”:

Conv2D	32 filtry, 5x5 kernel, ReLU f. akt
Conv2D	32 filtry, 5x5 kernel, ReLU f. akt
MaxPooling	2x2 pool size
Dropout	0.25 rate
Conv2D	64 filty, 3x3 kernel, ReLU f.akt
Conv2D	64 filty, 3x3 kernel, ReLU f.akt
MaxPooling	2x2 pool size
Dropout	0.25 rate
Flatten	
Dense	256, ReLU f.akt
Dropout	0.5 rate
Dense	SoftMAX

Architektura „3”:

Conv2D	32 filtry, 5x5 kernel, ReLU f. akt
MaxPooling	2x2 pool size
Conv2D	64 filty, 3x3 kernel, ReLU f.akt
MaxPooling	2x2 pool size
Conv2D	128 filty, 3x3 kernel, ReLU f.akt
MaxPooling	2x2 pool size
Flatten	
Dense	256, ReLU f.akt
Dropout	0.3 rate
Dense	SoftMAX

Architektura „4”:

Conv2D	32 filtry, 5x5 kernel, ReLU f. akt
MaxPooling	2x2 pool size
Dropout	0.25 rate
Conv2D	64 filty, 3x3 kernel, ReLU f.akt
MaxPooling	2x2 pool size
Dropout	0.25 rate
Conv2D	128 filty, 3x3 kernel, ReLU f.akt
MaxPooling	2x2 pool size
Dropout	0.25 rate
Flatten	
Dense	256, ReLU f.akt
Dropout	0.3 rate
Dense	SoftMAX
Dense	SoftMAX

W wykonywanym projekcie analizujemy cztery modele architektur konwolucyjnych sieci neuronowych. Architektura 2 jest rozszerzoną wersją architektury 1. Różnicą jest podwojenie części ekstrakcji cech, a więc dodanie dodatkowej grupy warstw składającej się z dwóch warstw konwolucji, jednej MaxPoolingu oraz Dropoutu. W przypadku architektury trzeciej oraz czwartej różnicą są dodatkowe trzy warstwy Dropoutu. W przeciwieństwie do architektury pierwszej oraz drugiej warstwa konwolucji jest pojedyncza.

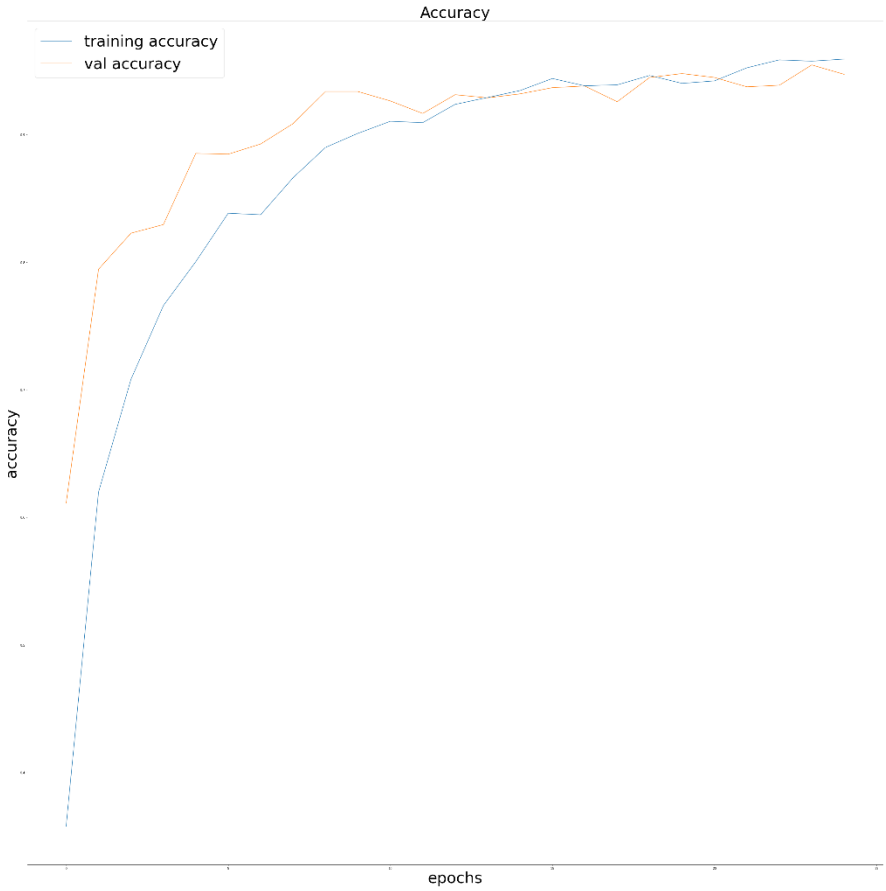
Metryki – porównanie

Poniżej prezentujemy wyniki porównawcze metryk dla czterech analizowanych modeli architektur sieci konwolucyjnych, które zostały przedstawione w postaci tabelarycznej powyżej. W każdym z analizowanych przypadków analizujemy wynik uzyskany po 25 epokach. Prezentujemy również wykres pokazujący następujące zmiany wraz z kolejnymi epokami.

