Maciej Rosoł, nr KD-247

Robert Kołakowski, nr KD-278

Biology Inspired Computations Project
Final Report

PB3. Application of the deep learning networks to recognize symbols in pictures

Project Overview

The aim of the project is to design and instantiate two deep network architectures that can be used for recognition of the predefined objects in images. The selected models are to use chosen optimization algorithms that could allow time-efficient training of the networks. Moreover, trained neural network classifiers will undergo the assessment to verify their accuracy and classification capabilities. Afterwards the quality comparison is to be made to enable selection of more suitable architecture in the given task.

Environment and tools

Foreseen implementation of the CNN classifier will use Python as a core and pythonic solutions that simplify deep neural network design and instantiation. Initial list of software packages and platforms to be used include:

- Tensorflow – end-to-end machine learning platform,
- Keras - python-based deep learning framework which provides APIs for creating Deep Neural Networks thus simplifying and accelerating its deployment. In the developed solution Keras will use Tensorflow as the backend,
- NumPy – package for handling multidimensional arrays and matrices,
- Seaborn/matplotlib - data visualization tools,
- sklearn – package providing multiple ML-related methods and classes.

The initial selection of the software tools was based on the provided capabilities, simplicity as well as other representative factors such as e.g., popularity of usage in classification tasks. For instance, Keras interface offers readymade building blocks thus lowering the entry barrier and increasing speed of implementation of deep-learning architectures. Moreover, Keras is a solution that is used widely in research community which include i.a. CERN, NASA or NIH.

The other solution which was considered was PyTorch. However, despite several methods for implementation of DNN solutions, it possesses some drawbacks in comparison to Keras with Tensorflow backend, which include:

- Low level API – more complex definition of network architecture
- Lower readability and concision
- Harder debugging

On the other hand, PyTorch performs considerably better when dealing with very large datasets. Nonetheless, the object recognition tasks will be realized using a relatively small set of images and less “production-like” environment, hence Keras seems to be the optimal solution.

Dataset and classes

The chosen dataset *is the Linnaeus 5 dataset,* which contains 8000 RGB pictures of size of 256 by 256 pixels that are grouped into five equipotent sets, where each represents one of the class: berry, bird, dog, flower and other (for the negative set). The dataset can be downloaded from [1]. Exemplary image from each class has been presented in Fig. 1.

Fig. 1. Images belonging to classes: berry, bird, dog, flower, other (from left to right).

Architecture overview

The designed classifiers will utilize Convolutional Neural Network (CNN) architecture as the basis. Optimizers that are considered to be used during the training of the networks include stochastic gradient descent or Adaptive Moment Estimation (ADAM). The initial selection has been based on the frequency of exploitation as well as achieved results and convergence time. Most probably, the designed architecture will utilize Rectified Linear Unit (ReLU) activation function in dense layers due to proven faster convergence and computational efficiency (in comparison to sigmoid like functions) [2]. The output layer will use the Softmax activation function, so the output values will determine the probability of the image belonging to each class.

Two well-known CNN architectures have been chosen for the classification task: MobileNet and Resnet50.

MobileNet

MobileNets [3] are small, low-latency, low-power CNN models which enable robust parameterization to enable operation in environments with resource constraints e.g., mobile devices. In addition to classification tasks, they can be utilized for object detection, embeddings and segmentation, which is a feature typical for large scale models, such as Inception or Xception.

The main building blocks of MobileNets are depthwise separable convolutions [4] presented in Fig. 2. Depthwise Separable Convolution consists of two steps. Frist, depthwise convolution applies a single convolutional filter per each input channel. Afterwards, the linear combination of the output of the depthwise convolution is created by using pointwise convolution.

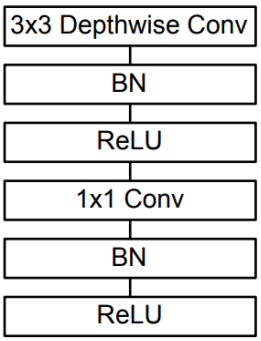


Fig. 2. Depthwise seperable convolution block.

The MobileNets also utilize standard convolutional layer (with batch normalization and ReLU activation function). The whole structure of the used network has been presented in Table 1.

Table 1. MobileNet architecture.

Type/Stride	Input Size	Filter Shape	Function
Conv / s2	224 × 224 × 3	3 × 3 × 3 × 32	Feature extraction
Conv dw / s1	112 × 112 × 32	3 × 3 × 32 dw	
Conv / s1	112 × 112 × 32	1 × 1 × 32 × 64	
Conv dw / s2	112 × 112 × 64	3 × 3 × 64 dw	
Conv / s1	56 × 56 × 64	1 × 1 × 64 × 128	
Conv dw / s1	56 × 56 × 128	3 × 3 × 128 dw	
Conv / s1	56 × 56 × 128	1 × 1 × 128 × 128	
Conv dw / s2	56 × 56 × 128	3 × 3 × 128 dw	
Conv / s1	28 × 28 × 128	1 × 1 × 128 × 256	
Conv dw / s1	28 × 28 × 256	3 × 3 × 256 dw	
Conv / s1	28 × 28 × 256	1 × 1 × 256 × 256	
Conv dw / s2	28 × 28 × 256	3 × 3 × 256 dw	
Conv / s1	14 × 14 × 256	1 × 1 × 256 × 512	
5x Conv dw / s1 + Conv / s1	14 × 14 × 512 14 × 14 × 512	3 × 3 × 512 dw 1 × 1 × 512 × 512	
Conv dw / s2	14 × 14 × 512	3 × 3 × 512 dw	
Conv / s1	7 × 7 × 512	1 × 1 × 512 × 1024	
Conv dw / s2	7 × 7 × 1024	3 × 3 × 1024 dw	
Conv / s1	7 × 7 × 1024	1 × 1 × 1024 × 1024	
Avg Pool	7 × 7 × 1024	Pool 7 × 7	Classification
FC / s1	1x1x1024	1024 × 5	
Activation (Softmax)	1x1x5	Classifier	

The high-level summary of the features of used model has been collected and presented in Table 2.

Table 2. Main features of the used MobileNet architecture.

Model	Trainable params	Non-trainable params	Depth	Input	Output
MobileNet	5,125	3,228,864	88	224x224x3	1x5

According to evaluation provided by Keras [5] and presented in Table 3, the model typically reaches accuracy in between 70 to 90% during classification of the members of the ImageNet dataset (1000 classes).

