1. Cel i tytuł projektu: Przetwarzanie chińskich znaków drogowych w celu ich późniejszej klasyfikacji.

- Model referencyjny: prosta sieć neuronowa z biblioteki Keras
- Model docelowy: sieć konwolucyjna CNN poprzedzająca sieć gęstą
- Strojenie hiperparametrów: Random search z keras tuner
- Regularyzacja: Dropout, wczesne zatrzymanie, augmentacja, wykorzystanie większego podzbioru danych
- Ewaluacja modelu: Precyzyjność (ang. Accuracy)
- Funkcja kosztu: Sparse Categorical Crossentropy
- Funkcje aktywacji: Relu w warstwach ukrytych i Softmax w warstwie wyjściowej
- Optymalizer: RMSProp
- Batch size: 512
- Ilość epok uczenia modelu: 100

2. Pozyskanie danych:

kaggle

Chinese Traffic Signs

5998 annotated traffic sign images of 58 categories

Usability ⓘ

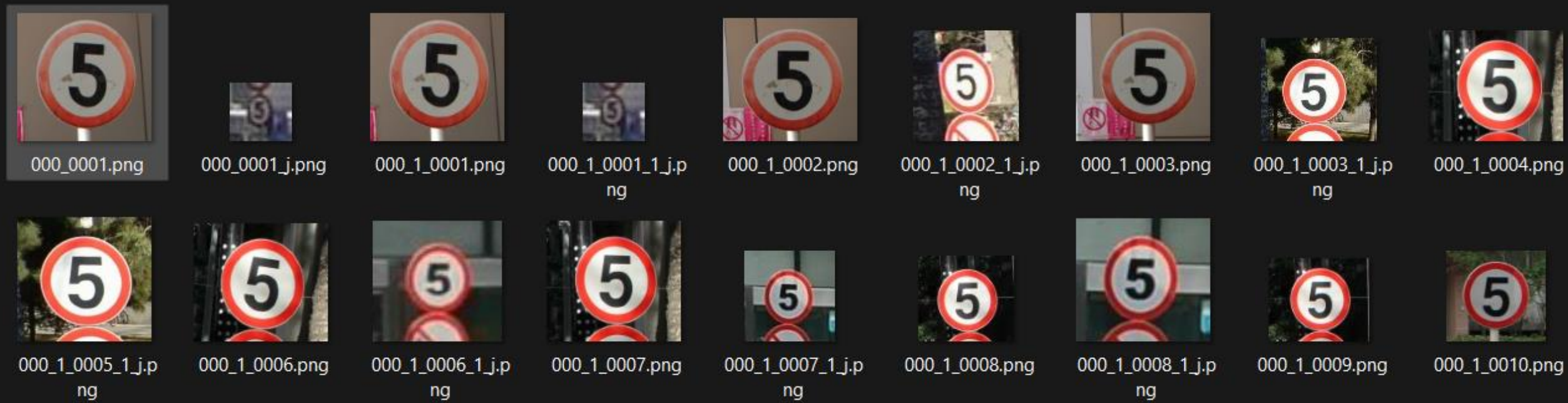
9.41

License

CC0: Public Domain

	file_name	width	height	x1	y1	x2	y2	category
0	000_0001.png	134	128	19	7	120	117	0
1	000_0002.png	165	151	23	12	149	138	0
2	000_0003.png	128	122	22	14	116	105	0
3	000_0010.png	80	73	14	8	67	63	0
4	000_0011.png	186	174	36	15	155	157	0
...	...	...	...	...	...	...	...	...
5993	056_1_0018_1_j.png	122	94	25	20	80	79	56
5994	056_1_0019_1_j.png	224	207	39	39	188	178	56
5995	056_1_0020_1_j.png	128	115	32	30	89	79	56
5996	057_1_0001_1_j.png	100	95	21	22	74	75	57
5997	057_1_0002_1_j.png	145	136	28	25	111	114	57

