Mateusz Erezman 171675
Michał Hajdasz 172156
Dariusz Kobiela 175656

TASK 3
Classification of types of ground vehicles

Advanced data preparation in machine learning

1. Basing on the problem you solve and the data you have, discuss and decide:
What is the type of the problem (classification, regression, detection, generation, etc.)
What kind of training methods you can use (supervised training, unsuperviced, etc.)
What are possible machine learning methods and models to solve your problem?
Choose a machine learning method, model and architecture suitable to your problem.
What are possible loss / fitness functions for your problem? Choose one of them and justify why. Knowing which machine learning methods you use, describe possible testing metrics and procedures (required metrics: Loss, Accuracy, F-score, ROC, Confusion matrix) which are suitable to your problem.

Because our problem is image data classification, we will use the transfer learning method and adapt MobileNet V2 developed by Google to our problem. Transfer learning enables us to use a pre-trained model (with some data used by creators), and to fine-tune the model to our problem by training it again with data collected by us. It is a supervised learning problem.
[https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c]
[https://www.tensorflow.org/tutorials/images/transfer_learning]

Because our problem is multi-class classification (labels from 0 to 4), possible loss functions are **Multi-Class Cross-Entropy Loss** (which is the default loss function to use for multi-class classification problems), **Sparse Multiclass Cross-Entropy Loss** (used to problems with a large number of labels, which require one hot encoding process; it is performing the same cross-entropy calculation of error, without requiring that the target variable be one hot encoded prior to training) and **Kullback Leibler Divergence Loss** (most commonly used when using models that learn to approximate a more

complex function than simply multi-class classification, such as in the case of an autoencoder used for learning a dense feature representation under a model that must reconstruct the original input. In this case, KL divergence loss would be preferred. Nevertheless, it can be used for multi-class classification, in which case it is functionally equivalent to multi-class cross-entropy). [https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/]

Implementation details. In keras library, the implementation of Sparse Multiclass Cross-Entropy Loss is called SparseCategoricalCrossentropy.

```
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
```

Chosen metrics to evaluate the model:

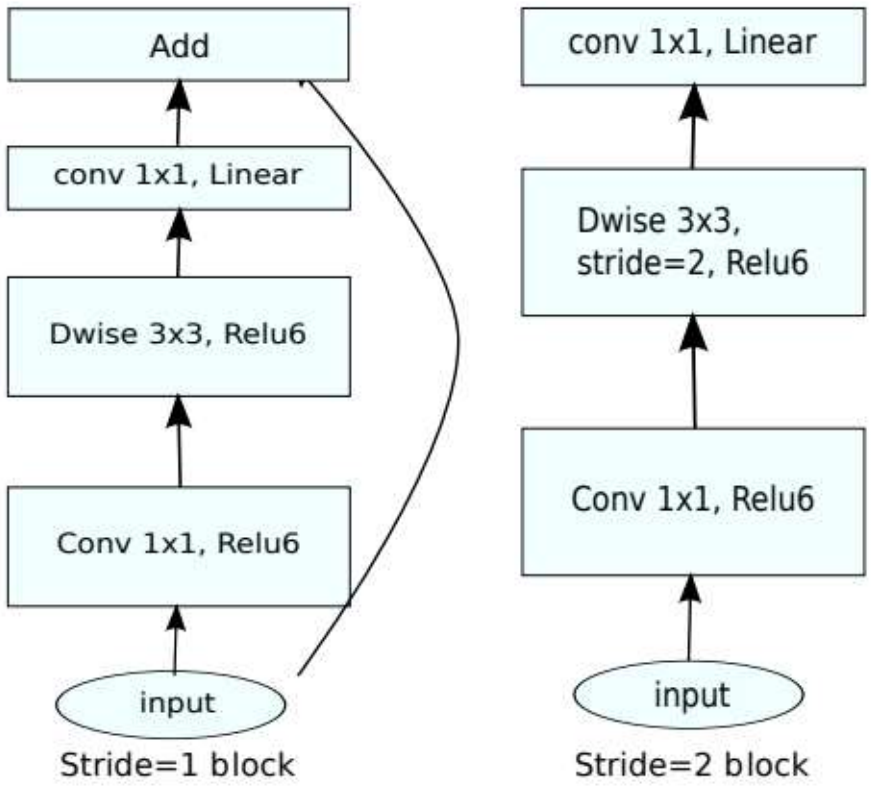
- **Loss: Sparse Multi-Class Cross-Entropy Loss**. Cross-entropy is the default loss function to use for multi-class classification problems. In this case, it is intended for use with multi-class classification where the target values are in the set {0, 1, 3, …, n}, where each class is assigned a unique integer value.
- **Accuracy**.
- **F-score**.
- **ROC**
- **Confusion matrix**.

2. Training. Perform training and validation of the chosen model using their corresponding datasets. Calculate and plot all metrics for the whole training (i.e. after every epoch of training). Training should run for at least 10 epochs.
Relying on validation results, choose and save few 'best' models as MODELS_BASELINE

The MobileNetv2 architecture consists of about 53 deep layers, with additional normalization layers (154 layers at all). Its architecture can be described as:

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

where t: expansion factor, c: number of output channels, n: repeating number, s: stride. 3×3 kernels are used for spatial convolution.



(d) Mobilenet V2

At the top of the layers, our own Dense layer with 5 neurons (because of the fact of having 5 classes multi-class classification problem) is added.

The architecture consists of 2,257,984 total params, which are initially frozen.

Model: "mobilenetv2_1.00_224"

_____ _____ Layer (type) Output Shape Param # Connected to

==========

input_1 (InputLayer) [(None, 224, 224, 3)] 0 []
Conv1 (Conv2D) (None, 112, 112, 32) 864 ['input_1[0][0']
bn_Conv1 (BatchNormalization) (None, 112, 112, 32) 128
['Conv1[0][0'] Conv1_relu (ReLU) (None, 112, 112, 32) 0
['bn_Conv1[0][0']

…. (more and more layers)

Conv_1 (Conv2D) (None, 7, 7, 1280) 409600 ['block_16
_project_BN[0][0'] **Conv_1_bn (BatchNormalization) (None, 7, 7, 1280)**
5120 ['Conv_1[0][0'] **out_relu (ReLU) (None, 7, 7, 1280) 0**
['Conv_1_bn[0][0']
==========
======Total params: 2,257,984
Trainable params: 0
Non-trainable params: 2,257,984

The input size (resolution) of the image can be chosen from a selected range values from 96 to 224. Because of the fact that our source images have a resolution of 256x256, we have decided to scale them into 224x224 size (the highest possible). Also, in the loss function parameter from_logits = True, so what comes out of the network is called logits - values that are an argument of a logistic function and this is where the "probability" comes from.

The final architecture with the added layer consists of 6405 trainable params (only last layer is unfrozen to be trained in order not to update weights in the basic MObileNetv2 architecture):

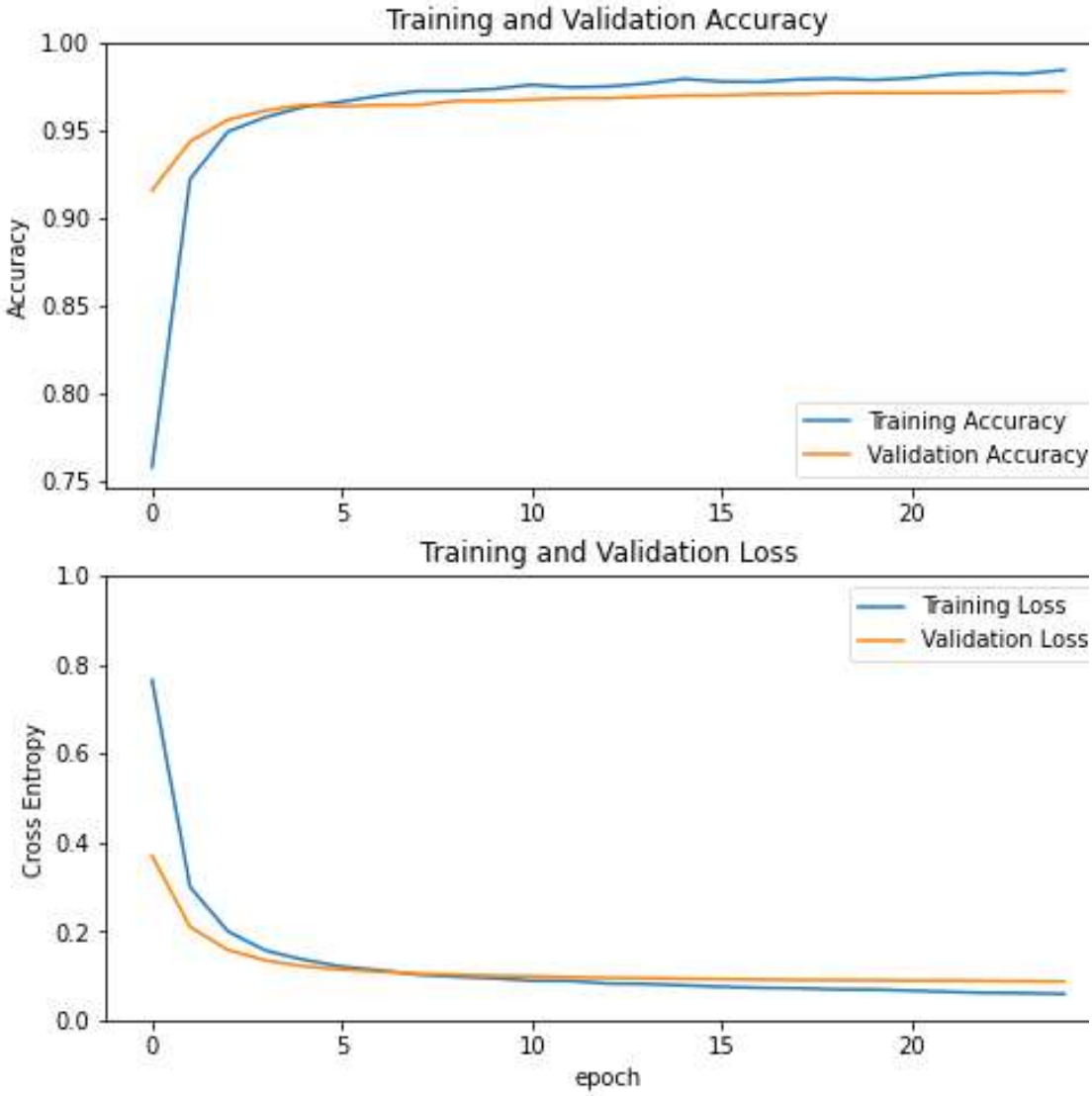
Model: "MODEL1_BASELINE"
_____
Layer (type) Output Shape Param #
==========
======
input_2 (InputLayer) [(None, 224, 224, 3)] 0
tf.math.truediv (TFOpLambda) (None, 224, 224, 3) 0
tf.math.subtract (TFOpLambda) (None, 224, 224, 3) 0
mobilenetv2_1.00_224 (Functional) (None, 7, 7, 1280) 2257984
global_average_pooling2d (GlobalAveragePooling2D) (None, 1280)

0
dropout (Dropout) (None, 1280) 0
dense (Dense) (None, 5) 6405
===
Total params: 2,264,389
Trainable params: 6,405
Non-trainable params: 2,257,984
_____

3. Training on SPLIT1 datasets
Perform the training on SPLIT1/TRAIN dataset and simple testing on SPLIT1/VAL dataset (the datasets should be created in "Task 2" of the lab).
Observe and present changes in loss/fitness during training. Compare results on TRAIN and VAL datasets.
Try to overfit you model. Describe this process and results.
Choose the best model (according to validation results), save it as MODEL1.

Sample images for classification:

MODEL: MODEL1_BASELINE
TRAIN: Found 6474 files belonging to 5 classes.
VAL: Found 1294 files belonging to 5 classes.
Number of train batches: 203
Number of validation batches: 41
loss after training: 0.0869
accuracy after training: 0.9722




```
images shape: (224, 224, 3)
IMG_SHAPE:  (224, 224, 3)
example batch size (32, 7, 7, 1280)
41/41 [==============================] - 228s 2s/step - loss:
1.8018 - accuracy: 0.2304
initial loss: 1.80
initial accuracy: 0.23



Epoch 1/25
203/203 [==============================] - 927s 2s/step - loss:
0.6184 - accuracy: 0.7974 - val_loss: 0.2497 - val_accuracy:
0.9334
Epoch 2/25
203/203 [==============================] - 47s 90ms/step - loss:
```

```
0.2204 - accuracy: 0.9387 - val_loss: 0.1672 - val_accuracy:
0.9528
Epoch 3/25
203/203 [==============================] - 47s 91ms/step - loss:
0.1647 - accuracy: 0.9518 - val_loss: 0.1368 - val_accuracy:
0.9595
Epoch 4/25
203/203 [==============================] - 47s 90ms/step - loss:
0.1389 - accuracy: 0.9583 - val_loss: 0.1193 - val_accuracy:
0.9641
Epoch 5/25
203/203 [==============================] - 47s 90ms/step - loss:
0.1214 - accuracy: 0.9633 - val_loss: 0.1082 - val_accuracy:
0.9677
Epoch 6/25
203/203 [==============================] - 47s 90ms/step - loss:
0.1116 - accuracy: 0.9645 - val_loss: 0.0997 - val_accuracy:
0.9702
Epoch 7/25
203/203 [==============================] - 47s 90ms/step - loss:
0.1033 - accuracy: 0.9681 - val_loss: 0.0930 - val_accuracy:
0.9720
```

Training was set to 25 epochs with an EarlyStopping parameter set to 3 epochs. Initial values of loss and accuracy were, respectively, 1.8 and 0.23. Because of the fact that we have used the Transfer Learning method, the Neural Network architecture was already pre-trained on images consisting of 1000 classes, which also contained the 5 classes defined by us: ['bicycle' 'bus' 'car' 'motorcycle' 'truck']. The ImageNetv2 architecture is so broad, that is was even more detailed in the class definition (eg. distinguishing types of buses and trucks).