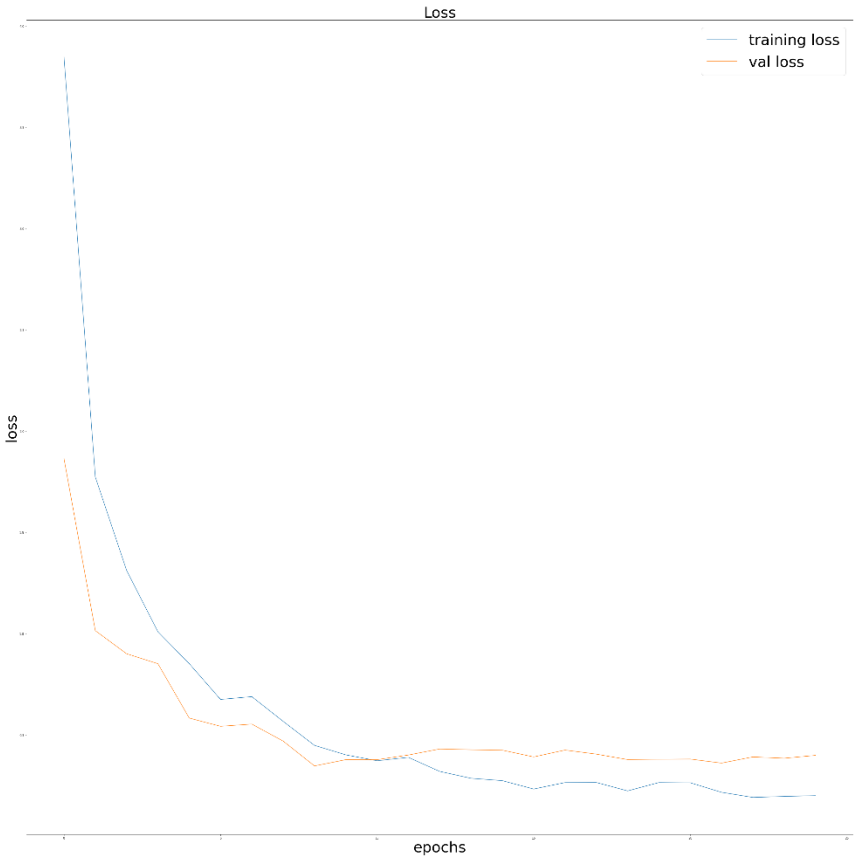
Architektura „1”

- Loss(%): 0.20060360431671143
- Loss: 0.39946460723876953
- Binary accuracy(%): 0.9991307258605957
- Categorical accuracy(%): 0.9591823220252991
- AUC(%): 0.9941418766975403
- AUC: 0.98846036195755
- Mean absolute error(%): 0.010048648846335709
- Mean absolute error: 0.0013406483922153711
- Poisson(%): 0.012958411127328873
- Poisson: 0.014552178792655468

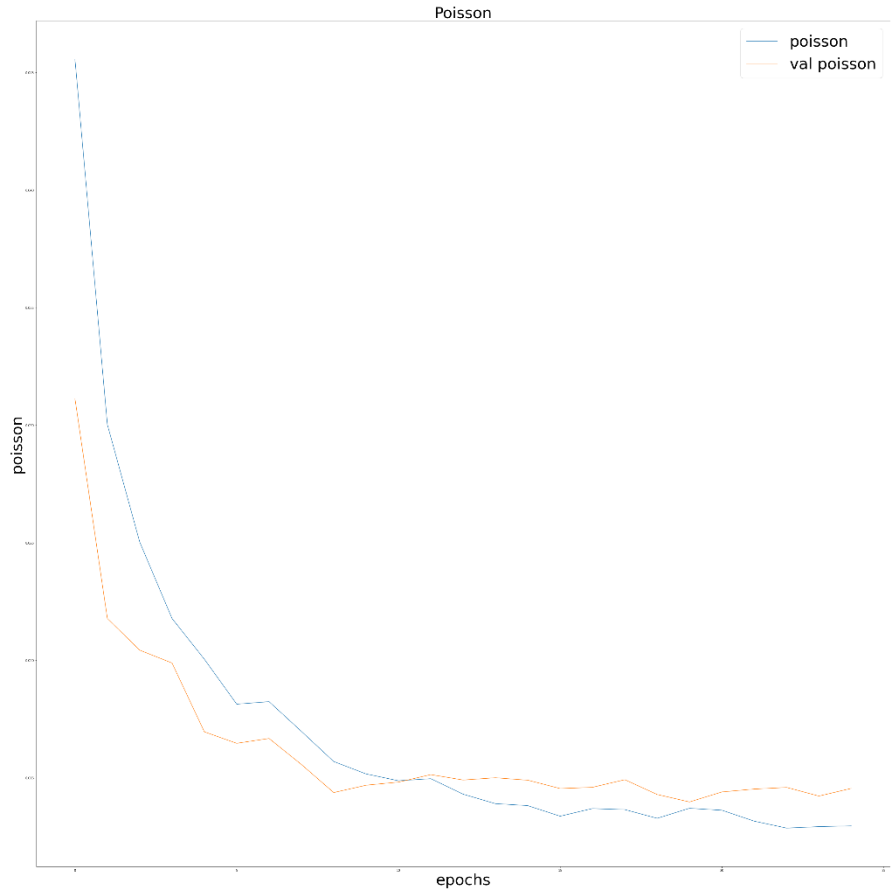
Z analizowanej architektury uzyskano poniższe wyniki przedstawione na Rys. 6 oraz Rys. 7 uzyskując poniższe dokładności.



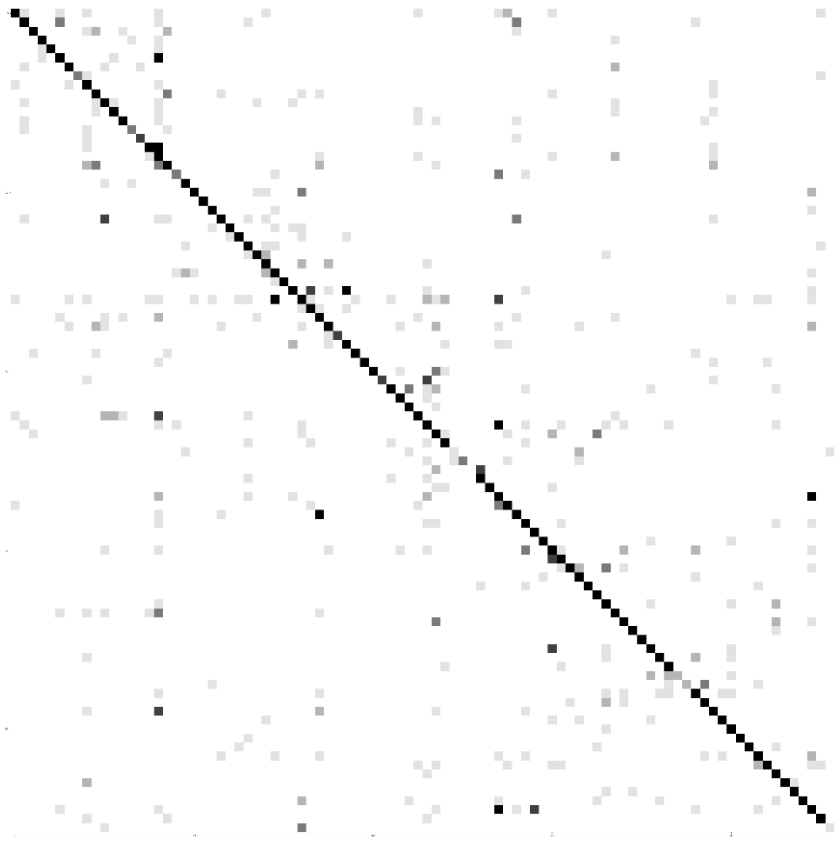
Rys 6. Analiza dokładności w modelu „2”