Table 3. Main features and evaluation of MobileNet provided by Keras.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
MobileNet	16 MB	0.704	0.895	4,253,864	88

Since the aim of the classification task is to assign the image to one of the 5 classes, the achieved accuracy will probably exceed the values presented in the table above, due to better specialization and higher margin for potential errors resulting from lower diversity of objects. Nevertheless, it shows the general values to be expected and to be aimed for during preparation of the model.

ResNet50

ResNet50 is a type of Residual Network, presented in 2015. This is the type of network, that won the 1st place on the ILSVRC 2015 classification task [5]. This type of architecture can be used for many computer visions tasks and it allows to train very deep neural networks, even over 150 layers. ResNet is the first architecture, that introduced the skip connection. In case of CNN the degradation problem might occur, which results in decreasing the accuracy with the increasing number of layers. This issue was addressed by Residual Network and adding skip connections to the architecture. This type of DNN architecture uses residual learning blocks as its component. Each of those blocks contain three layers:1×1, 3×3, and 1×1 convolutions, where the 1×1 layers are responsible for reducing and then increasing dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions. The residual learning block architecture is presented in Fig. 3.

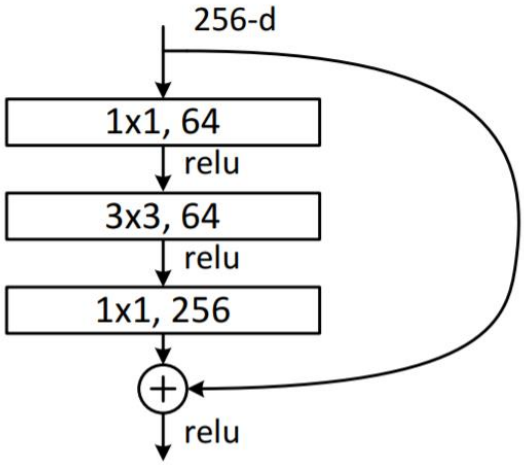


Fig. 3 Residual learning block for ResNet50.

Normally ResNet50 network has input size equal to 224x224x3, however it was changed to 256x256x3 to match the size of used images. Additionally, two fully connected layers were added with 1024 neurons each and dropout regularization with rate equal to 0.05. All weights from the original ResNet50 architecture were frozen, except from the last 4 layers. The whole used architecture of used network based on ResNet50 is presented in Table 4.

Table 4. Used neural network architecture based on ResNet50.

Layer name	Output size	Filter shape		Function
conv1	112x112	7x7, 64 stride 2		Feature extraction
conv2_x	56x56	3x3 max pool, stride 2		
		1x1, 64 3x3, 64 1x1,256	x3	
conv3_x	28x28	1x1, 128 3x3, 128 1x1, 512	x4	
conv4_x	14x14	1x1, 256 3x3, 256 1x1, 1024	x6	
conv5_x	7x7	1x1, 512 3x3, 512 1x1, 2048	x3	
Avg Pool	1x1x2048	Pool 7x7		Classification
Dense	1x1x1024	1x2048x1024		
Dropout		Rate 0.05		
Dense	1x1x1024	1x1024x1024		
Dropout		Rate 0.05		
Softmax	1x1x5	1x1024x5		

The high-level summary of the features of used model has been collected and presented in Table 5.

Table 5. Summary of ResNet50 model.

Model	Trainable params	Non-trainable params	Depth	Input	Output
ResNet50	4,207,621	22,532,992	50	256x256x3	1x5

According to evaluation provided by Keras [5] and presented in Table 6, the model typically reaches accuracy in between 70 to 90% during classification of the members of the ImageNet dataset (1000 classes).

Table 6. Evaluation of ResNet50 on ImageNet dataset.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
ResNet50	98 MB	0.749	0.921	25,636,712	50

Training approach

The selected models contain a very large number of parameters, which makes them difficult to train in non-cluster like environments in the relatively short time. Therefore, we prepared our CNNs, based on available pre-trained models provided by Keras [6], and adapting them by using transfer learning. Transfer learning is the approach of using knowledge gained during solving one problem and applying it to a different but related problem. Typically, in terms of convolutional neural networks, the gained knowledge relates to trained convolutional layers which act as features extractors. The adaptation of the network to specific classification task is performed in the dense layers (weights in convolutional layers are not modified during training) that are capable of interpreting the found patterns. In some cases, i.e., if the model does not provide satisfying classification results, the common practice is fine tuning of the last few convolutional layers to better fit the specificity of the classification problem.

To test the generalization of the prepared models and to choose the best hyperparameters for the model or the model architecture, 5-fold cross validation has been performed. During each iteration of cross-validation, and each epoch of training, the validation categorical accuracy has been observed as the main criteria of selection. The models that proved to give the best results during each iteration (CNN was monitored during the training) were stored. The model with the architecture and hyperparameters, which obtained the best average categorical accuracy after 5-fold cross validation was trained on the whole training dataset and then its performance was evaluated on the test set.

Evaluation approach

Evaluation of both CNN designs will involve comparison by means of the achieved quality measures that include: accuracy, specificity, precision, recall and F1 score. Receiver operating characteristic (ROC) curves for each individual class will also be plotted and area under the curve (AUC) will be calculated. Moreover, the assessment of time complexity of training is planned to be performed.

Results

MobileNet

During preliminary tests, the models achieved satisfactory results usually within 15-20 epochs of training,therefore, the duration of the training was set to a latter margin of 20 epochs. The other reason to limit the number of epochs to 20 was to reduce the network training time to reasonable length. The selected loss function was categorical cross entropy.

The optimization was performed by using ADAM algorithm with the constant learning rate during the training. In order to select learning rate that enables fast convergence, five models have been trained and evaluated using 5-fold CV with the learning rate (lr) hyperparameter equal to: 1e-2, 1e-3, 1e-4, 1e-5 and 1e-6.

The best results have been achieved while setting **lr=1e-4**. In Table 7 the qualitative parameters achieved by the best model during cross-validation have been presented.

Table 7. Metrics obtained for the lr equal to 1e-4 during each iteration of cross-validation process (k).

k	Categorical accuracy (training set)	Categorical accuracy (validation set)	Loss (training set)	Loss (validation set)
1	0.9806	0.9617	0.0762	0.1183
2	0.9829	0.9617	0.0675	0.1249
3	0.9796	0.9675	0.0809	0.1126
4	0.9804	0.9633	0.0714	0.1127
5	0.9760	0.9633	0.0865	0.1178
AVG	**0.9799**	**0.9635**	**0.0765**	**0.1173**

The comparison of the results obtained during fine tuning of the learning rate hyperparameter are provided in Table 8. For each case the averages of accuracies and losses obtained during 5-fold CV have been presented together with the models' performance after training on the whole training set and testing on the separate test set.