Learning rate was set to 0.001 (small learning rate) in order not to overfit the model. While using the Transfer Learning method, usually there is a need to set a very small learning rate. Batch size parameter was set to 32. Monitored metric was accuracy.

Validation and training loss converges quickly. That is a good sign - the chosen learning rate was neither too small nor too big. Only the first epoch learning took about 15 minutes (because of the fact that the model started updating weights). The next epochs' iterations took about 50 seconds each. It was possible to train the model longer than 25 epochs without overfitting, but we have decided to stop here as the BASELINE_MODEL.

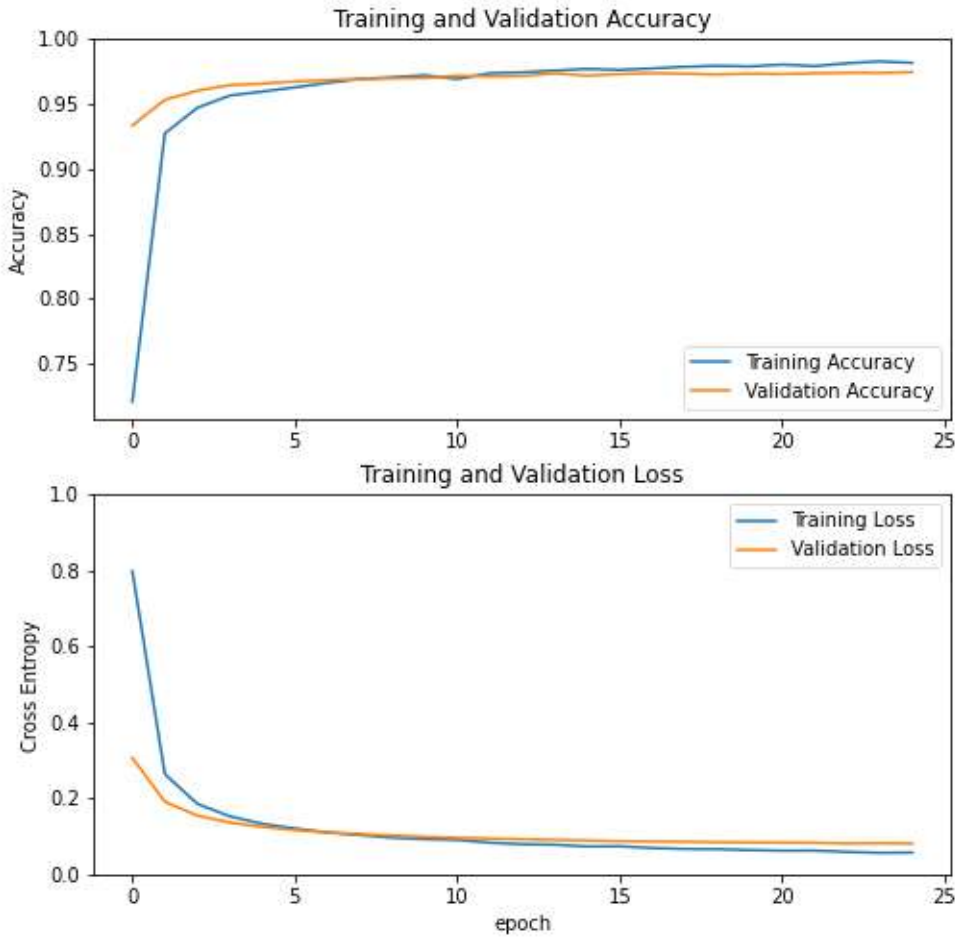
In order to overfit the model, the number of epochs can be simply set to a big number (like 800 hundred) without the EarlyStopping parameter. Then, the

achieved results will be high only on the training dataset, and low on the validation of test sets. 4. Training on SPLIT2 datasets *[10 points]*
Perform the training on SPLIT2/TRAIN dataset and simple testing on SPLIT2/VAL dataset.
Compare results (time, accuracy, etc.) to the training on SPLIT1
Choose the best model (according to validation results), save it as MODEL2.

Sample examples for classification:

MODEL: MODEL2_BASELINE
Train: Found 13151 files belonging to 5 classes.

Val: Found 3286 files belonging to 5 classes.



Number of train batches: 411
Number of validation batches: 103
loss after training: 0.08169762790203094
accuracy after training: 0.9744369983673096

Loss function converges fastly and smoothly. Training loss is lower than validation loss, so the model is not overfitted. The achieved validation accuracy is higher than in MODEL1_BASELINE, because here we have more data (augmentation used) and balanced classes (the same number of

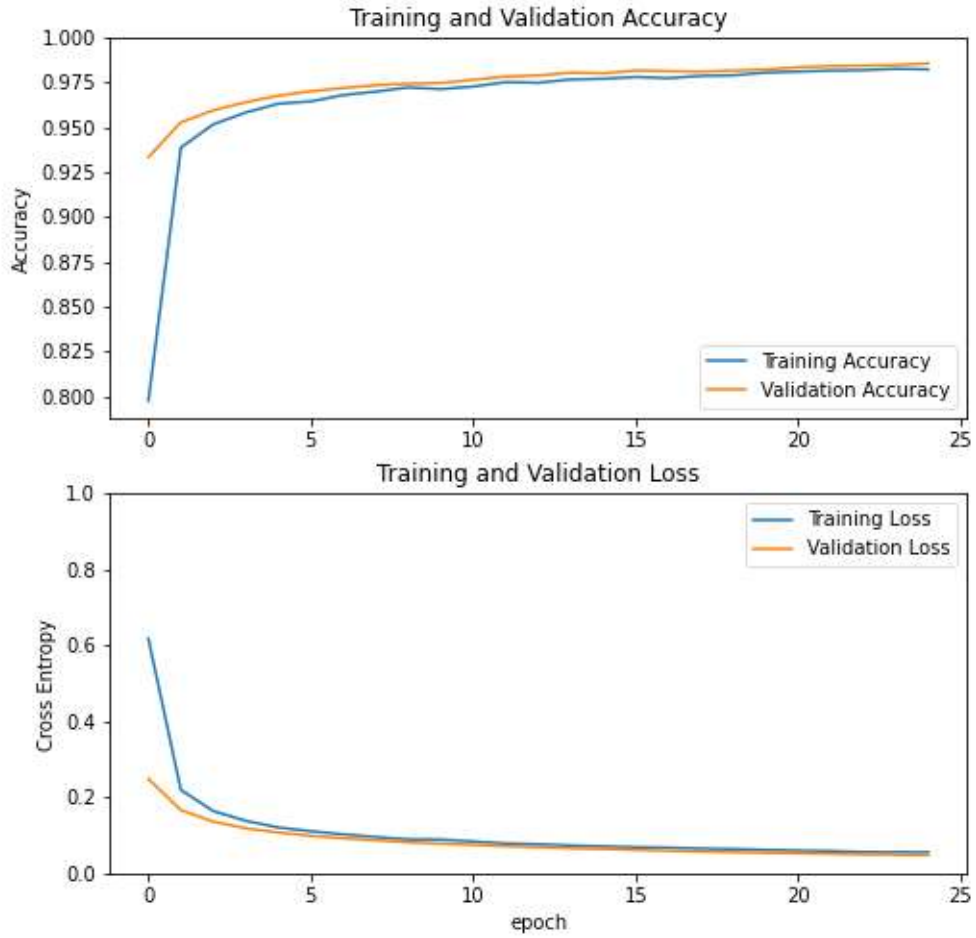
learning examples).

5. Training on SPLIT3 datasets *[5 points]*
Perform the training on SPLIT3/TRAIN dataset and simple testing on SPLIT3/VAL dataset
Compare results (time, accuracy/loss on TRAIN dataset, accuracy/loss on VAL dataset) to the training on SPLIT2
Choose the best model (according to validation results), save it as MODEL3.

NOTE. SPLIT3 shall be exactly the same as SPLIT2, with one exception: VAL set must be added to TRAIN set.

Sample examples for classification:

MODEL: MODEL3_BASELINE
Train: Found 16437 files belonging to 5 classes.
Val: Found 3286 files belonging to 5 classes.



Number of train batches: 514
Number of validation batches: 103
loss after training: 0.04837512969970703
accuracy after training: 0.9856969118118286

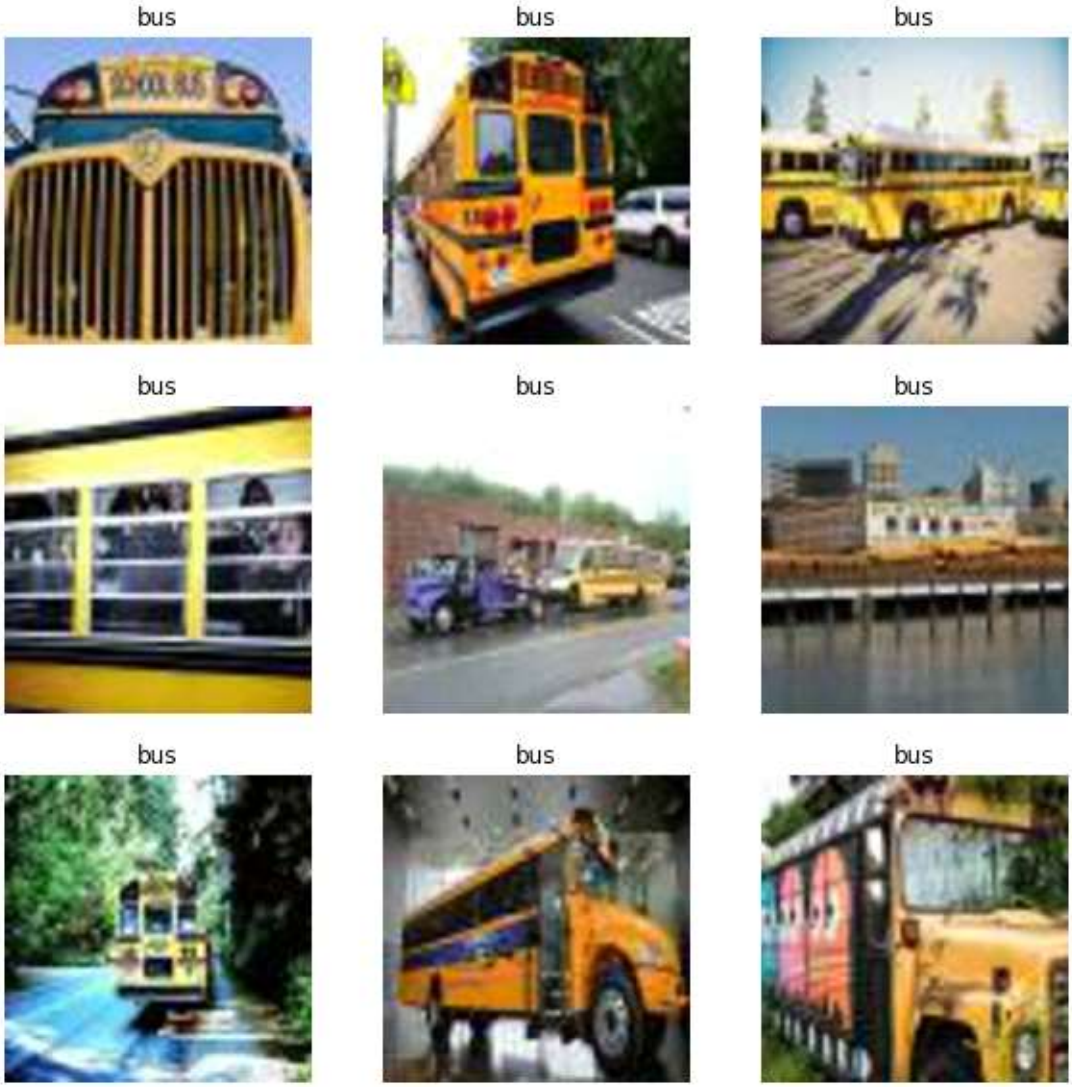
This model is exactly the same as MODEL2_BASELINE with one exception: here we have added a validation set to the train set.
Loss function converges fastly and smoothly. The achieved validation accuracy is higher than in MODEL2_BASELINE, because we have added a validation set to the train set. However, adding the validation set to the training set brings risk, that the model can overfit (learn the training examples by heart). Thus, we cannot reliably evaluate the results of the model.

6. Test your models created in Task 3 (MODEL1_BASELINE, MODEL2_BASELINE, MODEL3_BASELINE) using all defined measures. For each test, make some useful plots and result visualizations.
Test on TRAIN subsets. Test on VAL subsets. Test on TEST subsets
Discuss the results. Discussion may include: differences and similarities in results for different models and datasets, analysis of errors and their causes (according to data analysis from Task 2), differences in results for various test metrics (you shall describe what these differences mean), and others.