5998 rows × 8 columns



3. Wstępna ocena danych: nie występowały braki, w pliku csv pojawiały się duplikaty

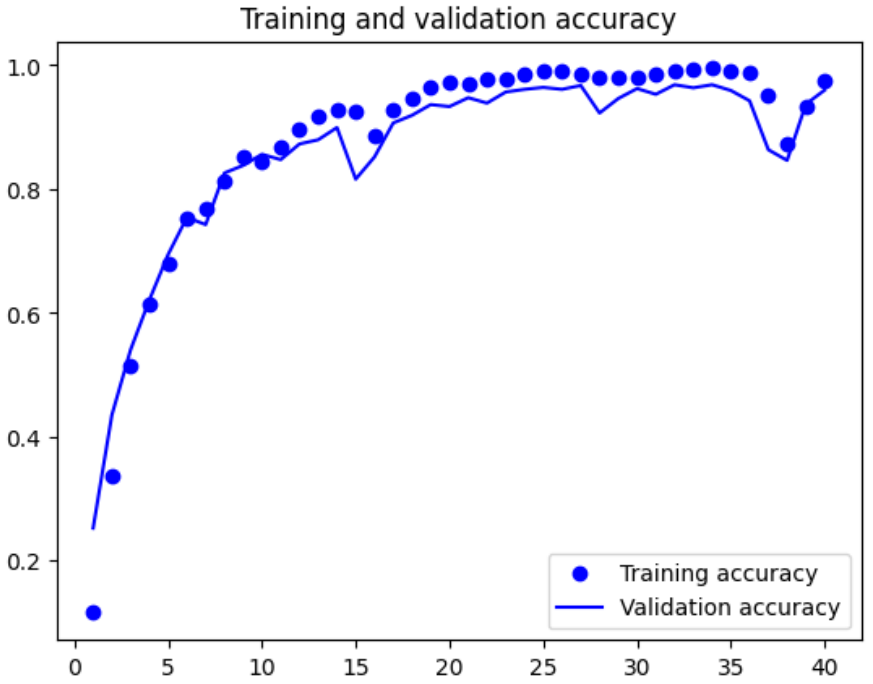
4. Przygotowanie danych: przycięcie obrazów do podanych wierzchołków skrajnych kwadratu opisanego na znaku drogowym, ujednolicenie rozmiarów obrazów do jednego rozmiaru, konwersja zdjęć RGB do skali szarości oraz przeskalowanie jej z przedziału od 0 do 255 na przedział od 0 do 1

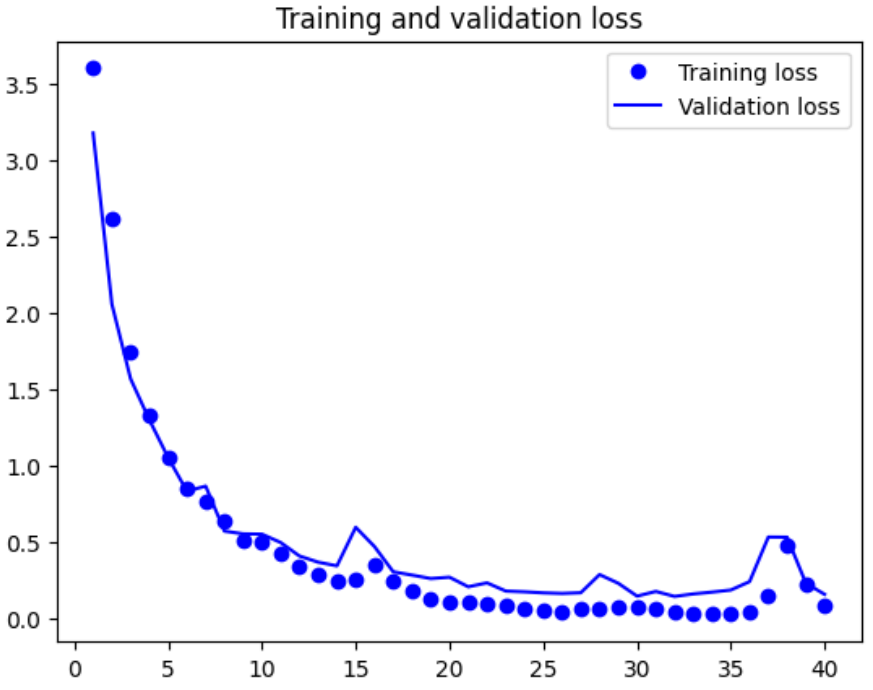
```
(3778, 40, 40, 1)
(1620, 40, 40, 1)
(600, 40, 40, 1)
(3778,)
(1620,)
(600,)
```