Table 8. Metrics of best models trained during each iteration of cross-validation process (k).

Learning rate	AVG categorical accuracy (training set)	AVG categorical accuracy (validation set)	Categorical accuracy (test set)	AVG Loss (training set)	AVG Loss (validation set)	Loss (test set)
1e-2	0.9635	0.9502	0.9465	0.0903	0.1263	0.1335
1e-3	0.9678	0.9528	0.9562	0.0817	0.1286	0.1275
1e-4	**0.9799**	**0.9635**	**0.9670**	**0.0765**	**0.1173**	**0.1150**
1e-5	0.9528	0.9599	0.9602	0.1397	0.1328	0.1294
1e-6	0.9449	0.9460	0.9545	0.1488	0.1437	0.1349

Each of the models performs well, however the very best one has been obtained while using ***lr=1e-4.*** For other learning rates the models seem to not converge equally well which is the effect of too fast learning for lr>1e-4 (divergence) and too slow for lr<1e-4 (not enough epochs of training). The exemplary plots of the accuracy and loss function that have been obtained during cross-validation procedure are shown in Figures 4 and 5.

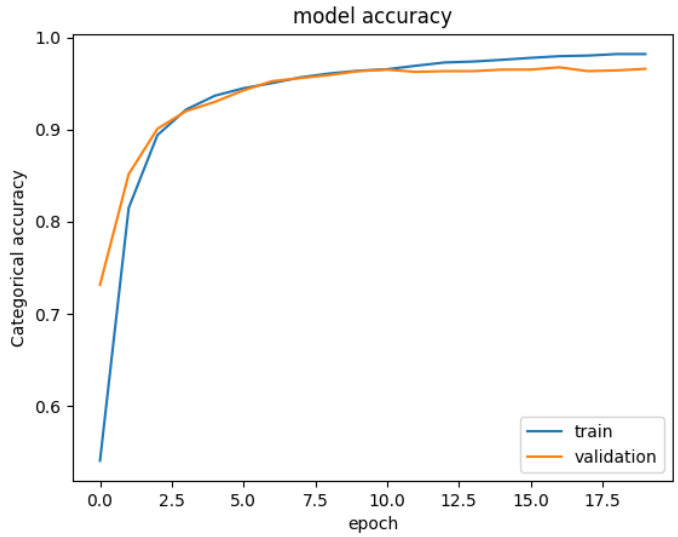


Fig. 4. Exemplary plot of categorical accuracy of the model on training and validation datasets.

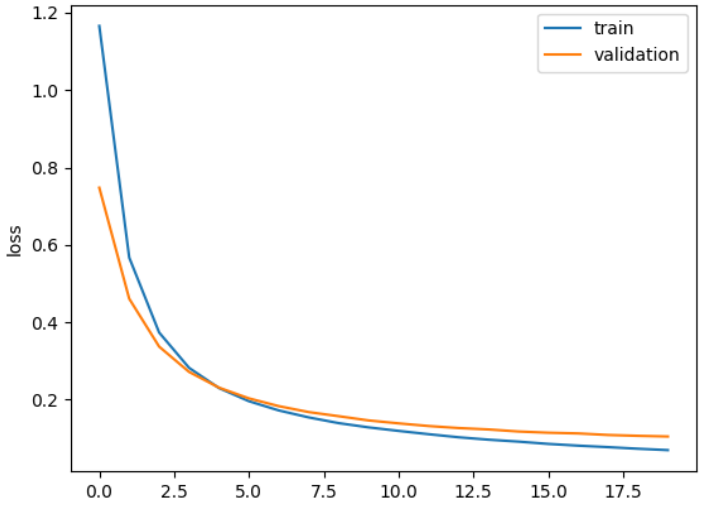


Fig. 5. Exemplary plot of loss function (categorical cross entropy) for training and validation datasets.

The MobileNet model has been once again trained with lr=1e-4, using the whole training set and afterwards tested on a separate test set of 2000 images (400 for each class). The accuracy and loss plots obtained during training are shown in Fig. 6 and Fig. 7.

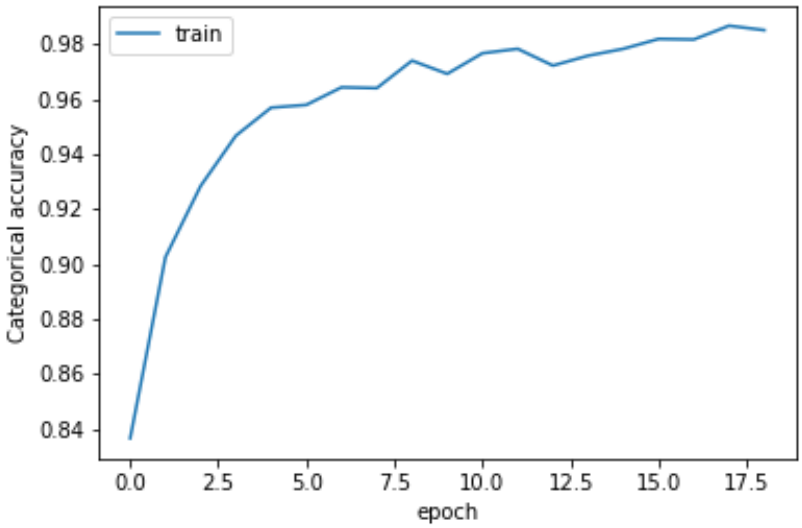


Fig. 6. Categorical accuracy achieved by the model during training.

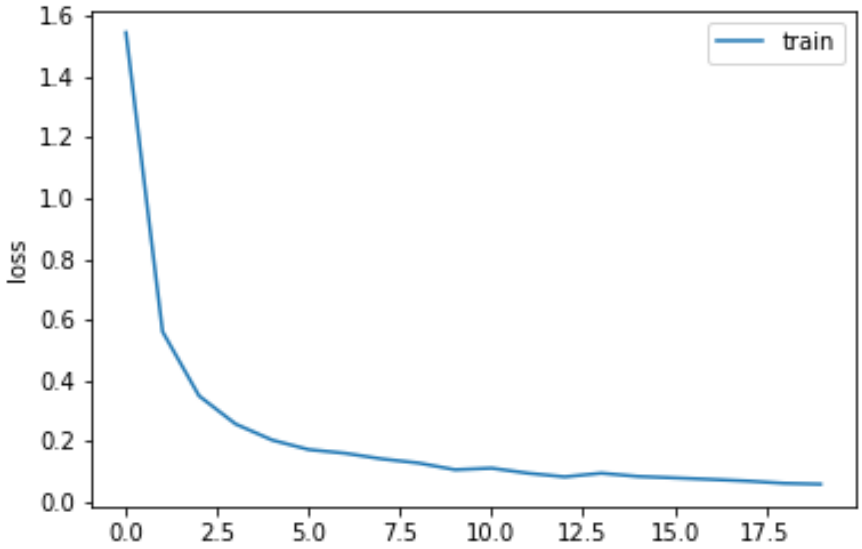


Fig. 7. Loss function values (categorical cross entropy) during training.