Rys 7. Analiza funkcji kary w modelu „2”



Rys 7. Analiza wartości funkcji Poisson w modelu „2”

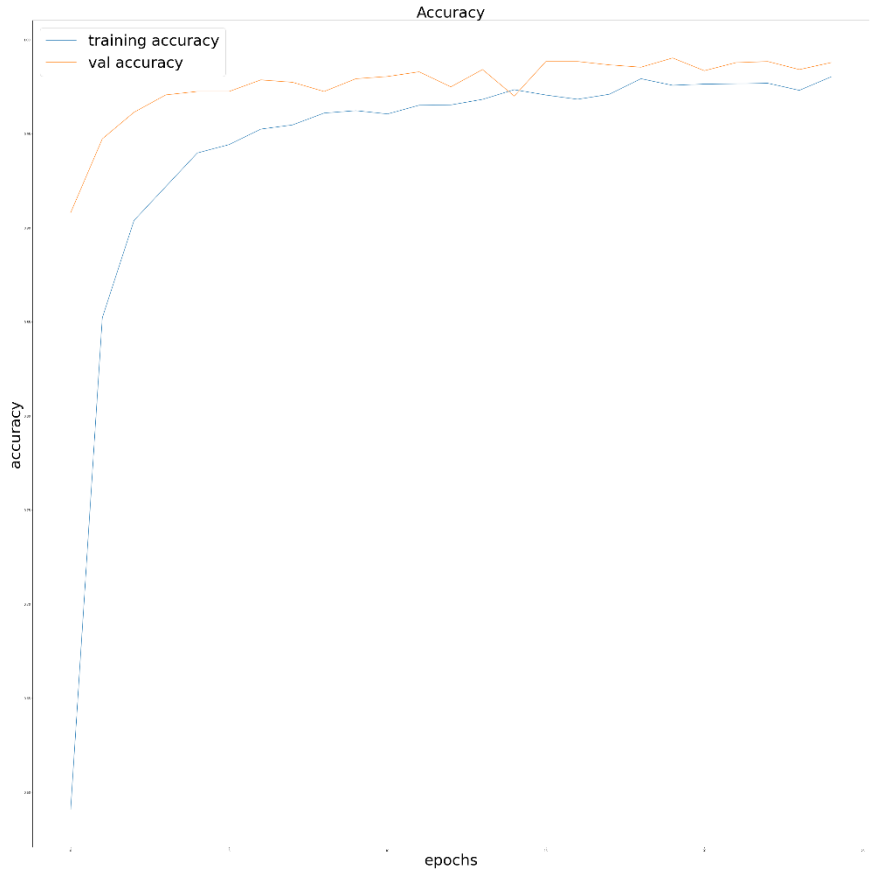


Rys 7. Analiza macierzy konfuzji w modelu „2”

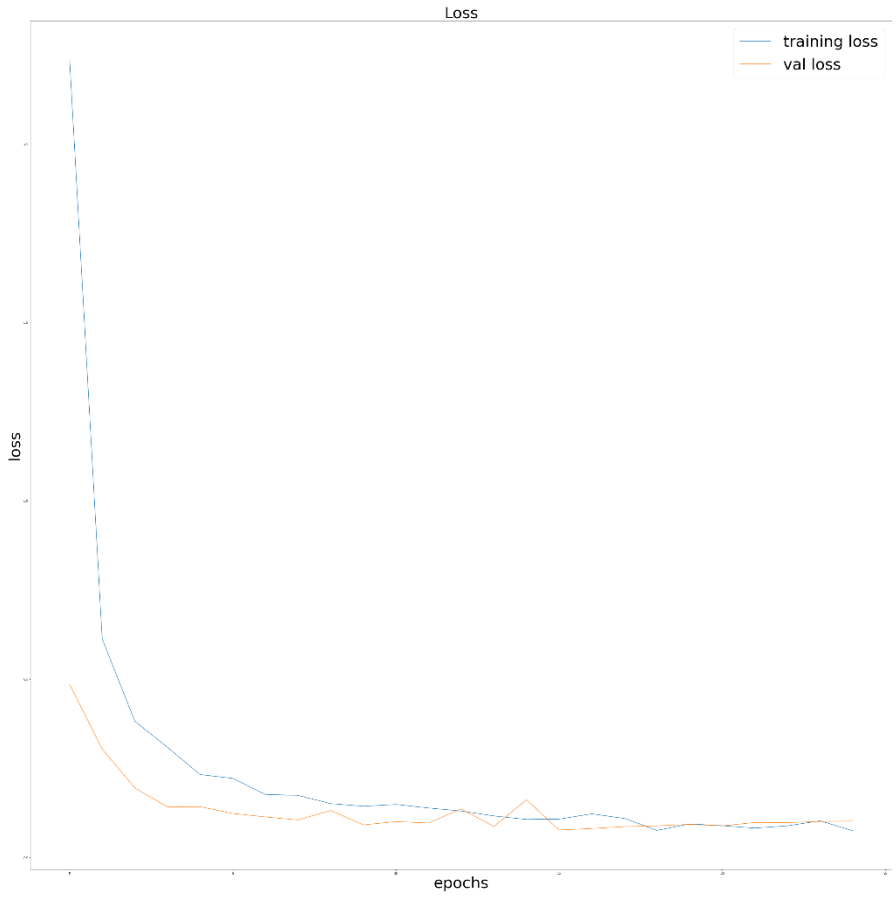
Architektura „2”