Sample testset1 examples:

Sample testset2 examples:


bicycle
bicycle
bicycle
bicycle
bicycle
bicycle
bicycle
bicycle
bicycle



At the beginning, data about labels and predictions is taken:

- test_dataset_labels:

```
<PrefetchDataset element_spec=(TensorSpec(shape=(None, 224, 224, 3)
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
[3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4]
[4 4 4]
```

- test_dataset_binarized_labels:

```
array([[1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0],
```

- test_dataset_predicted_probs:

```
array([[ 3.3810773 , -5.4127207 , -3.6946068 ,  1.7537885 , -5.3273964
       [ 5.0070186 , -5.17487   , -3.2289896 , -0.10815454, -3.8657198
       [ 5.0694475 , -5.0153027 , -3.6647482 , -1.490962  , -4.3484716
       [ 3.3227968 , -4.34748   , -3.1767912 ,  1.3998142 , -5.7245064
       [ 6.035137  , -5.0809507 , -3.850081  ,  0.3358262 , -6.2734246
       [ 3.5464375 , -6.4722114 , -4.8000503 , -0.1835701 , -5.3611765
       [ 5.4124613 , -4.024412  , -2.469121  , -1.2755262 , -4.5455475
       [ 5.3162746 , -3.8215678 , -2.55028   , -1.391141  , -4.66854
       [ 5.9005013 , -4.4253206 , -3.0238295 , -1.3518044 , -4.6868696
       [ 3.1337602 , -4.9183016 , -3.540956  ,  2.8302474 , -6.221245
       [ 6.324378  , -5.7644105 , -4.389067  ,  0.06812227, -5.569676
       [ 4.254759  , -4.9978604 , -2.3757915 , -3.8448176 , -5.0382586
       [ 4.44098   , -5.727446  , -2.4284787 , -3.6458142 , -6.240299
       [ 1.768094  , -5.9870033 , -3.113018  , -2.6688874 , -3.89831
```