	images	shapes	labels
0	[[[132, 114, 98], [132, 113, 98], [131, 113, 9...	(40, 40, 3)	0
1	[[[109, 119, 123], [82, 94, 98], [68, 80, 83],...	(40, 40, 3)	0
2	[[[126, 114, 109], [139, 117, 97], [140, 115, ...	(40, 40, 3)	0
3	[[[35, 37, 44], [45, 45, 51], [54, 54, 60], [5...	(40, 40, 3)	0
4	[[[170, 168, 169], [170, 168, 170], [169, 168,...	(40, 40, 3)	0

5. Modelowanie: model referencyjny – prosta sieć gęsta złożona z sześciu warstw Full Connected

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 1600)	0
dense (Dense)	(None, 64)	102,464
dense_1 (Dense)	(None, 128)	8,320
dense_2 (Dense)	(None, 256)	33,024
dense_3 (Dense)	(None, 512)	131,584
dense_4 (Dense)	(None, 1024)	525,312
dense_5 (Dense)	(None, 58)	59,450

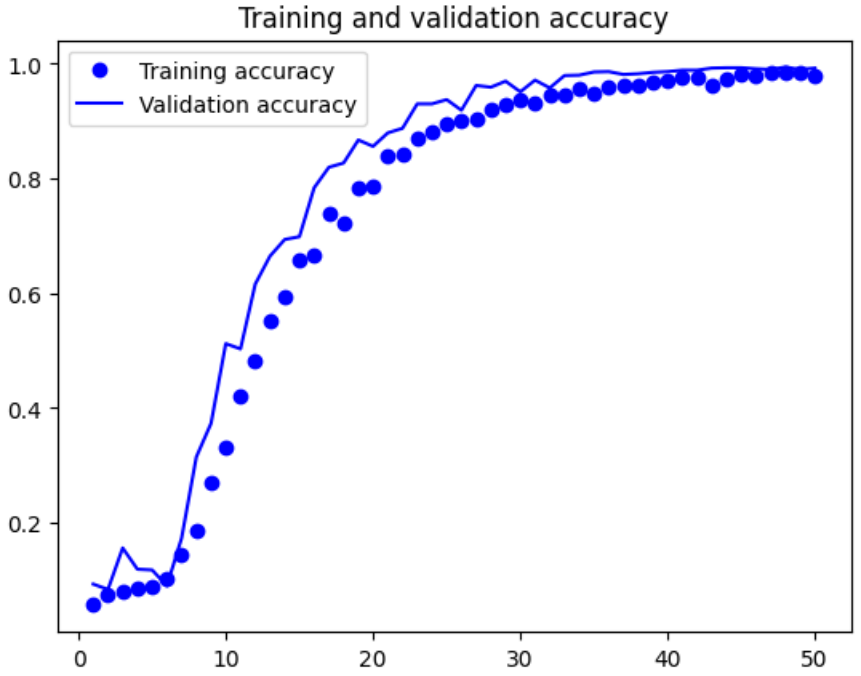


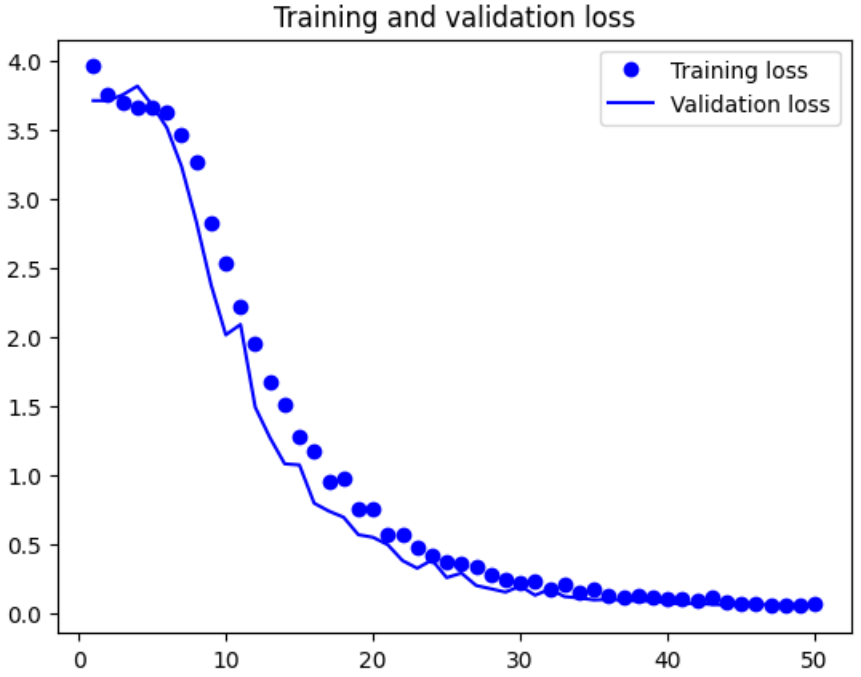


Sebastian Szczepański S29805
Warszawa, 29.06.2024r.