- Loss(%): 0.07454809546470642
- Loss: 0.10168413817882538
- Binary accuracy(%): 0.9995869994163513
- Categorical accuracy(%): 0.980153501033783
- AUC(%): 0.998315155506134
- AUC: 0.9968986511230469
- Mean absolute error(%): 0.0005751837743446231
- Mean absolute error: 0.0003583678335417062
- Poisson(%): 0.011674562469124794
- Poisson: 0.011887199245393276

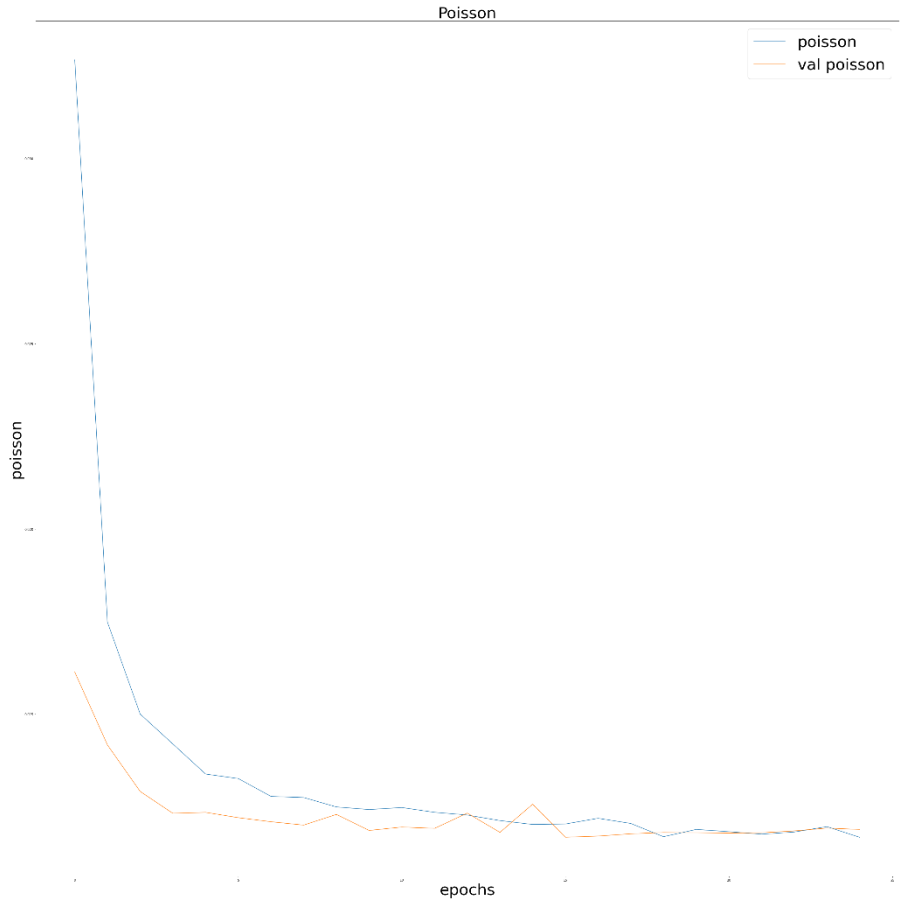
Z analizowanej architektury uzyskano poniższe wyniki przedstawione na Rys. 6 oraz Rys. 7 uzyskując poniższe dokładności.



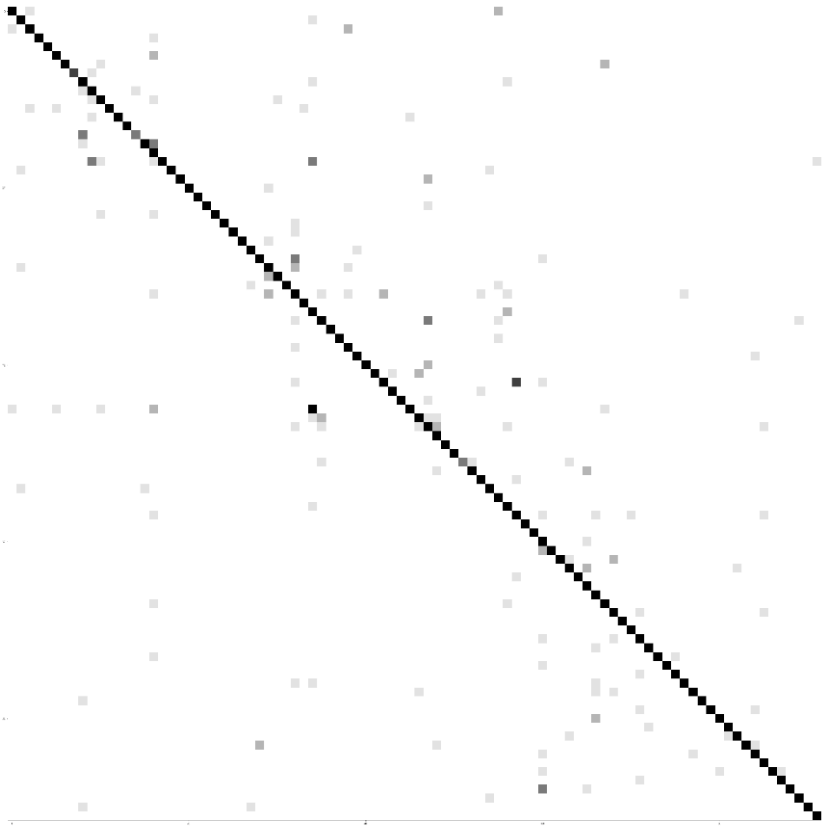
Rys 6. Analiza dokładności w modelu „2”