- test_dataset_predicted_labels:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 4, 2, 4, 4, 4, 4, 2, 2, 4, 4, 4, 4, 2, 2, 2, 2, 2,
       1, 4, 4, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 2,
```

Then, the model is assessed using defined metrics.

MODEL1_BASELINE

MODEL1_BASELINE	testset1	testset2	trainset	valset
Accuracy	0,770992	0,965	0,985944	0,972179
F1 score	0,71651	0,964818	0,985893	0,971833
ROC AUC	0,96605	0,99593	0,997542	0,994443
Precision	0,836037	0,965819	0,985945	0,971881
Recall	0,770992	0,965	0,985944	0,972179
Cohens kappa	0,709277	0,95625	0,97972	0,95979

Results obtained by the model are relatively high. Accuracy on the “harder” test set (testset1) has about 77%, and is supported by f1-score equal to 72%. Area under the curve (AUC) covers 96% of the data. However, from the confusion matrix of the testset1 we can see that the model has problems with distinguishing between truck and bus, car and bus and car and truck. It may be connected with the fact that in the collected data set some examples of car, bus and trucks were distinguishable only by having windows next to the passenger seat and by the size of the vehicle.

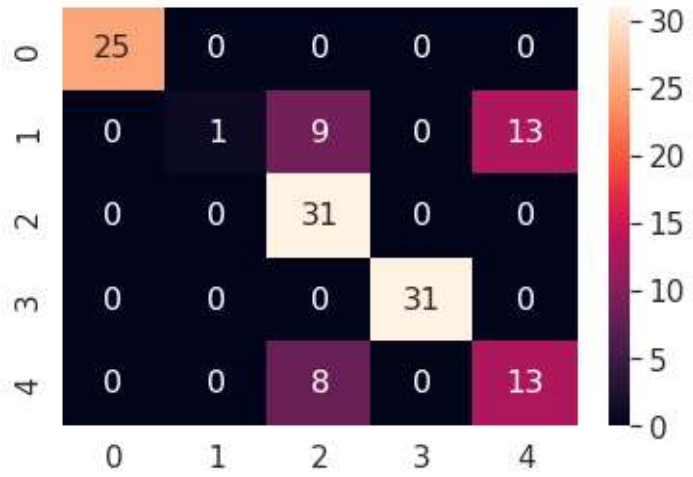
Accuracy on the “easier” test set (testset2) was 96,5%. F1-score was also about 96,5 %. Area under the curve covered almost 100% of the data. The results are very satisfactory.

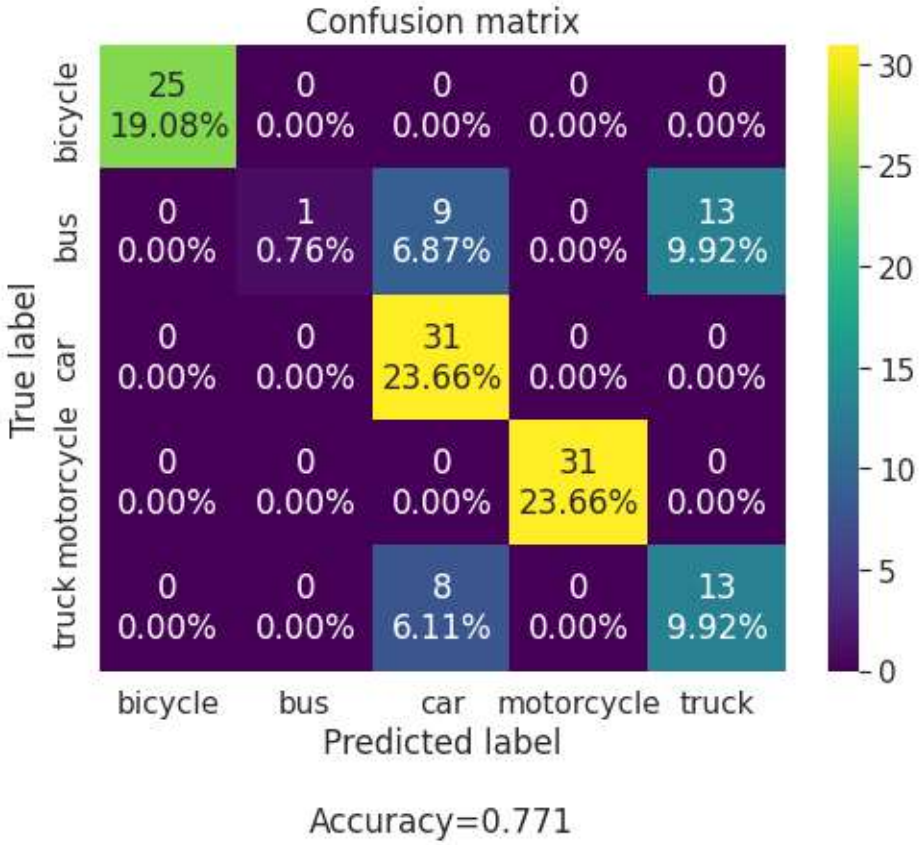
Of course, the highest results were obtained by evaluating the model using train and validation sets. Accuracy took, respectively, 98,6% and 97% (f1-score very similar). It was exactly the same data as used for training, so it was predictable. However, the model did not receive 100% accuracy, which is a good sign - it is not overfitted. The model had rather created classification rules to follow than to learn the training examples “by heart”. AUC took almost 100% of the data. Confusion matrix looks very good - small number of mistakes.

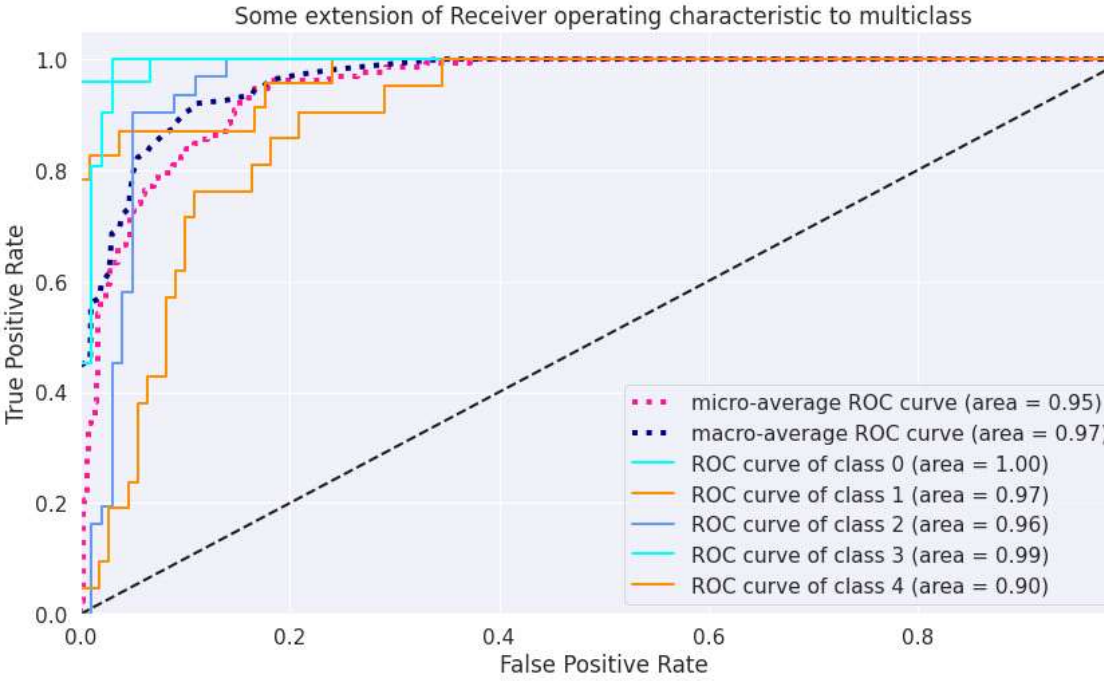
Evaluation results on the other models using training and validation dataset will be similar. We will not conduct them, because it takes a lot of time to

prepare such a big amount of data for the evaluation purposes. Only testset1 and testset2 will be taken into consideration.

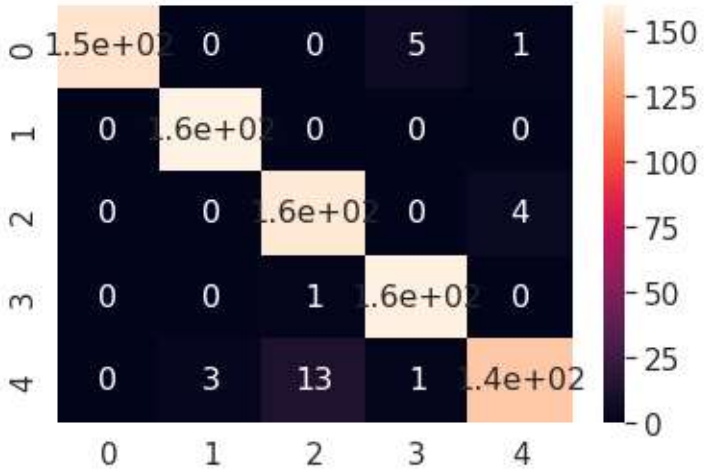
- **testset1**

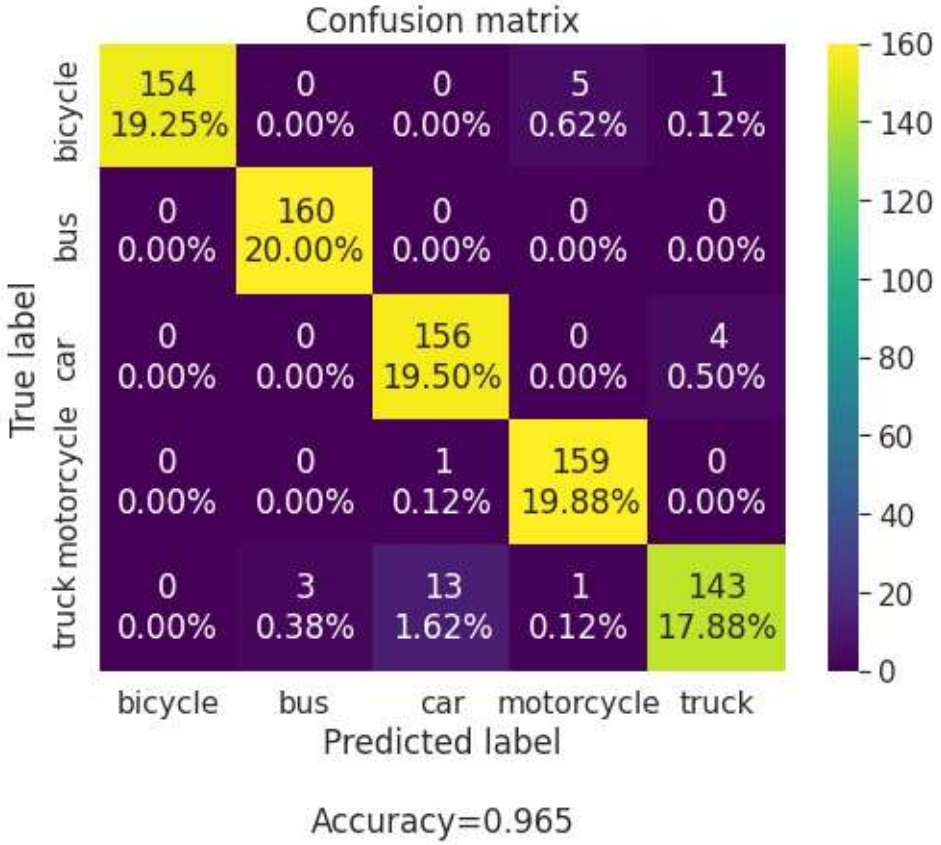



Confusion matrix
bicycle
bus
car
motorcycle
truck
True label
25
19.08%
0
0.00%
0
0.00%
0
0.00%
0
0.00%
0
0.00%
1
0.76%
9
6.87%
0
0.00%
13
9.92%
0
0.00%
0
0.00%
31
23.66%
0