The classification results obtained for the training and test set have been aggregated in form of a confusion matrix and presented in fig. 8 and fig. 9.

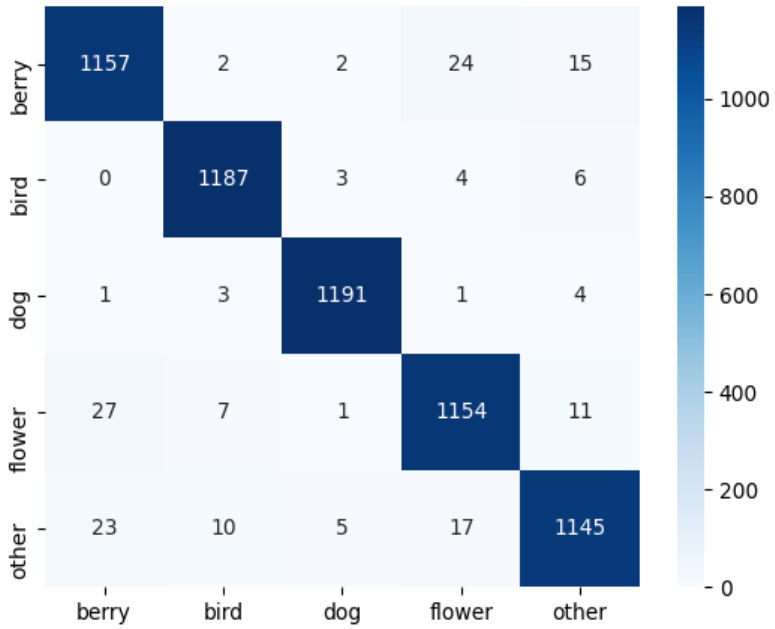


Fig. 8. Confusion matrix obtained for the training set.

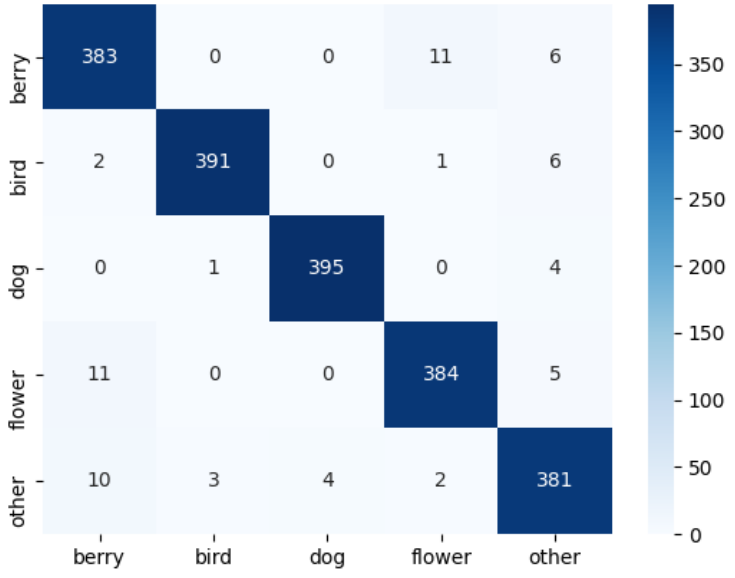


Fig. 9. Confusion matrix obtained for the test set.

To simplify further operations the basic metrics (TP, TN, FP, FN) obtained during classification of the test set members have been presented in Table 9.

Table 9. Classification results (TP – True Positive, TN – True Negative, FP- False Positive, FN – False Negative).

Class	TP	TN	FP	FN
Berry	383	1577	23	17
Bird	391	1596	4	9
Dog	395	1596	4	5
Flower	384	1586	14	16
Other	381	1579	21	19

Based on the results in Table 10, metrics that enable further assessment of the classifier have been calculated.

Table 10. Classification metrics achieved for each class from the test set.

Class	Precision	Recall	Specificity	Accuracy	F1 score
Berry	0.94	0.96	0.98	0.98	0.95
Bird	0.99	0.98	0.99	0.99	0.98
Dog	0.99	0.99	0.99	0.99	0.99
Flower	0.96	0.96	0.99	0.98	0.96
Other	0.95	0.95	0.98	0.98	0.95

The trained classifier achieves best results for bird and dog classes, which are the ones that are most distinct from the others. The worst results have been achieved for berry class and other. Berries have been mistakenly taken for flowers (and vice versa), probably due to similarity in shape (typically round objects). The Other class is a mix of pictures of different objects, therefore with high probability, some of the objects have similar sets of features as berries or flowers (and vice versa). Having analysed the confusion matrix in Fig. 8, the similar behaviour can be observed, i.e., the classifier has much bigger problems with distinguishing berries, flowers and other class for both training and test datasets.

The classifiers ability to distinct each class from the rest i.e., ROC curves, have been presented in Fig. 10 together with the values of AUC (Area Under Curve) – the metric showing the quality of a binary classifier (one vs all approach).

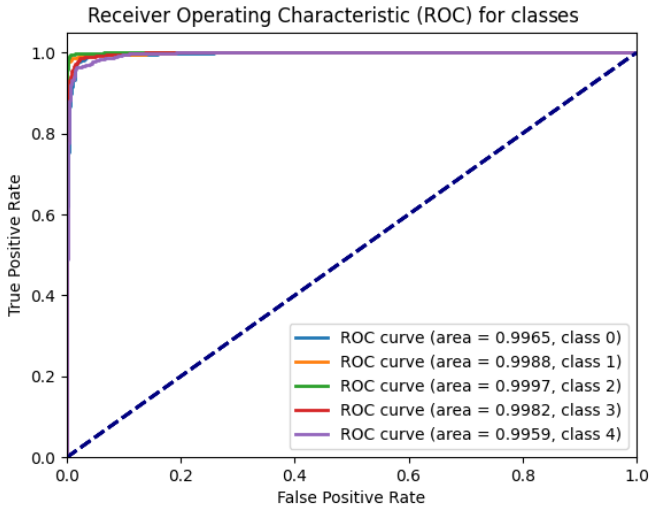


Fig. 10. ROC curves for each class: berry (0), bird (1), dog (2), flower (3), other (4) and respective value of AUC (area).