5. Modelowanie: model docelowy – sieć konwolucyjna z warstwamy Full Connected ma końcu architektury (dobrany metodą prób i błędów)

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 40, 40, 1)	0
conv2d (Conv2D)	(None, 38, 38, 64)	640
max_pooling2d (MaxPooling2D)	(None, 19, 19, 64)	0
dropout (Dropout)	(None, 19, 19, 64)	0
conv2d_1 (Conv2D)	(None, 17, 17, 128)	73,856
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 128)	0
dropout_1 (Dropout)	(None, 8, 8, 128)	0
conv2d_2 (Conv2D)	(None, 6, 6, 256)	295,168
max_pooling2d_2 (MaxPooling2D)	(None, 3, 3, 256)	0
dropout_2 (Dropout)	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
dense_6 (Dense)	(None, 256)	590,080
dropout_3 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 512)	131,584
dropout_4 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 58)	29,754





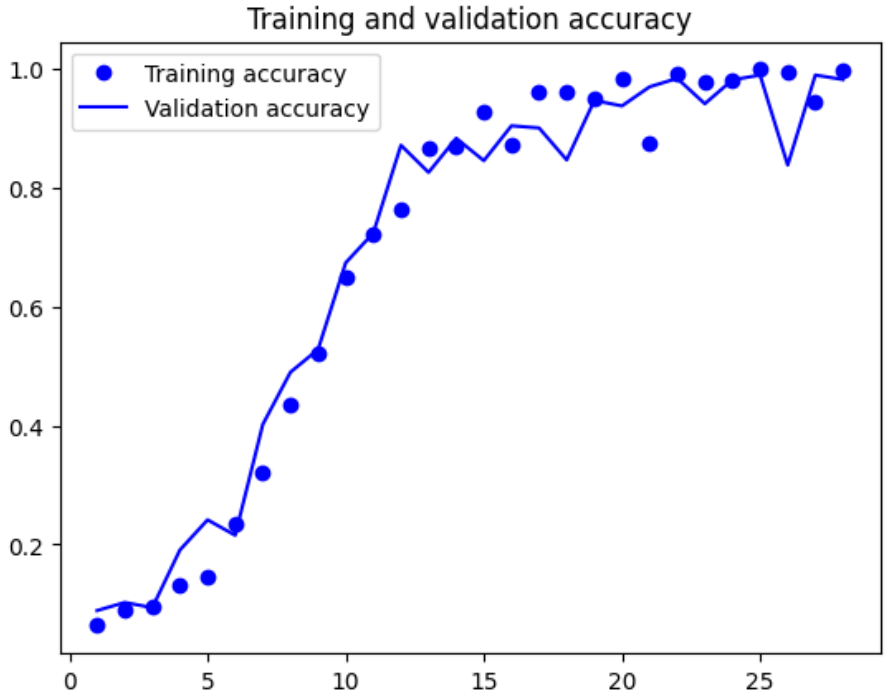
Sebastian Szczepański S29805
Warszawa, 29.06.2024r.