0.00%
0
0.00%
0
0.00%
0
0.00%
31
23.66%
0
0.00%
0
0.00%
0
0.00%
8
6.11%
0
0.00%
13
9.92%
30
25
20
15
10
5
0
bicycle
bus
car
motorcycle
truck
Predicted label
Accuracy=0.771

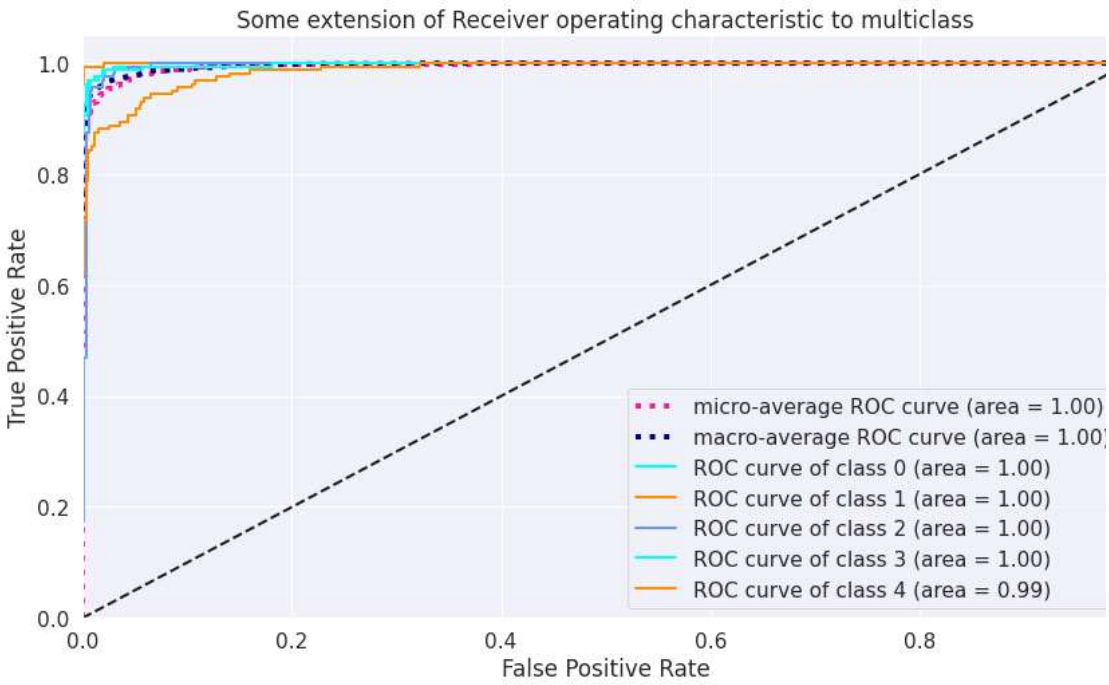

Some extension of Receiver operating characteristic to multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.95)
macro-average ROC curve (area = 0.97)
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 0.97)
ROC curve of class 2 (area = 0.96)
ROC curve of class 3 (area = 0.99)
ROC curve of class 4 (area = 0.90)

- **testset2**

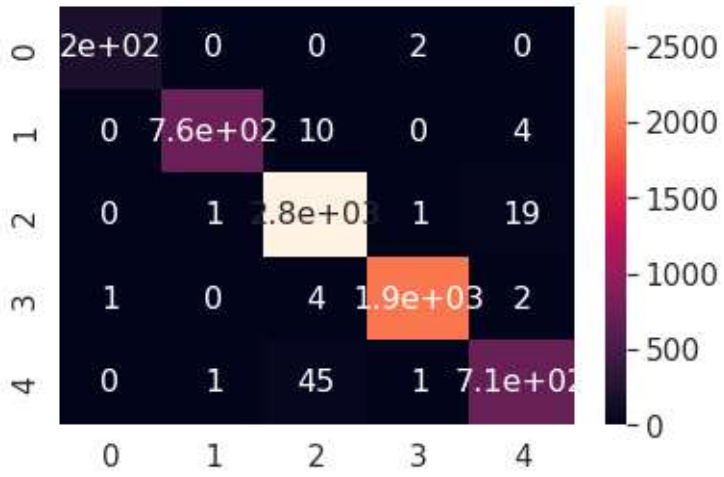


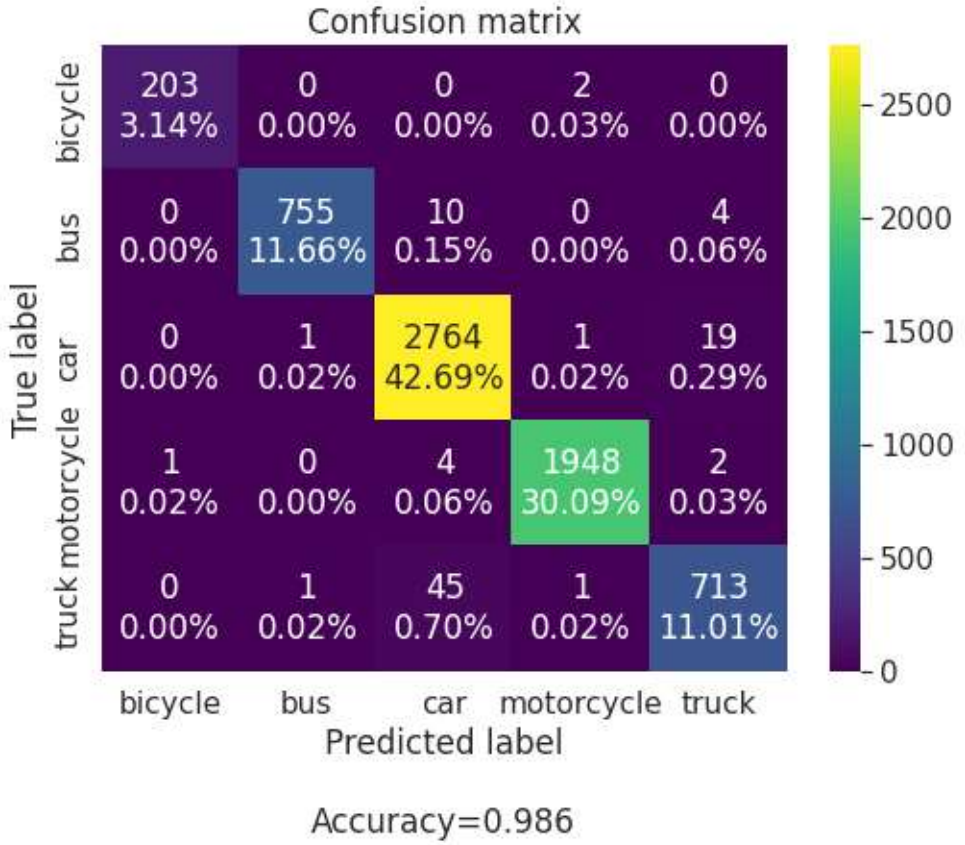


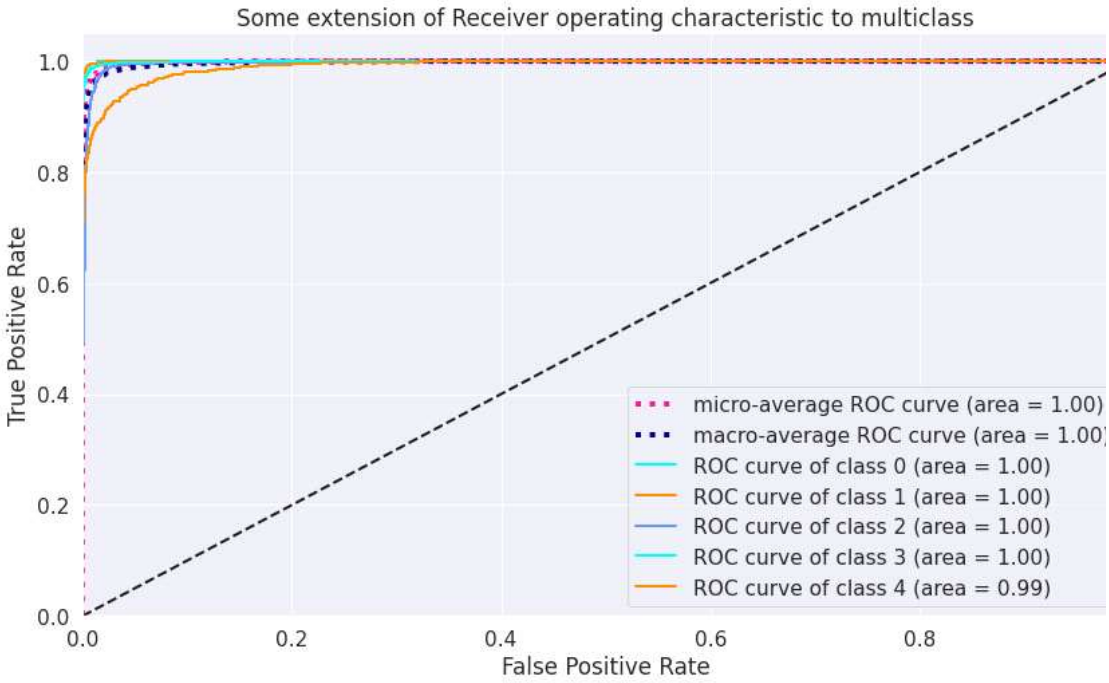
Accuracy=0.965


Some extension of Receiver operating characteristic to multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 1.00)
ROC curve of class 2 (area = 1.00)
ROC curve of class 3 (area = 1.00)
ROC curve of class 4 (area = 0.99)

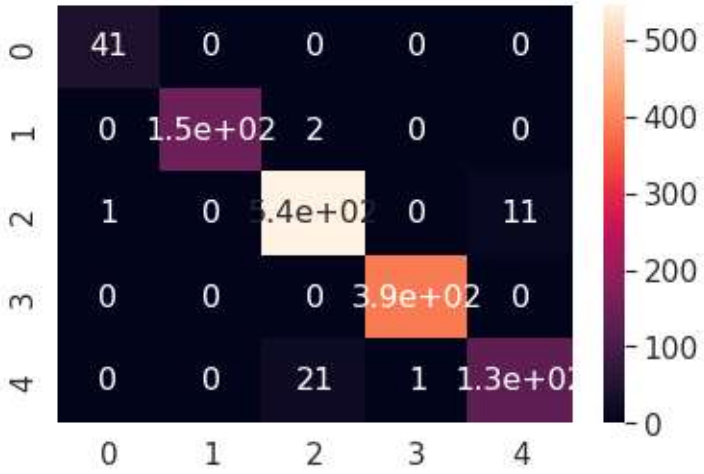
- **trainset**

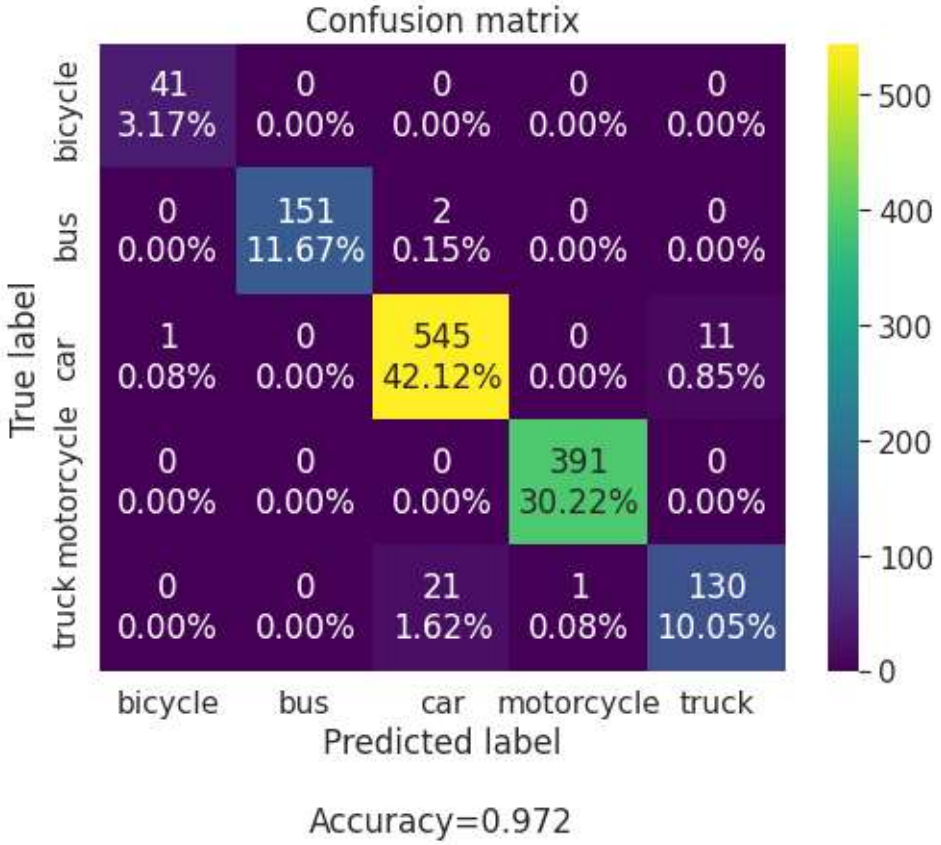

2e+02 0 0 2 0
0 7.6e+02 10 0 4
0 1 2.8e+01 1 19
1 0 4 1.9e+03 2
0 1 45 1 7.1e+02
0 1 2 3 4
2500
2000
1500
1000
500
0


Confusion matrix
True label
bicycle
bus
car
motorcycle
truck
203
3.14%
0
0.00%
0
0.00%
2
0.03%
0
0.00%
0
0.00%
755
11.66%
10
0.15%
0
0.00%
4
0.06%
0
0.00%
1
0.02%
2764
42.69%
1
0.02%
19
0.29%
1
0.02%
0
0.00%
4
0.06%
1948
30.09%
2
0.03%
0
0.00%
1
0.02%
45
0.70%
1
0.02%
713
11.01%
2500
2000
1500
1000
500
0
bicycle
bus
car
motorcycle
truck
Predicted label
Accuracy=0.986

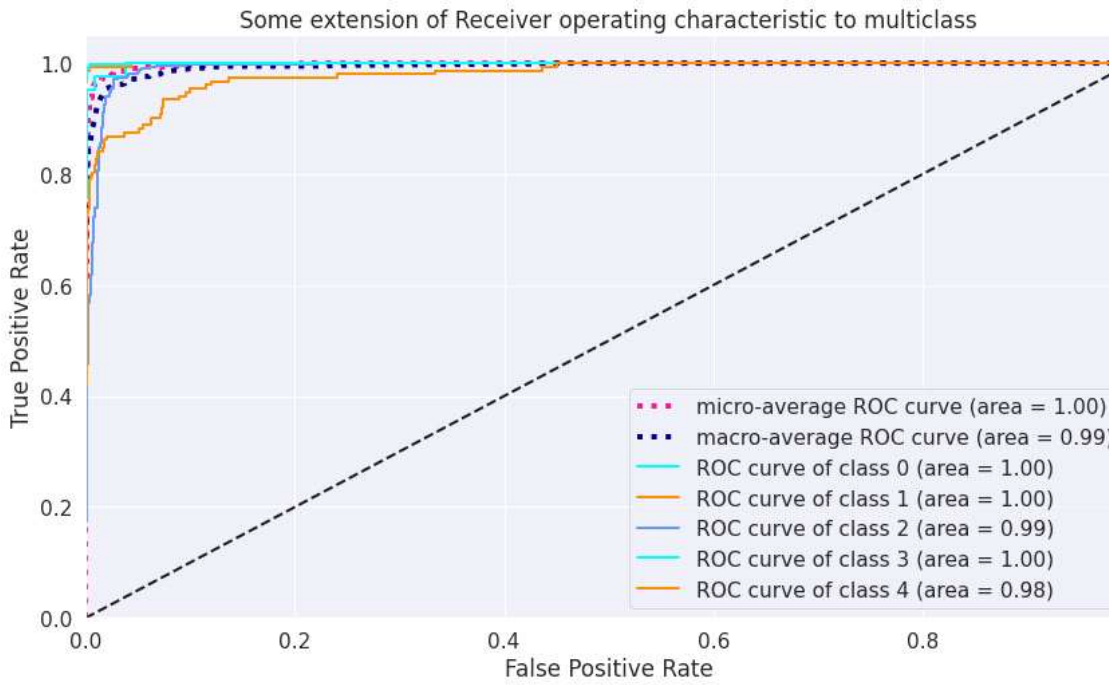

Some extension of Receiver operating characteristic to multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 1.00)
ROC curve of class 2 (area = 1.00)
ROC curve of class 3 (area = 1.00)
ROC curve of class 4 (area = 0.99)

- **validation set**





Accuracy=0.972



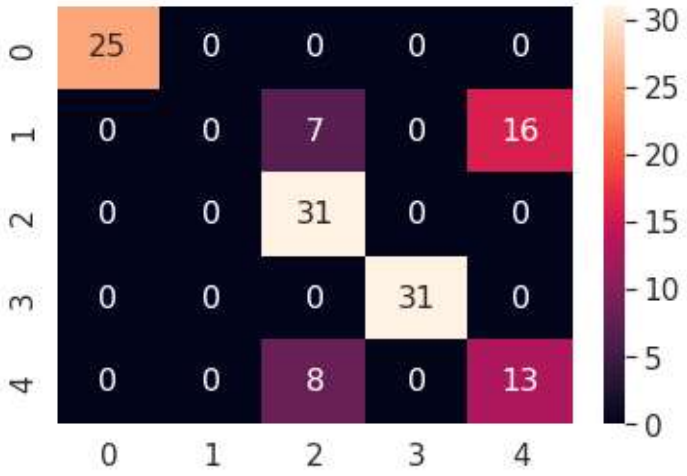
MODEL2_BASELINE

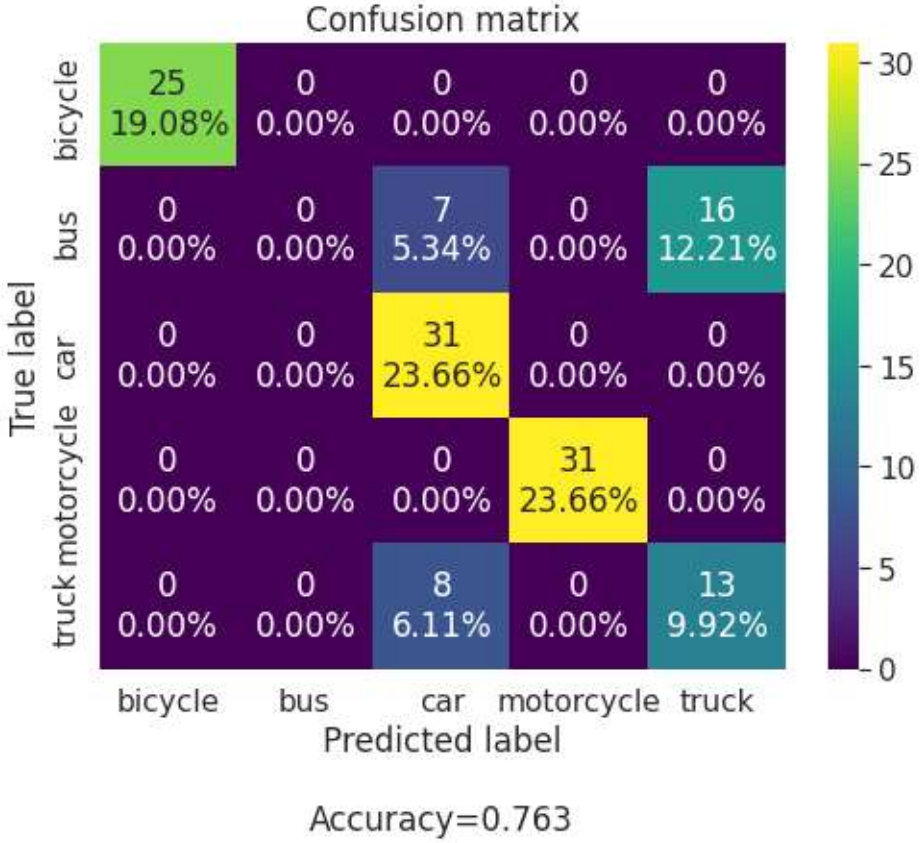
MODEL2_BASELINE	testset1	testset2
Accuracy	0,763359	0,9775
F1 score	0,701382	0,977309
ROC AUC	0,95734	0,996211
Precision	0,658818	0,977768
Recall	0,763359	0,9775
Cohens kappa	0,700074	0,971875

Once again, results obtained on the testset1 are worse than on the testset2. On the first set, accuracy is about 76%, but the f1-score is worse (only 70%). Confusion matrix shows a lot of mistakes, especially in the car-bus-truck recognition. However, this results may not be so bad, if the difference between car, bus and truck is hard to obtain even for the human eye.

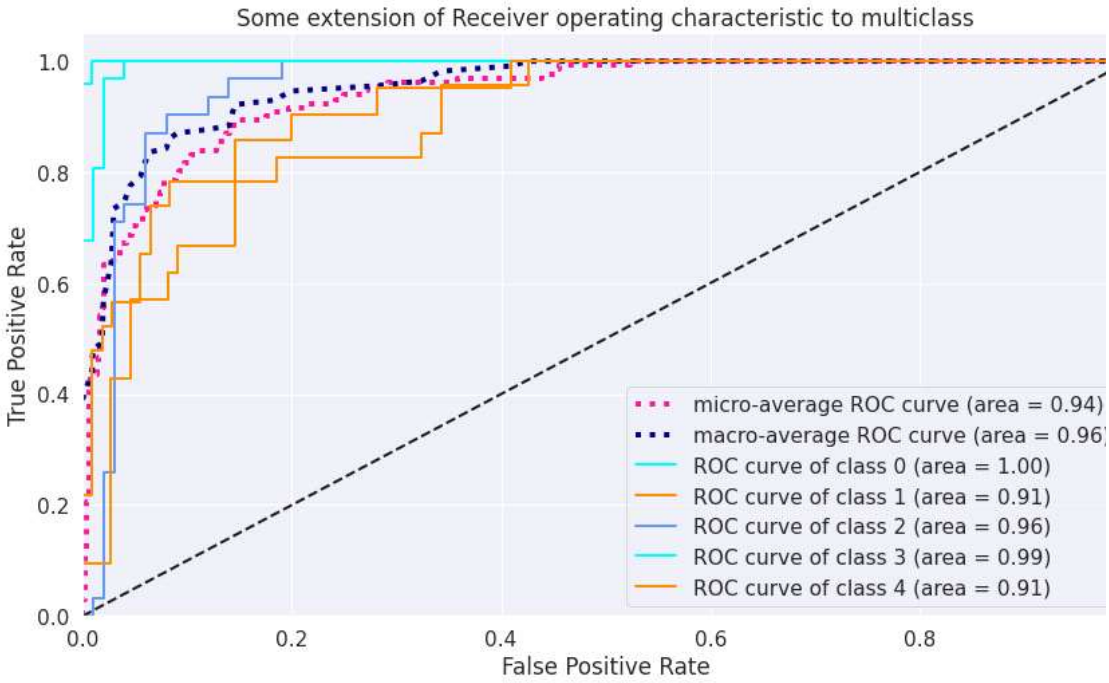