ResNet50

Three different architecture modifications (different layers were added at the end of the basic ResNet50 architecture) were tested using 5-Fold cross-validation performed on a test set. Each network was trained for 20 epochs with learning rates equal to 1e-3, 1e-4, 1e-5, 1e-6 respectively for each 5 consecutive epochs. As the optimalization method ADAM algorithm was used. Sparse categorical crossentropy was used as a loss function. The final model architecture was chosen based on the highest average accuracy obtained on validation sets.

The qualitative parameters achieved by the network after performing cross-validation has been presented in Table 11. The example of accuracy values is presented in Fig. 11, while the example values of the loss function for training and validation sets are presented in Fig. 12.

Table 11. Metrics of models for the best architecture modification trained during cross-validation process.

k	Categorical accuracy (training set)	Categorical accuracy (validation set)	Loss (training set)	Loss (validation set)
1	0.9925	0.9758	0.3579	0.3281
2	0.9925	0.9742	0.3576	0.3505
3	0.9946	0.9717	0.3488	0.3563
4	0.9954	0.9633	0.3591	0.3684
5	0.9925	0.9650	0.3749	0.3512

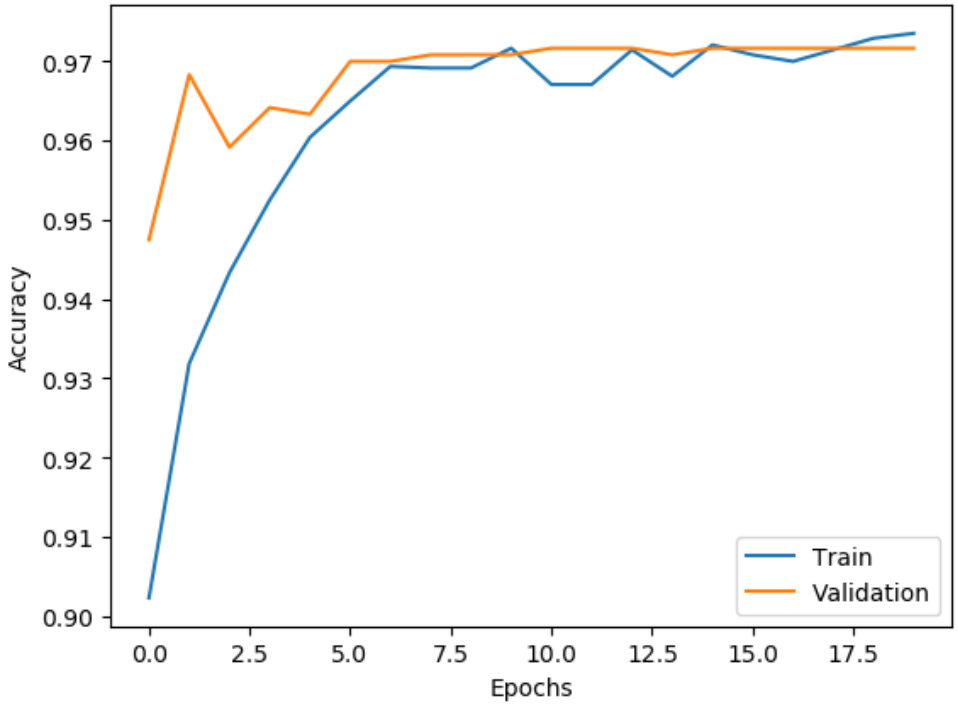


Fig. 11. Example plot of categorical accuracy of the model during cross-validation.

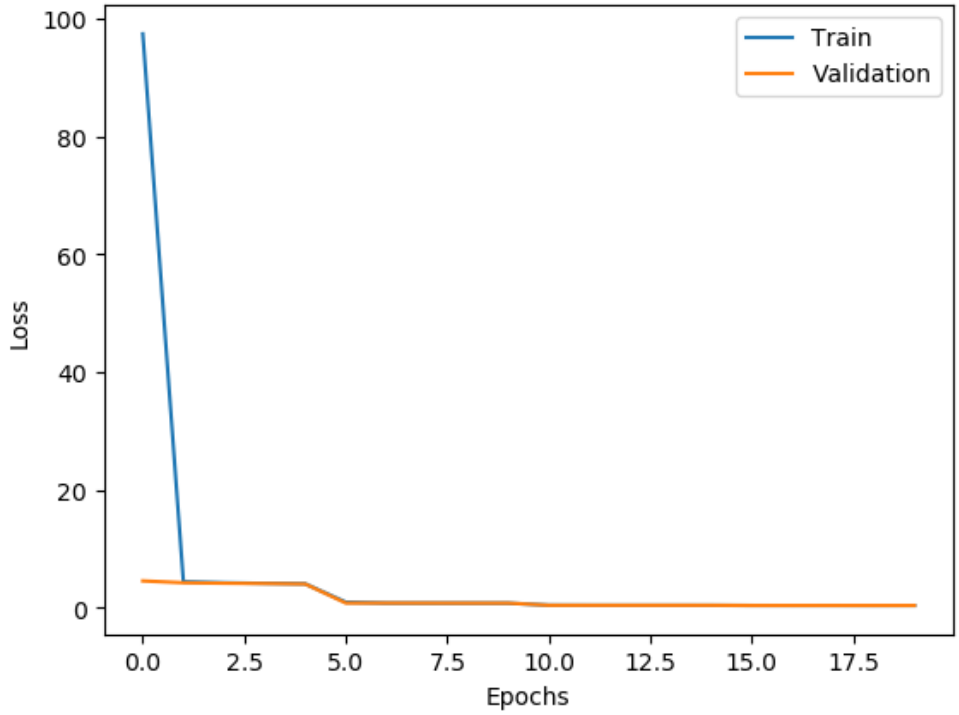


Fig. 12. Example plot of values of loss function of the model during cross-validation.

Based on the comparison of cross-validation results the final architecture was selected (presented in Table 4) and trained on the whole training set. Values of accuracy and loss function during consecutive epochs were visualized in Fig. 13 and Fig. 14 respectively.