Rys 7. Analiza funkcji kary w modelu „2”



Rys 7. Analiza wartości funkcji Poisson w modelu „2”

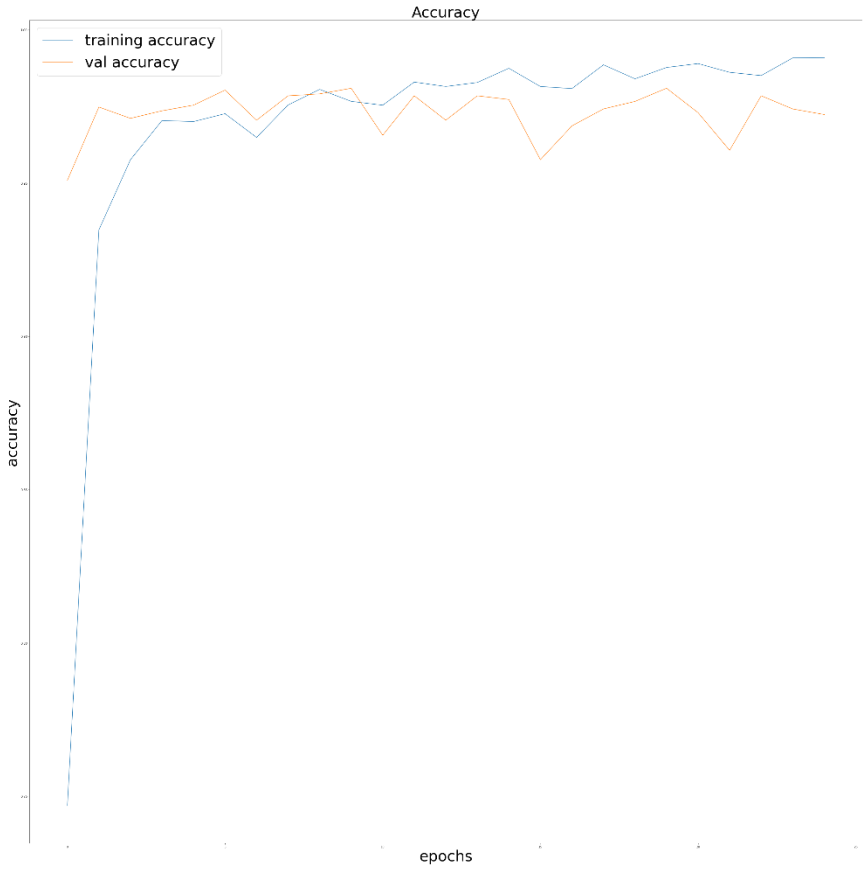


Rys 7. Analiza macierzy konfuzji w modelu „2”

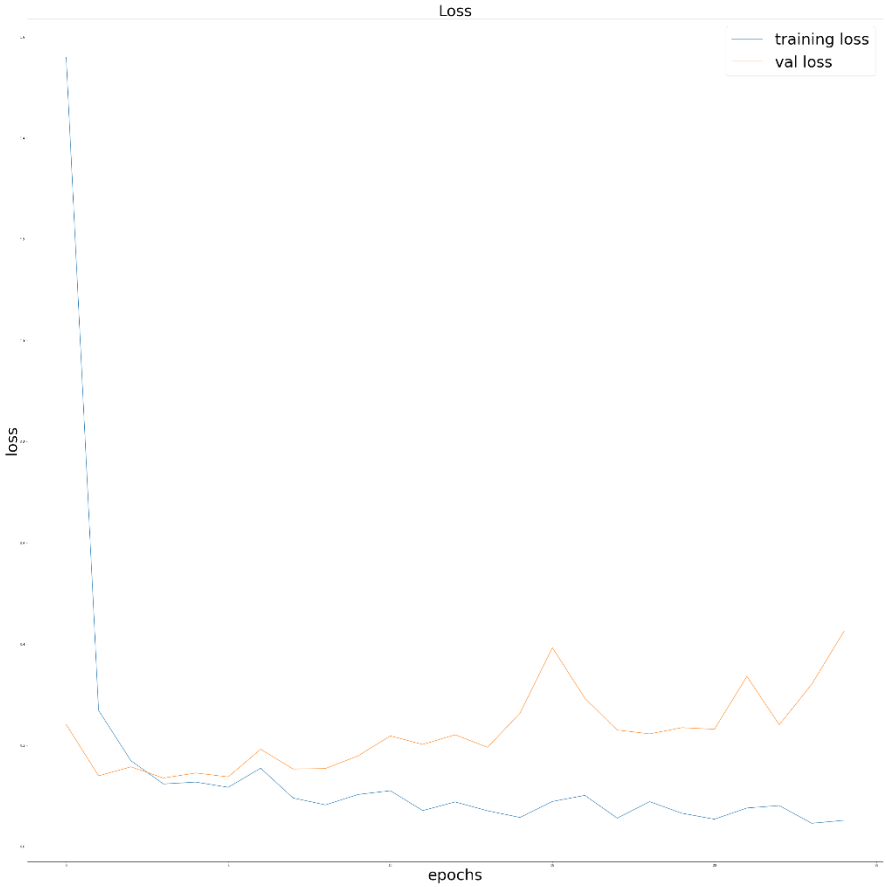
Architektura „3”

- Loss(%): 0.051595598459243774
- Loss: 0.42575162649154663
- Binary accuracy(%): 0.9998055696487427
- Categorical accuracy(%): 0.9909367561340332
- AUC(%): 0.9985980987548828
- AUC: 0.9916397929191589
- Mean absolute error(%): 0.00021987332729622722
- Mean absolute error: 0.0006526715005747974
- Poisson(%): 0.011358468793332577
- Poisson: 0.013577768579125404