Second test set performs much better. Accuracy on the level of 97,8%, f1-score about 97,7%.

- **testset1**

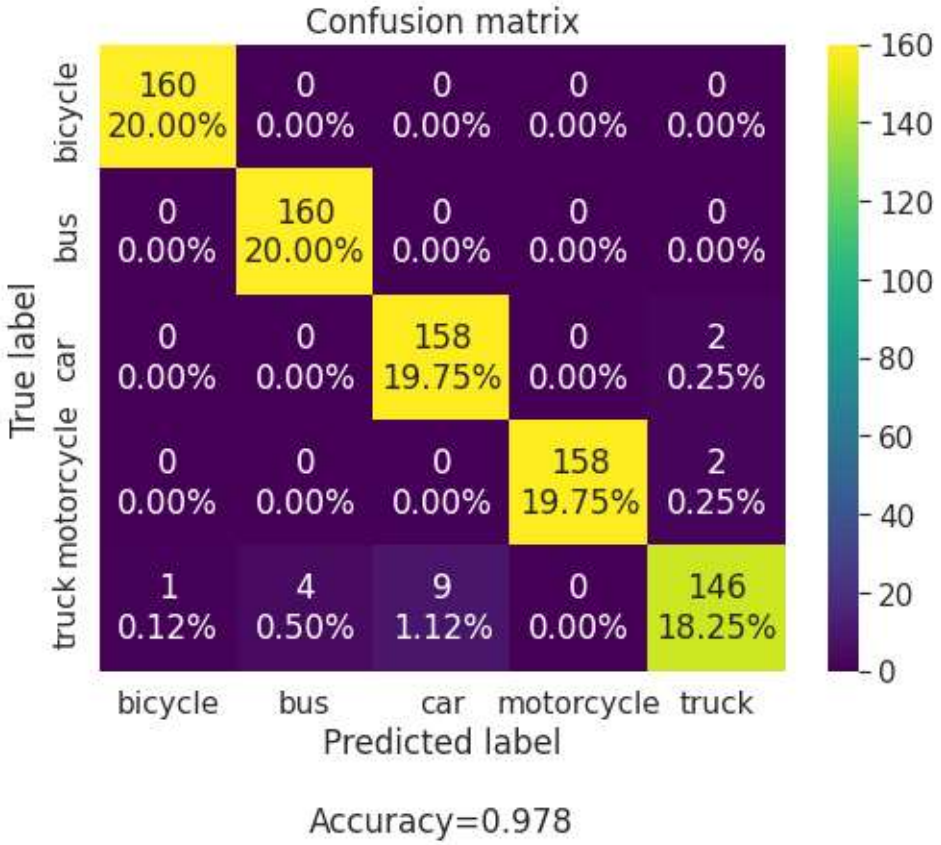


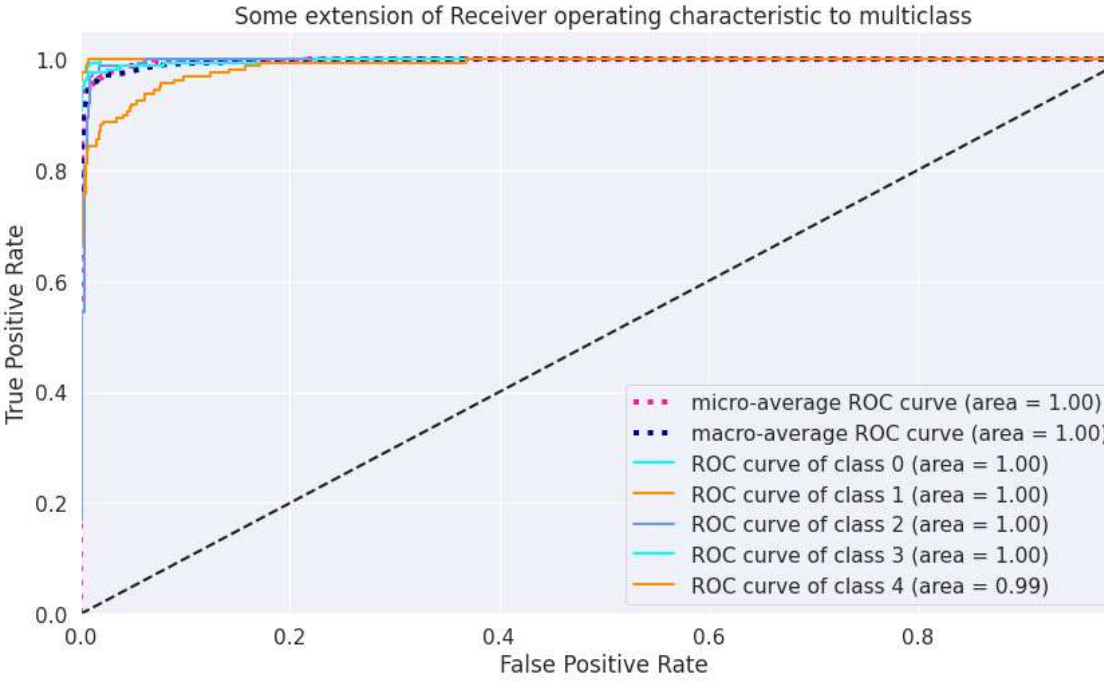


Accuracy=0.763


Some extension of Receiver operating characteristic to multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.94)
macro-average ROC curve (area = 0.96)
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 0.91)
ROC curve of class 2 (area = 0.96)
ROC curve of class 3 (area = 0.99)
ROC curve of class 4 (area = 0.91)

- **testset2**


Confusion matrix
bicycle
bus
car
motorcycle
truck
True label
160 20.00%
0 0.00%
0 0.00%
0 0.00%
0 0.00%
0 0.00%
160 20.00%
0 0.00%
0 0.00%
0 0.00%
0 0.00%
0 0.00%
158 19.75%
0 0.00%
2 0.25%
0 0.00%
0 0.00%
0 0.00%
158 19.75%
2 0.25%
1 0.12%
4 0.50%
9 1.12%
0 0.00%
146 18.25%
160
140
120
100
80
60
40
20
0
bicycle
bus
car
motorcycle
truck
Predicted label
Accuracy=0.978

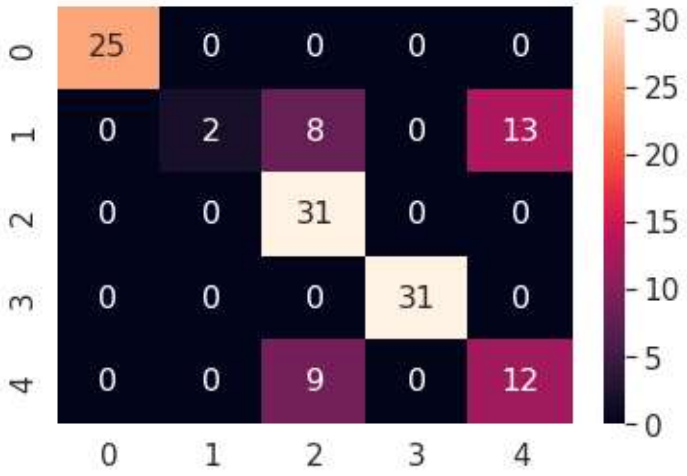

Some extension of Receiver operating characteristic to multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 1.00)
ROC curve of class 2 (area = 1.00)
ROC curve of class 3 (area = 1.00)
ROC curve of class 4 (area = 0.99)

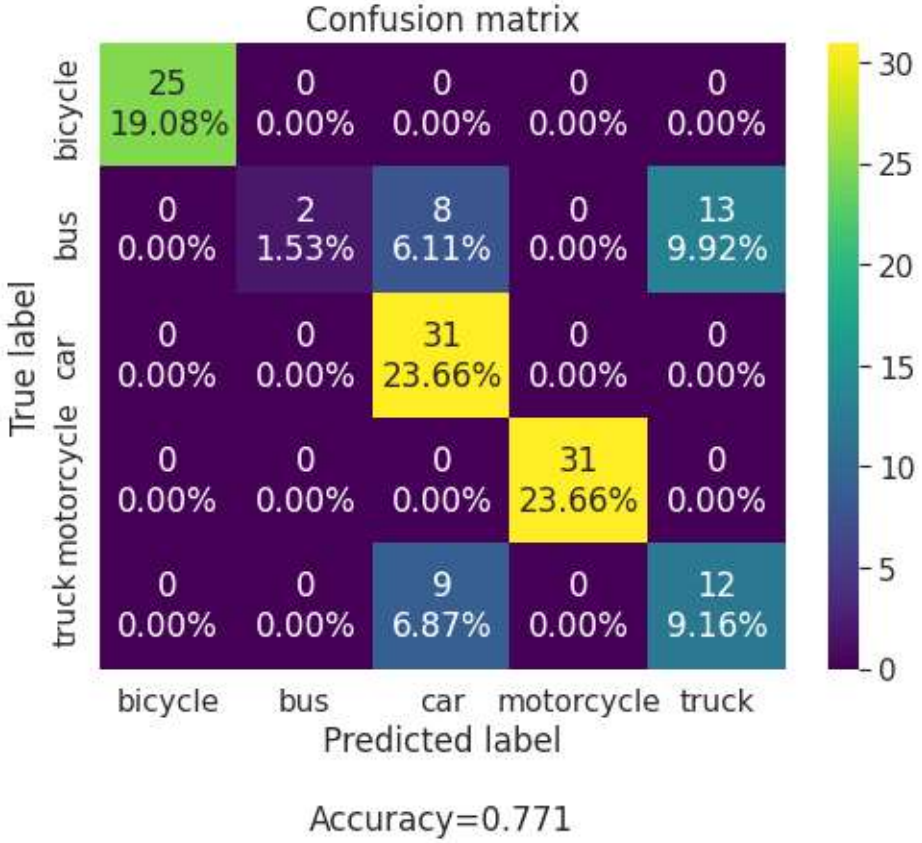
MODEL3_BASELINE

MODEL3_BASELINE	testset1	testset2
Accuracy	0,770992	0,97875
F1 score	0,724929	0,978626
ROC AUC	0,967337	0,996771
Precision	0,832831	0,979164
Recall	0,770992	0,97875
Cohens kappa	0,709234	0,973437

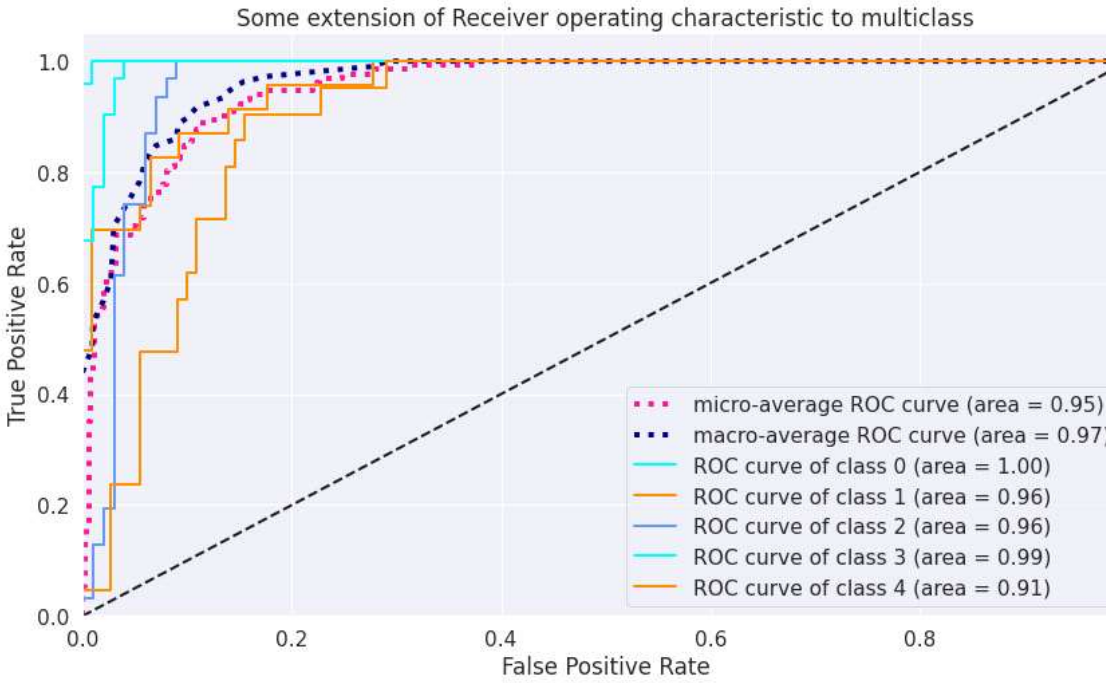
Metrics values obtained by this model are slightly better than in MODEL2 _BASELINE. It comes from the fact that the validation set is added to the train set. On the testset1 - accuracy about 77%, f1-score 72,5%, ROC 97%. On the testset2 - accuracy 97,9%, f1-score 97,9%, AUC 99,7%.

- **testset1**

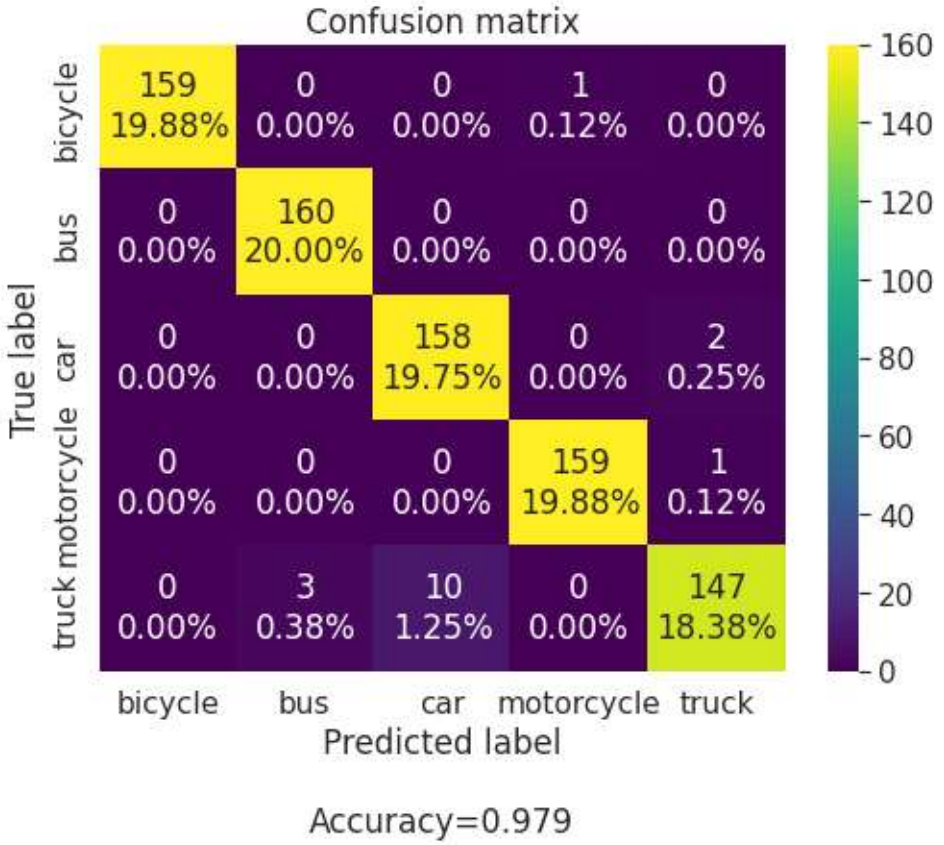


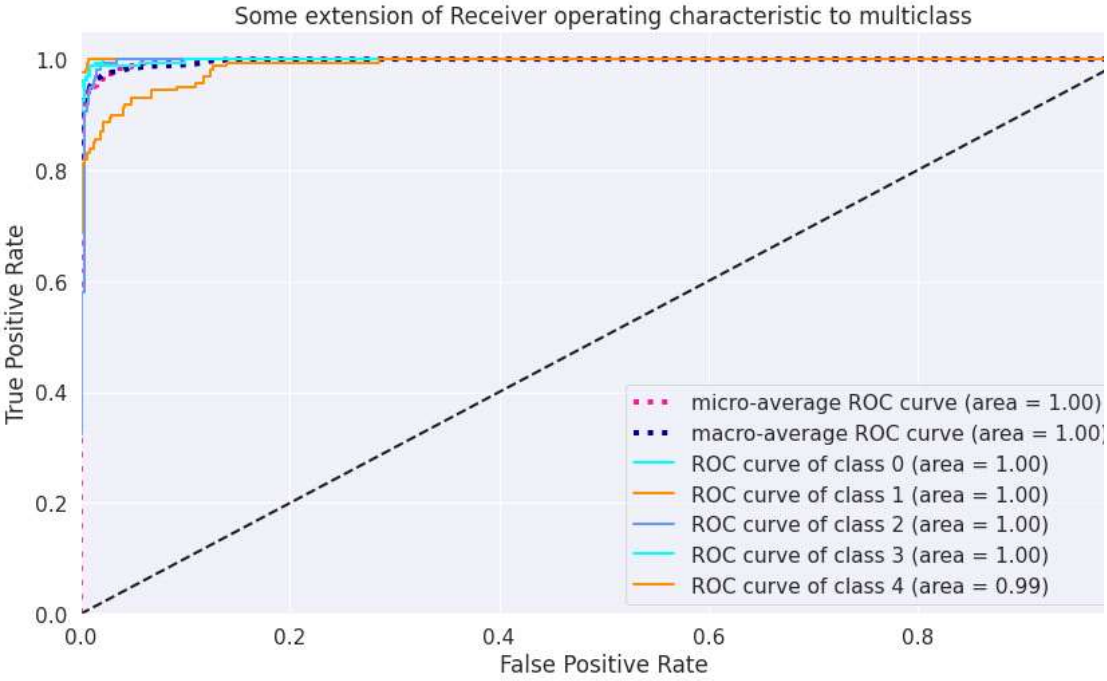


Accuracy=0.771


Some extension of Receiver operating characteristic to multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.95)
macro-average ROC curve (area = 0.97)
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 0.96)
ROC curve of class 2 (area = 0.96)
ROC curve of class 3 (area = 0.99)
ROC curve of class 4 (area = 0.91)

- **testset2**


Confusion matrix
159 19.88%
0 0.00%
0 0.00%
1 0.12%
0 0.00%
0 0.00%
160 20.00%
0 0.00%
0 0.00%
0 0.00%
0 0.00%
0 0.00%
158 19.75%
0 0.00%
2 0.25%
0 0.00%
0 0.00%
0 0.00%
159 19.88%
1 0.12%
0 0.00%
3 0.38%
10 1.25%
0 0.00%
147 18.38%
bicycle
bus
car
motorcycle
truck
True label
Predicted label
160
140
120
100
80
60
40
20
0
Accuracy=0.979