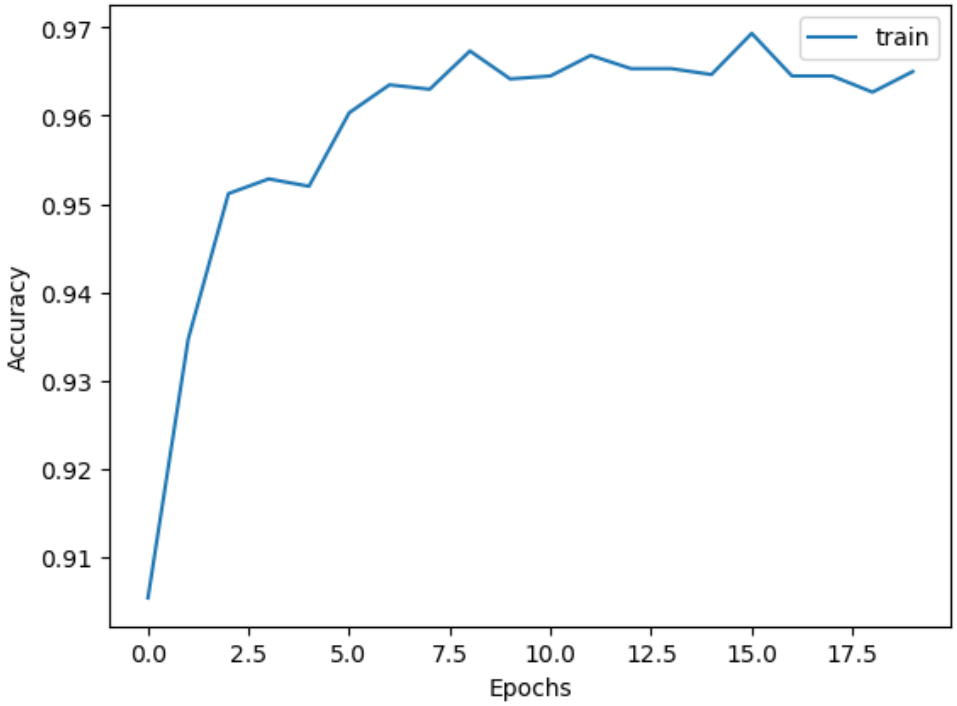


Fig. 13. Categorical accuracy of the final model during training.

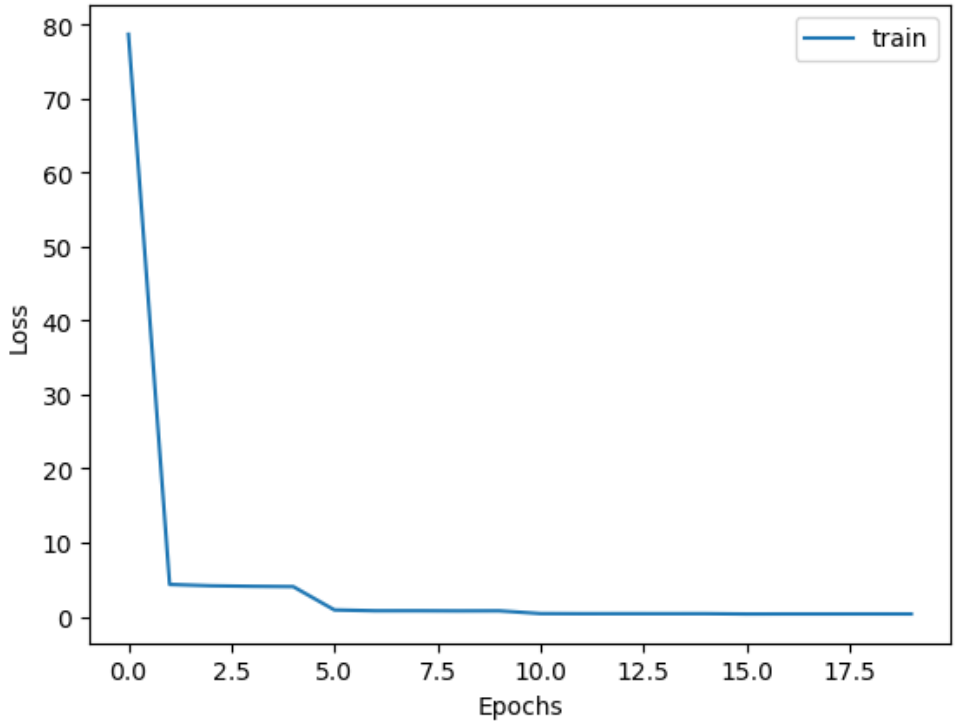


Fig. 14. Values of loss function of the final model during training.

The final evaluation of the model was performed on a separate testing dataset, which contained 2000 images, 400 for each class. The ROC curves for the testing set for each class along with AUC values were presented in Fig. 15.

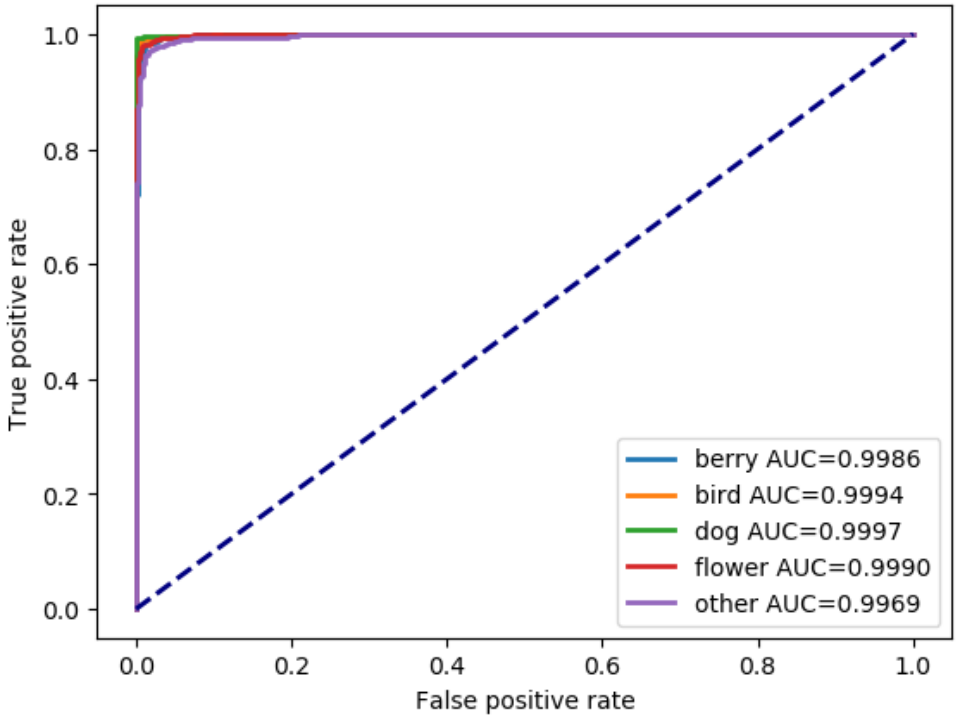


Fig. 15. ROC curves for each class with AUC values.