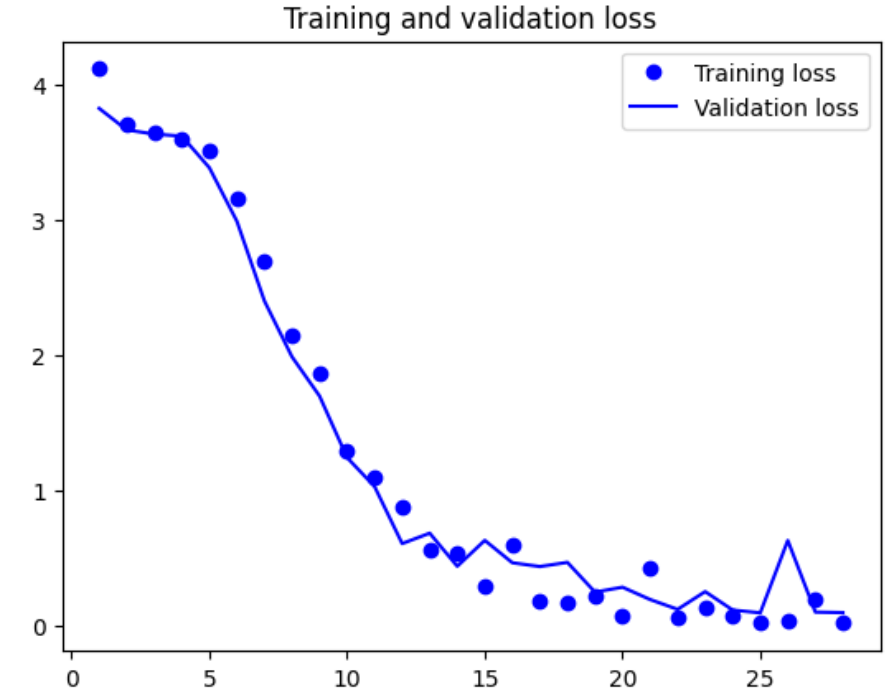
5. Modelowanie: model dobrany za pomocą keras_tuner.RandomSearch() złożony z sieci konwolucyjnej i gęstej

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 38, 38, 256)	2,560
max_pooling2d (MaxPooling2D)	(None, 19, 19, 256)	0
conv2d_1 (Conv2D)	(None, 17, 17, 256)	590,080
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 256)	0
conv2d_2 (Conv2D)	(None, 6, 6, 256)	590,080
max_pooling2d_2 (MaxPooling2D)	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 512)	1,180,160
dense_1 (Dense)	(None, 512)	262,656
dense_2 (Dense)	(None, 58)	29,754

```
Search space summary
Default search space size: 3
filters (Choice)
{'default': 32, 'conditions': [], 'values': [32, 256, 512], 'ordered': True}
dropout (Boolean)
{'default': False, 'conditions': []}
units (Choice)
{'default': 128, 'conditions': [], 'values': [128, 256, 512], 'ordered': True}
```

```
tuner = keras_tuner.RandomSearch(
    build_model,
    objective='val_loss',
    max_trials=5,
    overwrite = True)
```





6. Ewaluacja modeli:

	Model referencyjny	Model docelowy	Model dobrany za pomocą keras tuner
Wynik testowy (Precyzyjność)	97,8 %	99,7 %	99,0 %

7. Wdrożenie i wnioski:

- model docelowy okazał się lepszy od modelu referencyjnego ze względu na wykorzystanie warstw sieci konwolucyjnych i warst Dropout, które umożliwiły naukę modelu przez 100 epok bez przetrenowania,
- do strojenia docelowego modelu za pomocą keras tuner użyto propozycji słabszych modeli z siecią konwolucyjna w celu zniwelowania złożoności czasowej ewaluacji wielu modeli, co mogło wpłynąć na słabszy wynik i słabszą regularyzację, która zatrzymała uczenie najlepszego modelu po 28 epoce,
- przy tak wysokich wynikach można byłoby spróbować stworzyć modele na danych bez usuniętego tła w celu stworzenia odporniejszych modeli na szum,