Some extension of Receiver operating characteristic to multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 1.00)
ROC curve of class 2 (area = 1.00)
ROC curve of class 3 (area = 1.00)
ROC curve of class 4 (area = 0.99)

7. Try to improve your results. You are advised (but not limited) to do to at least some of following tasks:
Repeat the training and testing on SPLIT2 datasets.
During the training, calculate all defined testing measures on TRAIN and VAL datasets
Plot measurements' results for the whole training.
Choose "best" models basing on different test measures on VAL datasets
Test selected "best" models on TEST dataset. Do some plots and discuss the results.
Do hyperparameter optimization and/or cross-validation
Perform hard-example mining and retraining
Propose improvements in the whole workflow (data collection, preprocessing, models, training, optimization, hyperparameters, etc.)
Implement these improvements, do the training and testing, and analyze and discuss results.
Do whatever other you want, to improve your models.
Experiment with your models, data, parameters etc. Be creative :) If your experiment gives worse results - no worries, you can still include it in the report.
Relying on validation results, choose and save a few 'best' models as MODELS_IMPROVED.

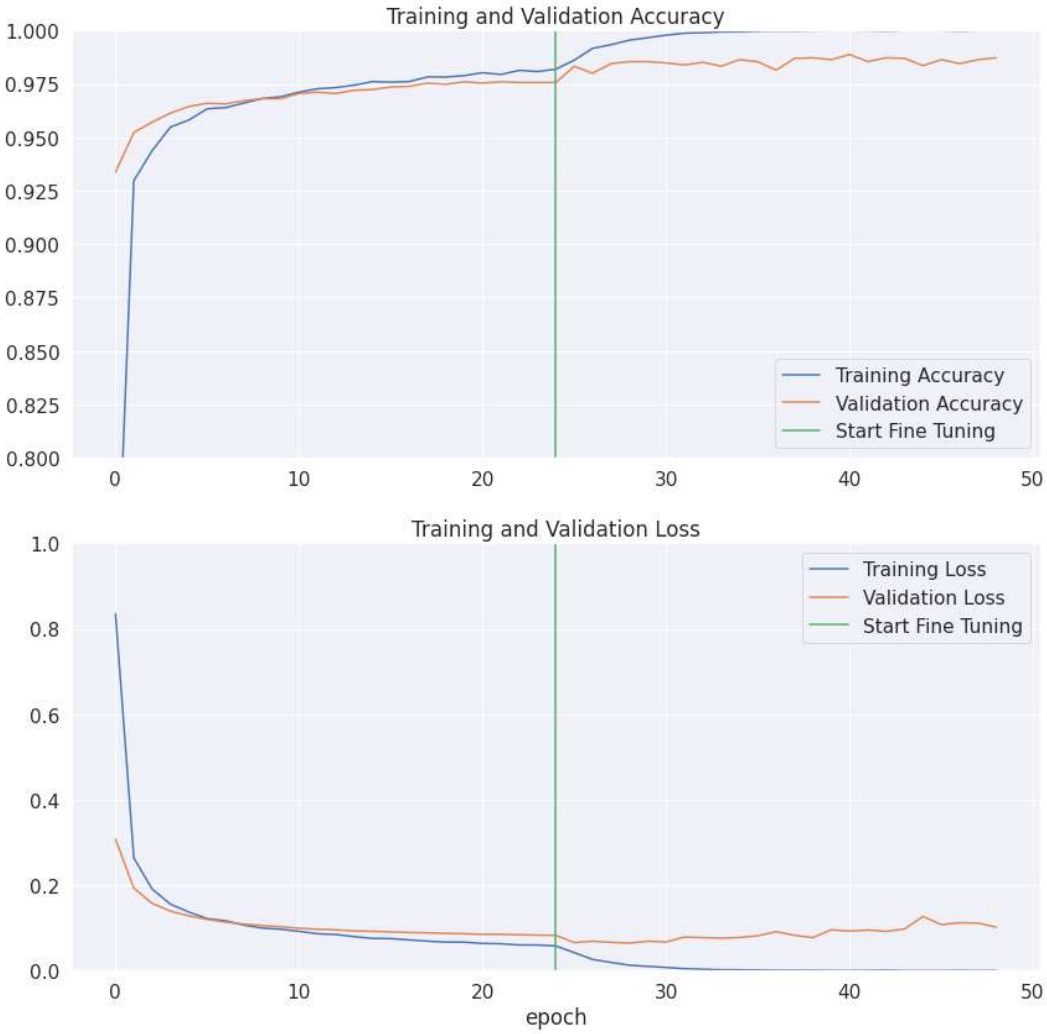
MODEL2_IMPROVED exemplary training data:

Because of the fact that MODEL2_BASELINE has the biggest potential to be furtherly explored (the biggest number of training and validation data, without breaking learning rules as in the case of MODEL3_BASELINE), we have decided to perform fine-tuning on this model. In order to do so, we unfreezed the top 54 layers from MobileNetv2 architecture (layers from 100 to 154) and continued learning the model for the next 25 epochs (50 in total with the previous 25). As the tensorflow documentation tells us, in most convolutional networks, the higher up a layer is, the more specialized it is. The first few layers learn very simple and generic features that generalize to almost all types of images. As you go higher up, the features are increasingly more specific to the dataset on which the model was trained. The goal of fine-tuning is to adapt these specialized features to work with the new dataset, rather than overwrite the generic learning.

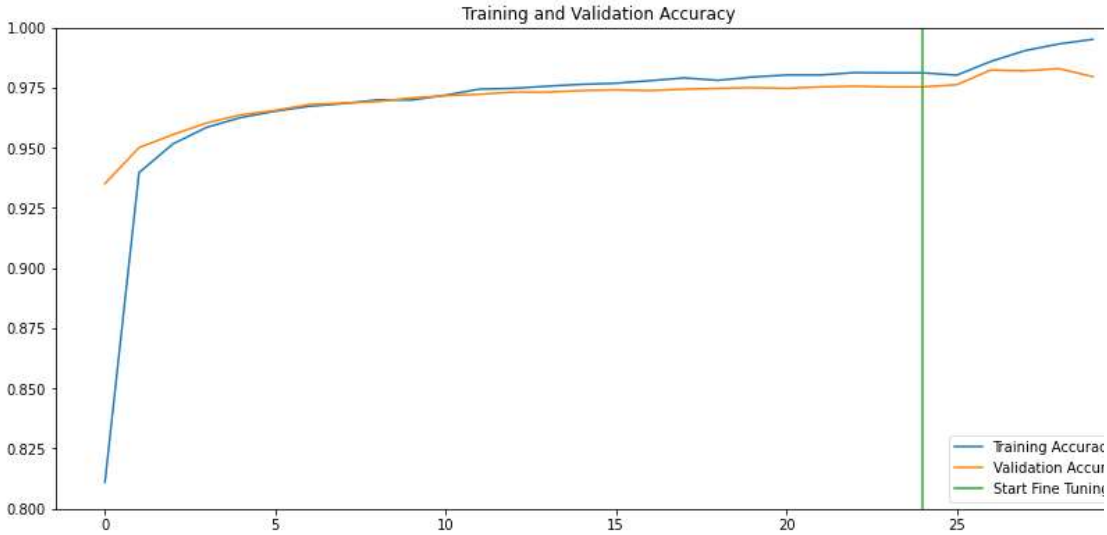
The learning rate for fine-tuning was 10 times smaller than before (learning_rate=0.00001), because otherwise the model could overfit very

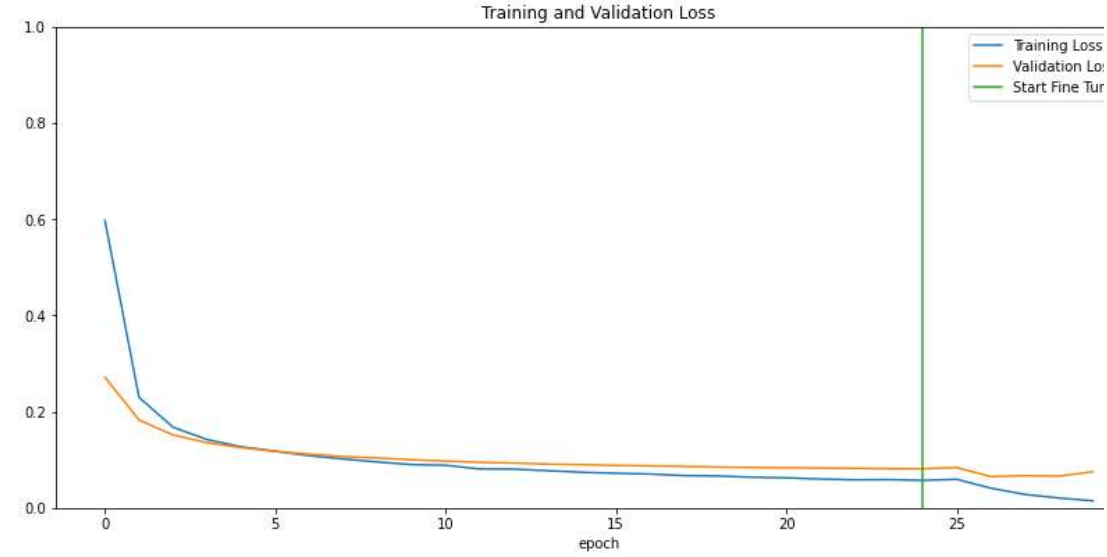
quickly. Early stopping was still set to 3 epochs.