The confusion matrix obtained on the training set is shown in Fig. 16, while the confusion matrix obtained on the test set is presented in Fig. 17.

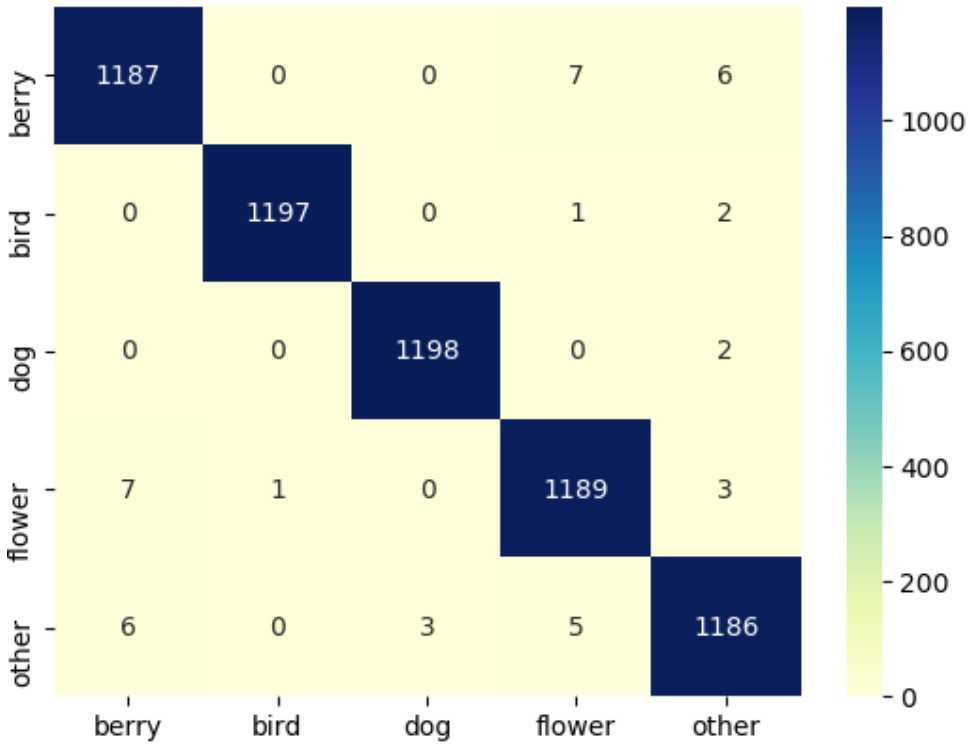


Fig. 16. Confusion matrix calculated on the train set.

Fig. 17. Confusion matrix calculated on the test set.

Metrics for the final model based on the ResNet50 calculated one the test set are presented in Table 12, while the number of false and true positive and negative values are presented in Table 13.

Table 12. Metris characterising the final model calculated on test set.

Class	Precision	Recall	Specificity	Accuracy	F1 score
Berry	0.96	0.98	0.99	0.99	0.97
Bird	0.99	0.99	1.00	0.99	0.99
Dog	1.00	1.00	1.00	1.00	1.00
Flower	0.97	0.97	0.99	0.99	0.97
Other	0.96	0.95	0.99	0.98	0.95

Table 13. The number of true positive (TP), true negative (TN), false positive (FP) and false negative (FN) values for each class.

Class	TP	TN	FP	FN
Berry	390	1582	18	10
Bird	394	1594	6	6
Dog	398	1598	2	2
Flower	386	1589	11	14
Other	379	1584	16	21

Created model obtains accuracy over 90% very quickly, just after the first training epoch. Decreasing the learning rate during training seems to have a good influence on obtaining even higher accuracy (especially at the beginning of the training), as the loss is visibly decreasing when the learning rate is changed. The best results were obtained for “dog” class. Only two pictures of dogs were misclassified and only two photos from other classes were misclassified as dogs. The worst performance was obtained for class “other”, where 21 pictures were misclassified and 16 pictures from other classes were incorrectly labeled as “other” by the model.

Comparison

The general overview of the performance of both architectures as well as main features has been presented in Table 14.

Table 14. Main features of the models based on MobileNet and ResNet50 architectures.

Model	Categorical Accuracy (test set)	Loss (test set)	Training time (step)	Trainable params	Non trainable params	Layers
MobileNet	0.9670	0.1150	176 ms	5,125	3,228,864	88
ResNet50	0.9735	0.4271	2 s	4,207,621	22,532,992	50

Both models proved to be very effective in the classification task for the used dataset with pictures of berries, birds, dogs, flowers, and other objects. Categorical accuracy for both models were greater than 0.96. The model based on ResNet50 obtained a slightly higher accuracy than MobileNet. However, it has

much more trainable parameters resulting in longer time per training step than in MobileNet case. The value of loss function (categorical crossentropy) was smaller for the MobileNet model.

The metrics for all classes obtained from both models are presented in Table 15.

Table 15. Precision, recall, specificity, accuracy and F1 score obtained for each class from MobileNet and ResNet50 models.

Class	MobileNet				
	Precision	**Recall**	**Specificity**	**Accuracy**	**F1 score**
Berry	0.94	0.96	0.98	0.98	0.95
Bird	0.99	0.98	0.99	0.99	0.98
Dog	0.99	0.99	0.99	0.99	0.99
Flower	0.96	0.96	0.99	0.98	0.96
Other	0.95	0.95	0.98	0.98	0.95
Class	**ResNet50**				
	Precision	**Recall**	**Specificity**	**Accuracy**	**F1 score**
Berry	**0.96**	**0.98**	**0.99**	**0.99**	**0.97**
Bird	**0.99**	**0.99**	**1.00**	**0.99**	**0.99**
Dog	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Flower	**0.97**	**0.97**	**0.99**	**0.99**	**0.97**
Other	**0.96**	**0.95**	**0.99**	**0.98**	**0.95**

For both models the best results were obtained for class “dog”. In case of MobileNet all metrics for this class were equal to 0.99, while for ResNet50 they were approximately equal to 1.00. "Bird” class had the second-best score for both models. The worst metrics were obtained for class “other”. Considering the diversity of pictures in the dataset, the per-class accuracy over 0.98 achieved for both of the tested models is a promising result.

The ROC curves are very similar for both MobileNet and ResNet50. It can be observed that the best results were obtained for the "dog" class, while the worst for the "other" class. Based on the shape of the curves, it can be concluded that both models are characterized by very high accuracy for each class. The AUC values for all classes for both models were greater than 0.99. The ROC curves for all individual classes are presented in Appendix to the Report.