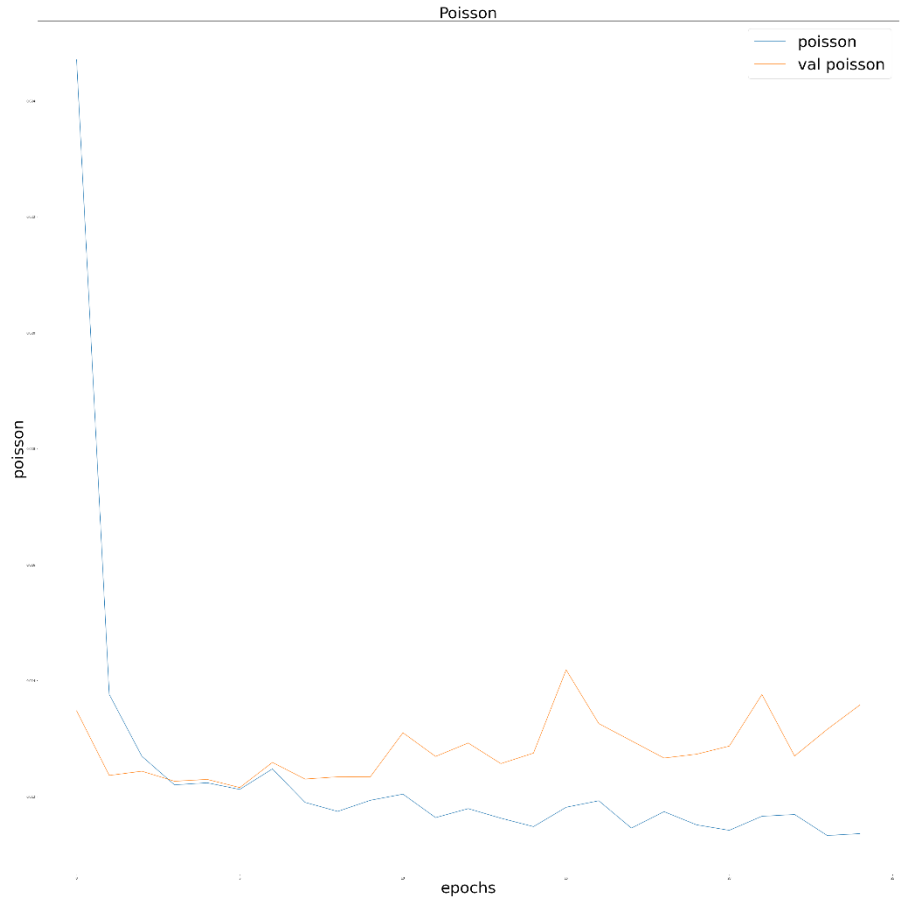
Z analizowanej architektury uzyskano poniższe wyniki przedstawione na Rys. 6 oraz Rys. 7 uzyskując poniższe dokładności.



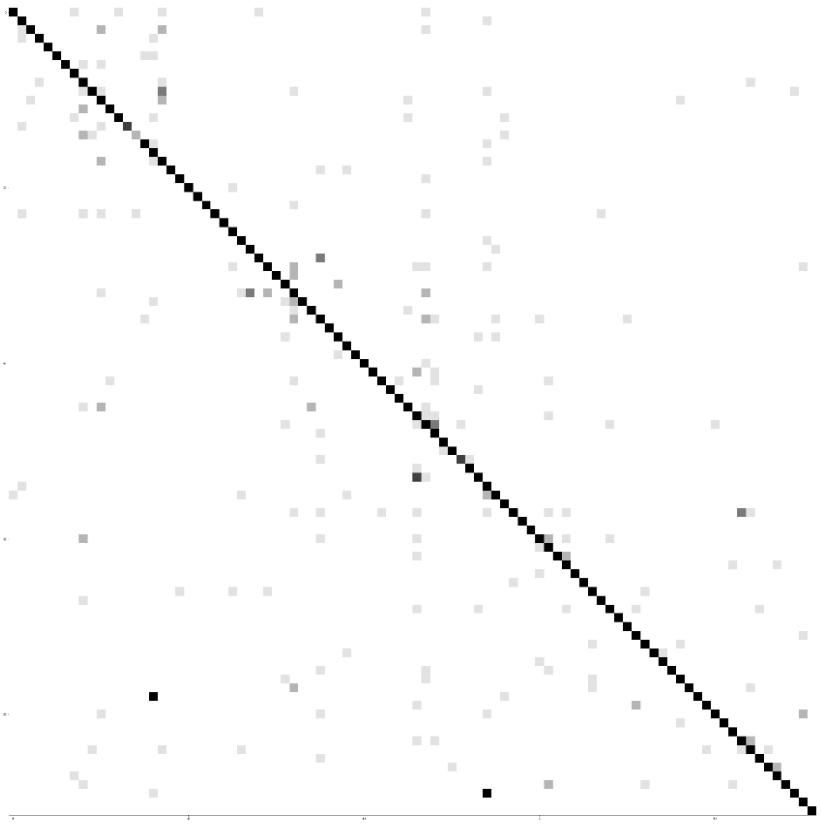
Rys 6. Analiza dokładności w modelu „3”



Rys 7. Analiza funkcji kary w modelu „3”



Rys 7. Analiza wartości funkcji Poisson w modelu „3”