```
Model: "MODEL2_IMPROVED"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_8 (InputLayer)         [(None, 224, 224, 3)]     0
tf.math.truediv_3 (TFOpLambda)  (None, 224, 224, 3)       0
tf.math.subtract_3 (TFOpLambda)  (None, 224, 224, 3)       0
mobilenetv2_1.00_224 (Functional)  (None, 7, 7, 1280)
2257984 global_average_pooling2d_3 (GlobalAveragePooling2D)(None,
1280) 0 dropout_3 (Dropout)          (None, 1280)              0
dense_3 (Dense)              (None, 5)                 6405
=================================================================
Total params: 2,264,389
Trainable params: 1,867,845
Non-trainable params: 396,544
_________________________________________________________________
```



During training, 23 epochs were performed (48 in total). However, the usage of EarlyStopping parameter worked and cut the learning in the 30th epoch.





8. Final model evaluation and discussion

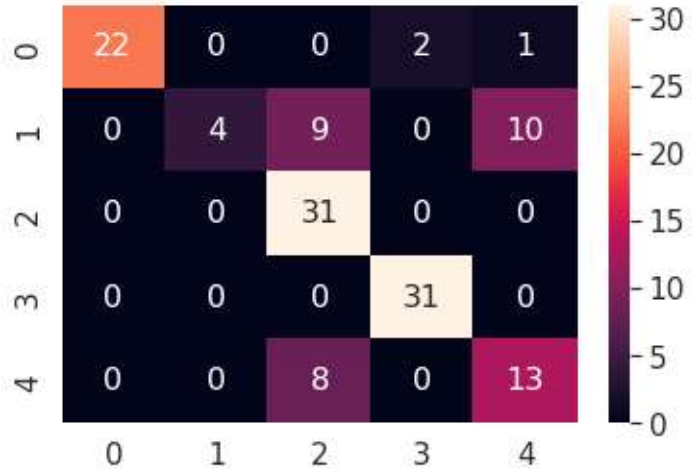
Evaluate MODELS_IMPROVED on the testing dataset using all defined metrics
Plot the results and discuss them. Discussion may include for example:
- Differences and similarities in results for different models and optimizations
- Analysis of errors and their causes
- Differences in results for various test metrics (you shall describe what these differences mean)
- Other observations, considerations and conclusions.

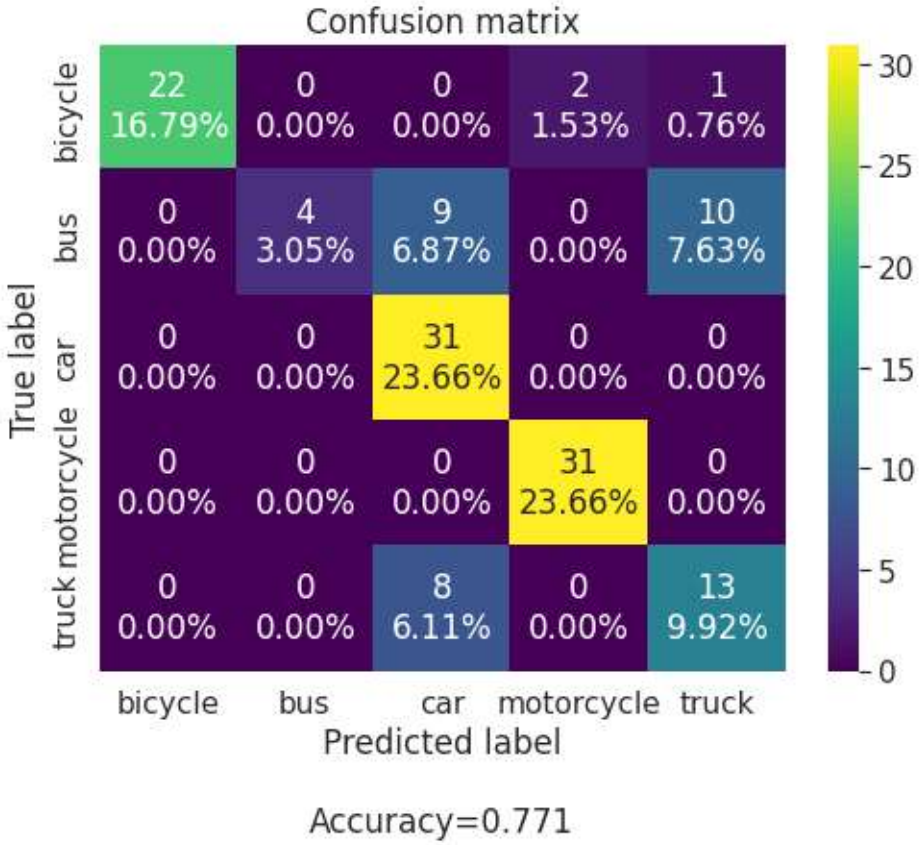
MODEL2_IMPROVED

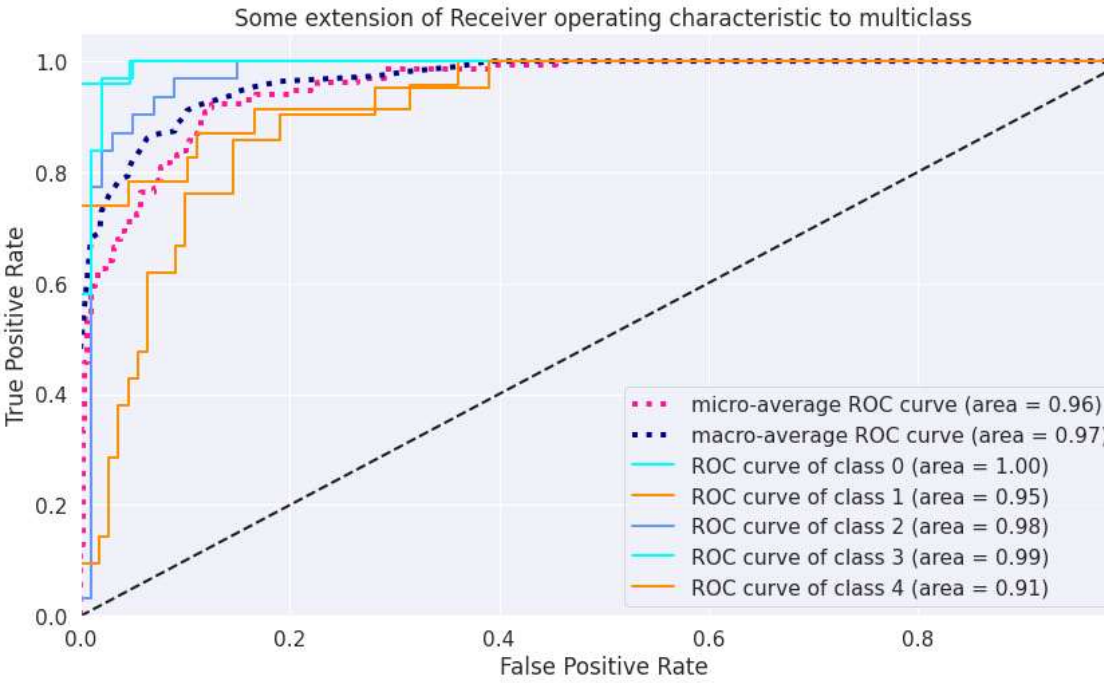
MODEL2_IMPROVED	testset1	testset2	valset	trai
Accuracy	0,770992339	0,97625	0,9872	
F1 score	0,738265	0,976119	0,8932	0,8
ROC AUC	0,970014	0,998572	-	
Precision	0,828374	0,977136	0,7997	0,7
Recall	0,770992	0,97625	1,016	1,0
Cohens kappa	0,708975	0,970313	-	

Tuned model achieves results similar to the MODEL2_BASELINE model. On the first test set, accuracy reached 77,1%, f1-score 73.9%. On the second test set, accuracy reached 97,6%, f1-score also 97,6%.

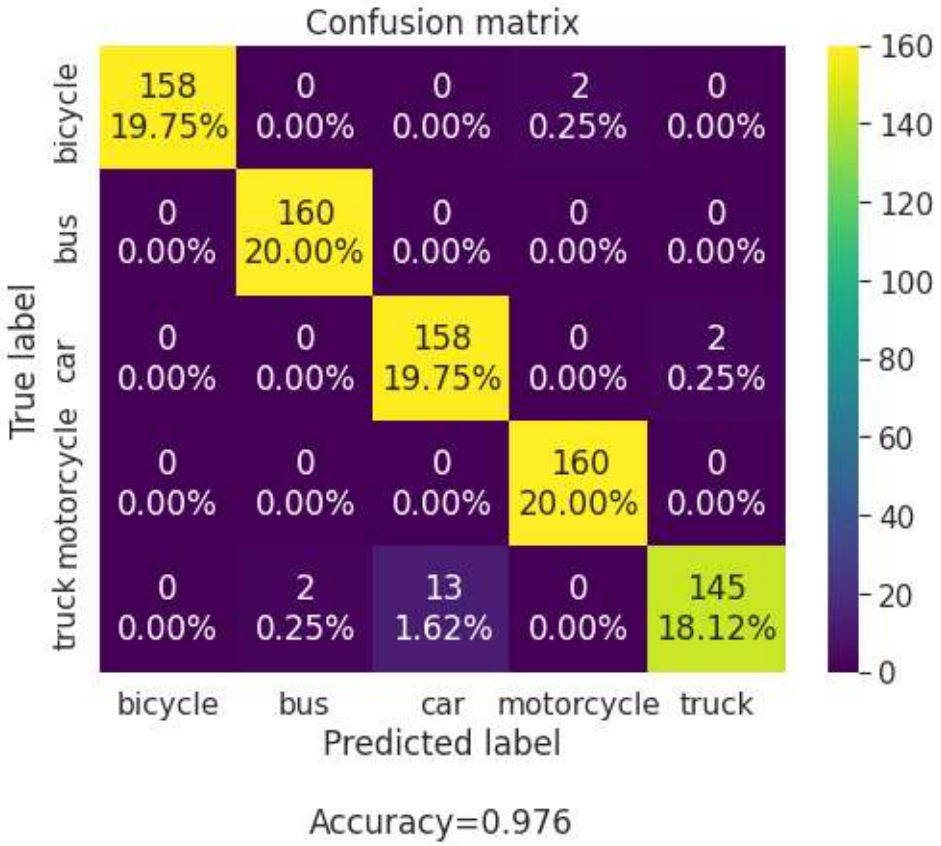
- **testset1**

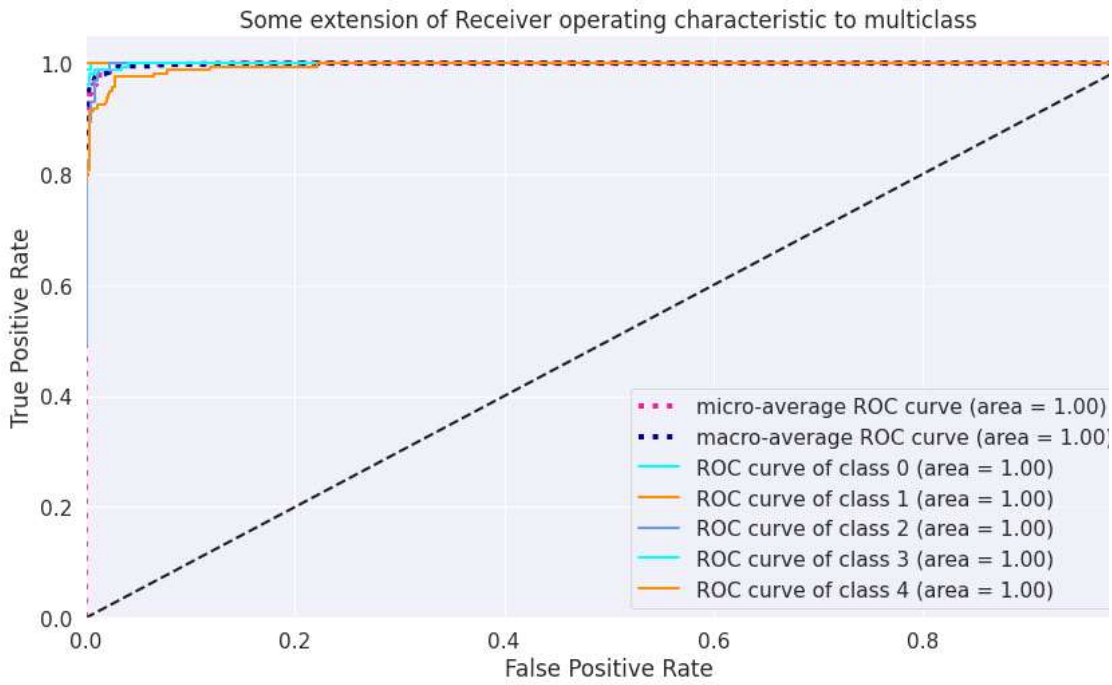



Confusion matrix
bicycle
bus
car
motorcycle
truck
True label
22
16.79%
0
0.00%
0
0.00%
2
1.53%
1
0.76%
0
0.00%
4
3.05%
9
6.87%
0
0.00%
10
7.63%
0
0.00%
0
0.00%
31
23.66%
0
0.00%
0
0.00%
0
0.00%
0
0.00%
0
0.00%
31
23.66%
0
0.00%
0
0.00%
0
0.00%
8
6.11%
0
0.00%
13
9.92%
30
25
20
15
10
5
0
bicycle
bus
car
motorcycle
truck
Predicted label
Accuracy=0.771


Some extension of Receiver operating characteristic to multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.96)
macro-average ROC curve (area = 0.97)
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 0.95)
ROC curve of class 2 (area = 0.98)
ROC curve of class 3 (area = 0.99)
ROC curve of class 4 (area = 0.91)

- **testset2**


Confusion matrix
158
19.75%
0
0.00%
0
0.00%
2
0.25%
0
0.00%
0
0.00%
160
20.00%
0
0.00%
0
0.00%
0
0.00%
0
0.00%
0
0.00%
158
19.75%
0
0.00%
2
0.25%
0
0.00%
0
0.00%
0
0.00%
160
20.00%
0
0.00%
0
0.00%
2
0.25%
13
1.62%
0
0.00%
145
18.12%
True label
bicycle
bus
car
motorcycle
truck
bicycle
bus
car
motorcycle
truck
Predicted label
160
140
120
100
80
60
40
20
0
Accuracy=0.976



9. Project summary. Summarize the entire project. Write about problems you encountered and what you learned.

The goal of the entire project was to create the system (machine learning model) which will recognize ground vehicles in the photos and classify them accordingly. The goal was fully achieved and the results are satisfactory. The featured classes were car, bus, truck, motorcycle and bicycle. In the case of motorcycles and bicycles the classification went perfectly - the model has no problem classifying it and usually makes no mistakes. In the case of car, bus and truck mistakes occurs. The reason for that is the problem with distinguishing minibus-like passenger cars and vans. The possible solution for the problem would be to create a separate class for the minivan-like vehicles and store all ambiguous examples there.

The table below presents the comparison of all the models used and evaluated in the project. The source code is attached to the report.

model_name	MODEL1_BASELINE	MODEL2_BASELINE	MODEL3_BASELINE	MODEL2_IMPROVE
train_examples	6474	13151	16437	131
val_examples	1294	3286	3286	32
epochs_performed	25	25	25	
learning_rate	0.0001	0.0001	0.0001	0.000
optimizer	Adam	Adam	Adam	RMSpr
batch_size	32	32	32	
val_loss	0.086922	0.081698	0.048375	0.0744
val_accuracy	0.972179	0.974437	0.985697	0.9875
test1_loss	0.832436	0.912332	0.803597	1.6623
test1_accuracy	0.770992	0.763359	0.770992	0.7557
test2_loss	0.099143	0.068278	0.065667	0.0863
test2_accuracy	0.965	0.9775	0.97875	0.976
test1_f1_score	0.71651	0.701382	0.724929	0.7054
test1_ROC	0.96605	0.95734	0.967337	0.9680
test2_f1_score	0.964818	0.977309	0.978626	0.9761
test2_ROC	0.99593	0.996211	0.996771	0.9985