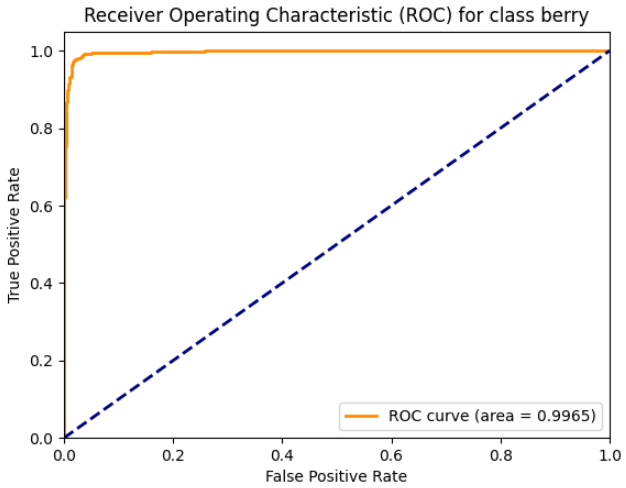
Despite being a much more complex model, ResNet50 performed only a little bit better than MobileNet in terms of accuracy. The reason for such an outcome may be caused by the characteristics of the used dataset i.e., it consisted of only 5 different classes. In such “conditions” the true advantages of ResNet50, which come at a cost of much larger number of parameters and resulting longer training, might not have the opportunity to be fully exposed. Considering the evaluation presented in [5], ResNet50 proved to be much more accurate than MobileNet on the very large datasets (1000 classes). As a conclusion, the MobileNet is more suitable for applications demanding faster learning abilities containing fewer number of classes while ResNet50 provides the capabilities for better operation on much larger and diverse datasets.

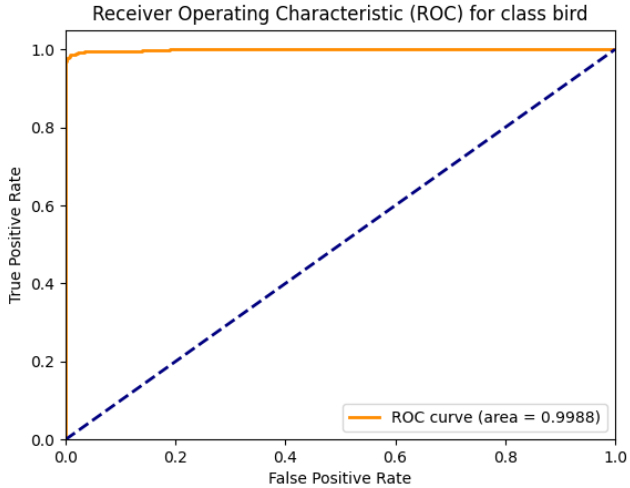
References

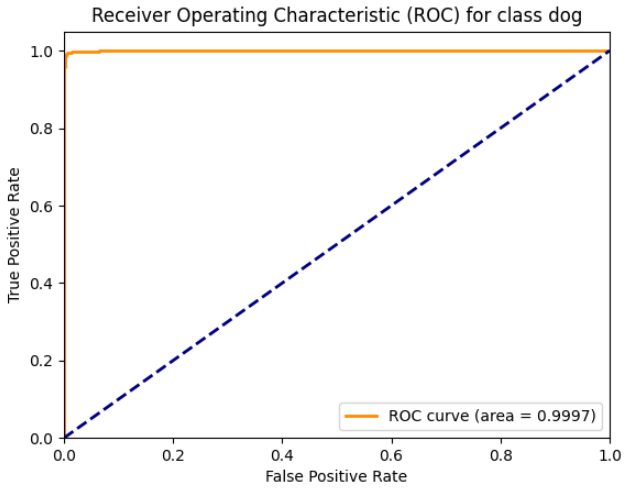
1. Chaladze G., Kalatozishvili L., “Linnaeus 5 Dataset for Machine Learning”, 2017 [Online]. Available: http://chaladze.com/l5/
2. Krizhevsky A., Sutskever I., Hinton G.E., “ImageNet Classification with Deep Convolutional Neural Networks”, [Online]. Available: http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf
3. Howard G. A. et al. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision” Applications”, [Online]. Available: https://arxiv.org/pdf/1704.04861.pdf
4. Guo Y. et al., “Depthwise Convolution is All You Need for Learning Multiple Visual Domains”, [Online]. Available: https://arxiv.org/pdf/1902.00927v2.pdf
5. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun “Deep Residual Learning for Image Recognition” Avaliable: https://arxiv.org/pdf/1512.03385.pdf
6. Keras Applications, [Online]. Available: https://keras.io/api/applications/

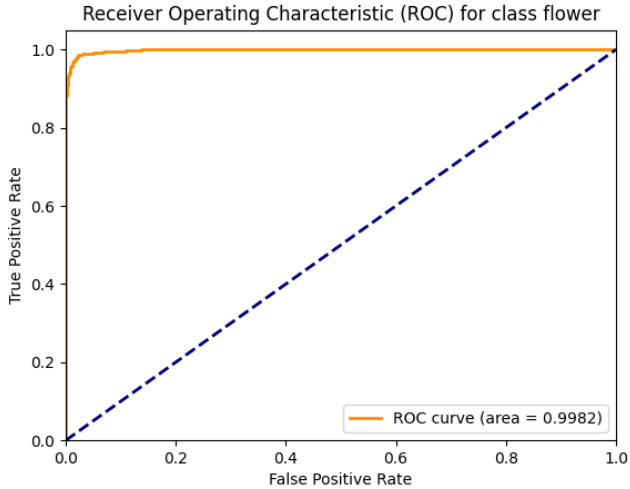
Appendix

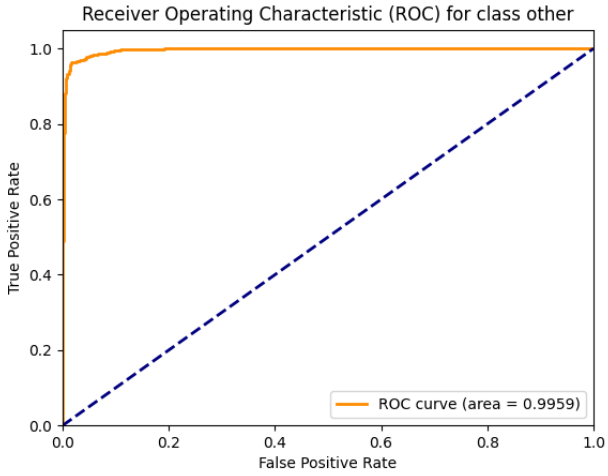
ROC curves for each class for the MobileNet Classifier











ROC curves for each class for the ResNet50 classifier