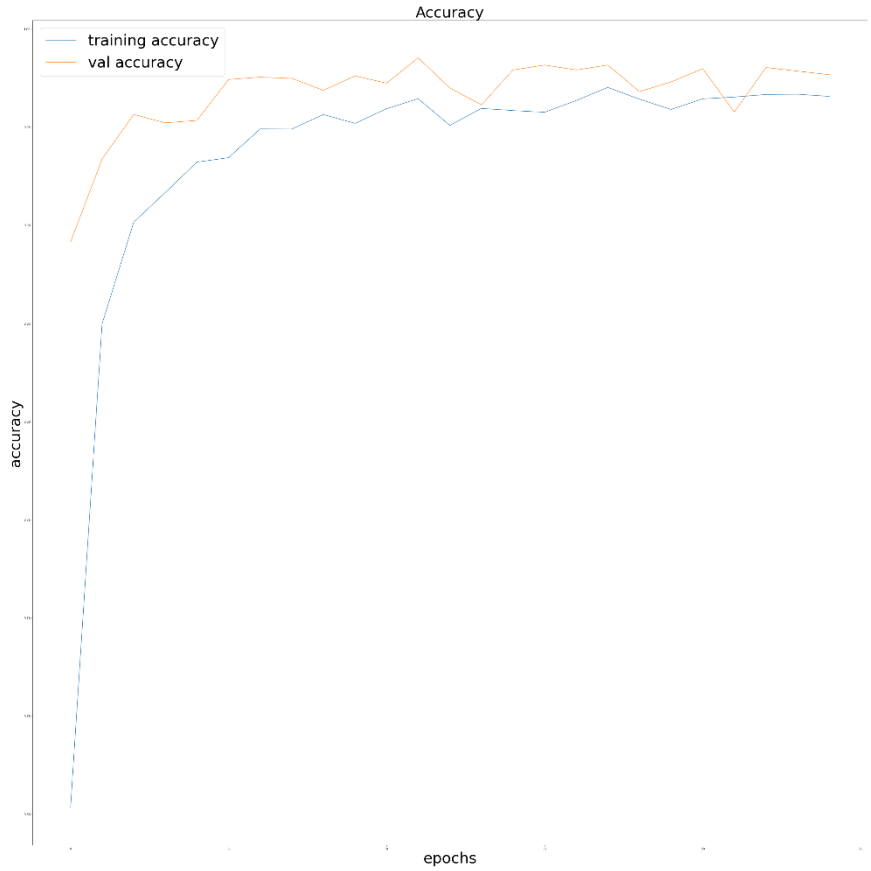


Rys 7. Analiza macierzy konfuzji w modelu „3”

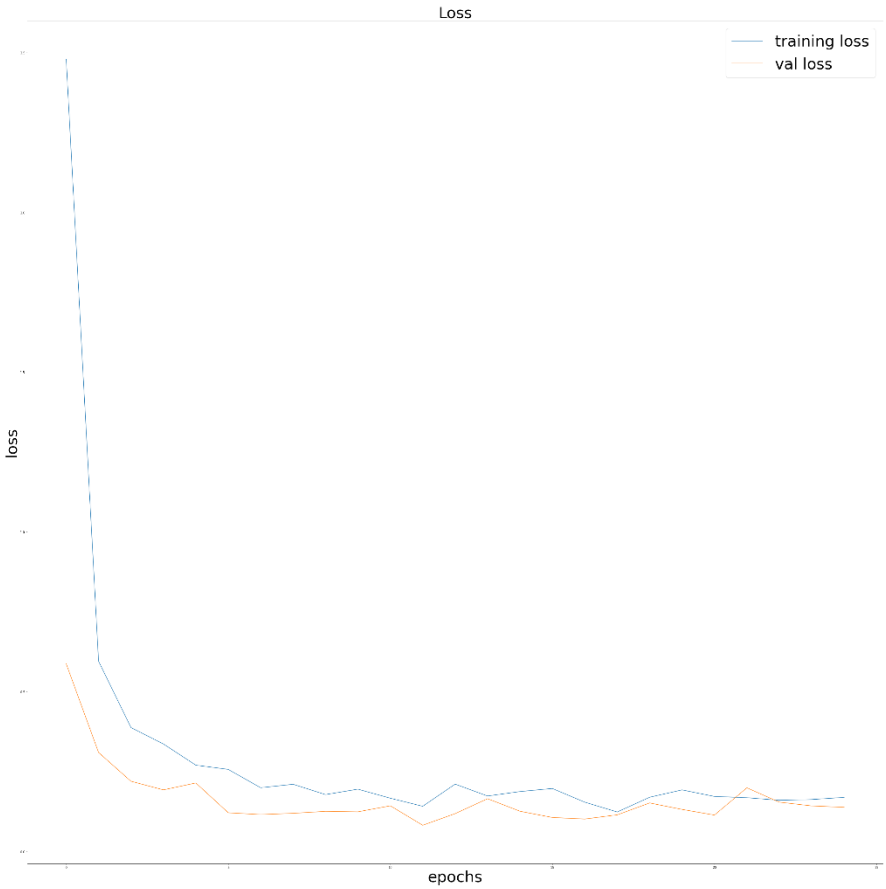
Architektura „4”

- Loss(%): 0.16994282603263855
- Loss: 0.13899561762809753
- Binary accuracy(%): 0.9992703199386597
- Categorical accuracy(%): 0.9656655192375183
- AUC(%): 0.9949628114700317
- AUC: 0.9962550401687622
- Mean absolute error(%): 0.0008603263413533568
- Mean absolute error: 0.0006547034136019647
- Poisson(%): 0.012651807628571987
- Poisson: 0.012182093225419521

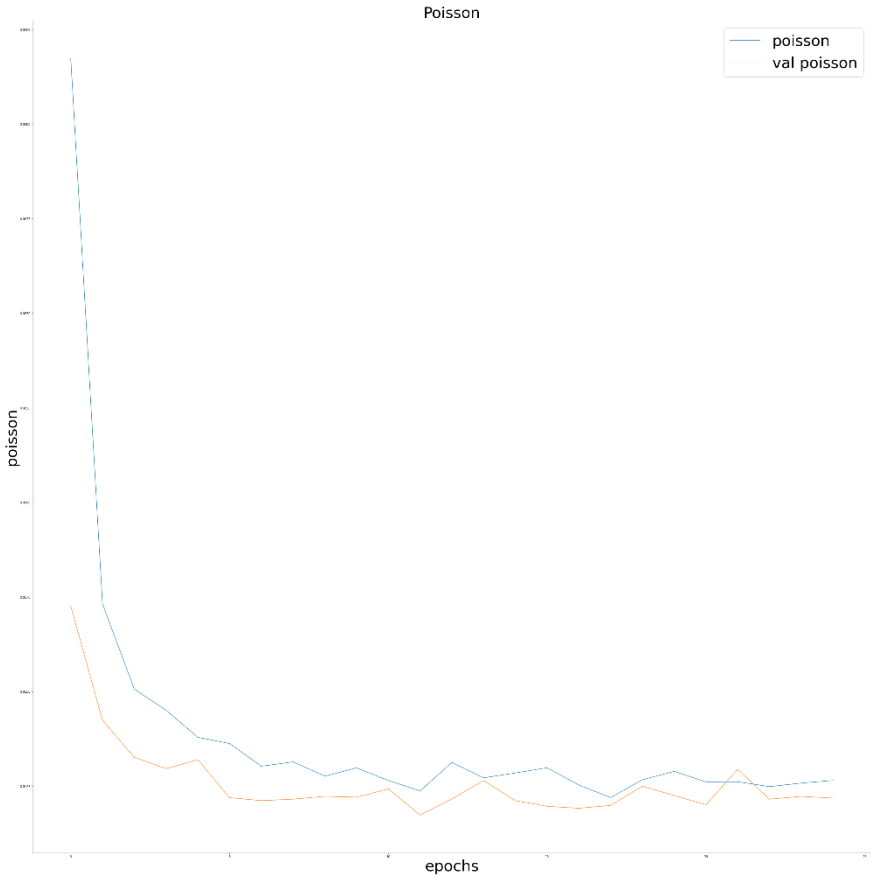
Z analizowanej architektury uzyskano poniższe wyniki przedstawione na Rys. 6 oraz Rys. 7 uzyskując poniższe dokładności.



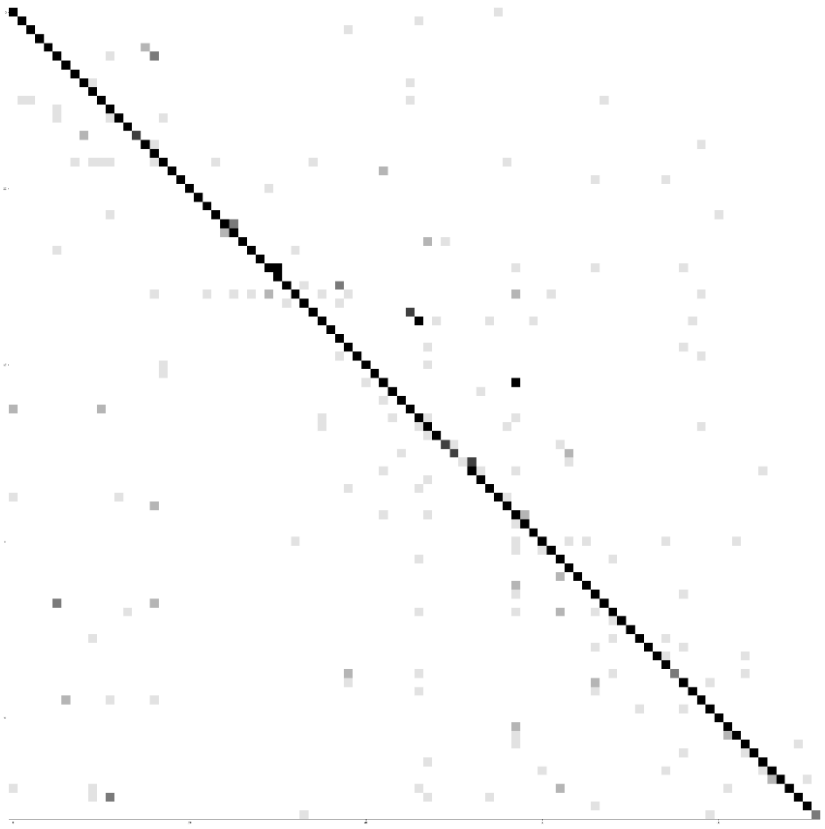
Rys 6. Analiza dokładności w modelu „2”



Rys 7. Analiza funkcji kary w modelu „2”



Rys 7. Analiza wartości funkcji Poisson w modelu „2”



Rys 7. Analiza macierzy konfuzji w modelu „2”

Porównanie uzyskanych w fazie walidacji wyników

Lp.	Loss(%)	Loss	Binary accuracy(%)	Categorical accuracy(%)	AUC(%)	AUC	Mean absolute error(%)	Mean absolute error	Poisson(%)	Poisson
1	20.060%	0.399465	99.913%	95.918%	99.414%	0.988460	0.105%	0.001341	1.296%	0.014552
2	7.455%	0.101684	99.959%	98.015%	99.832%	0.996899	0.058%	0.000358	1.167%	0.011887
3	5.160%	0.425752	99.981%	99.094%	99.860%	0.991640	0.022%	0.000653	1.136%	0.013578
4	16.994%	0.138996	99.927%	96.567%	99.496%	0.996255	0.086%	0.000655	1.265%	0.012182

Poprzez analizę wyników uzyskanych w fazie walidacji można stwierdzić, że najlepsze wyniki uzyskuje model trzeci. Model ten cechuje się najmniejszym procentowym wynikiem w metryce dokładność. Posiada również najniższa metrykę funkcji kary. Również w pozostałych metrykach cechuje się bardzo dobrym wynikiem.

Porównanie wyników z fazy testowania

Lp.	Loss(%)	Accuraccy(%)
1	92.674%	87.552%
2	21.899%	95.510%
3	75.102%	93.788%
4	25.357%	94.718%

W trakcie fazy testowania najlepszy wynik uzyskał model drugi. Model ten osiągnął dokładność na poziomie 95.5% oraz stratę wynoszącą 22%.

Dyskusja wyników oraz wnioski

Uważamy, że uzyskane wyniki są satysfakcjonujące. Specyfika rozpatrywanego przez nas problemu jest problemem skomplikowanym. Jakość używanych zdjęć różnie nie jest rewelacyjna. Liczność 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 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.