Flowers classification using convolutional and pre-trained neural networks

Machine Learning lectured by Prof. Pétia Georgieva - June 21, 2020

Bartosz Stoń
DETI
University of Aveiro
Wrocław University of Technology
bston12@gmail.com

Dominik Raška
DETI
University of Aveiro
VSB - Technical University of Ostrava
dominik.raska@gmail.com

***Abstract*—In this report, we afresh discuss one of the most important supervised machine learning method – classification. The objective was to implement a convolutional neural network that would be able to recognise flowers species based on computer image data with the emphasis on accuracy and efficiency of the final model. For this purpose, we again have used a dataset that contains 4214 pictures of flowers of 5 different species. Due to the relatively small database, on-line augmentation technique was performed. Furthermore, to increase accuracy of our model we additionally used pre-trained convolutional neural network VGG16. We compared the obtained results with Project 1 – *Flowers classification using feedforward neural network*, which brought unsatisfactory results due to too simple models and a small training database.**

***Index Terms*—deep learning, convolutional neural networks, data augmentation, flower image classification, multiclass classification, Keras, VGG16**

I. Introduction

The recent advancements of technologies that underline machine learning and image processing have given rise to a vast amount of applications that solve real life issues. Among others, classification of plant species was recognized as a useful and also a challenging problem. Developers of flower recognition systems need to consider a high complexity and intra-class variance of a given plant collection. [1]

To solve this problem, we have previously implemented and tuned a simple neural network with up to several densely connected layers. Here, we will expand our methodology with convolutional neural networks (CNN). These networks have the ability to extract the best features from images by including one or more convolutional layers, and as such, they are widely used for the image classification purposes [2].

In the next section, we will describe the data set and explain how this data were augmented and fed to the CNN models as they were trained. Part III describes the algorithms that were used for the training of the model. Part IV presents the set of trained models and their prediction accuracy. Building upon the best model, in section V, we have fine-tuned some of its parameters and chose the best configuration of the model. After that, our model was retrained and subjected to the final evaluation. From there, we obtained the performance measures that are presented in Part VI. The last section discusses and summarizes the obtained results.

Herein follows our declaration of responsibilities on following parts of the project:

- Bartosz Stoń – data pre-processing and augmentation, architecture of the models, fine-tuning
- Dominik Raška – implementing pre-trained models, training and graph plotting, fine-tuning

II. Data

As in the previous project, database was obtained from Kaggle[1] containing 4323 color JPEG photographs of 5 different flower species (classes). Total number of instances of each class are showed in Table IV. Due to relatively small data set, data augmentation was preformed to increase the amount of training examples. We used on-line implementation with Keras `ImageDataGenerator`, in which data are being constantly loaded from hard disk and distorted to form mini-batches that are constantly passed to a training algorithm. We performed pretesting to choose the best hyper-parameters of augmentation for the training set:

1) re-scale: $1/255$,
2) rotation range: 45,
3) width shift range: 0.2,
4) height shift range: 0.2,
5) shear range: 0.2,
6) zoom range: 0.2,
7) vertical flip: True,
8) fill mode: nearest.

Examples of augmented pictures are presented in Fig. 1. Generator for validation and test data was used only with re-scaling parameter. Furthermore, all images were resized into a resolution 150×150 in order to represent uniform feature space with the final dimensionality given as $150 \times 150 \times 3$. One-hot

encoding was used to represent data classes. The entire dataset was shuffled and split into 2 parts:

1) Training set (80%) – used for performing k-fold cross validation.
2) Test set (20%) – needed to evaluate the final accuracy of the best model.

TABLE I: Data classes

Class	Samples
Daisy	747
Dandelion	1033
Rose	763
Sunflower	728
Tulip	943
	4214

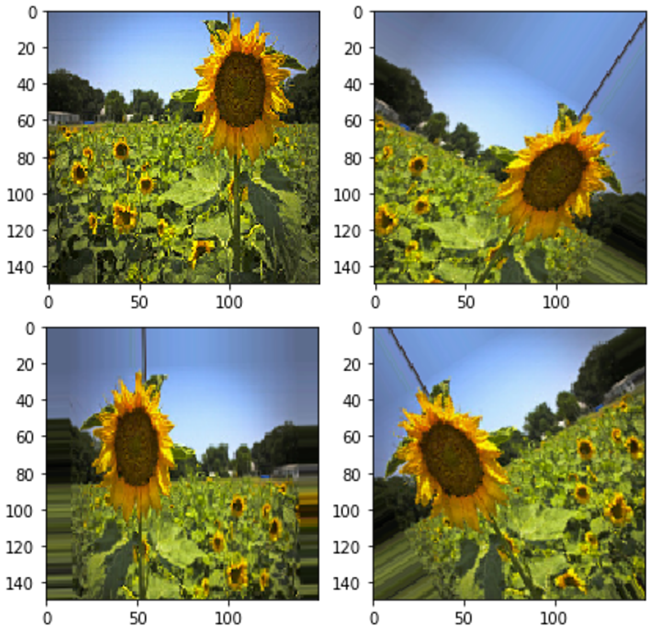


Fig. 1: Examples of augmented data

III. Methodology

For the efficient implementation of CNN models, we have decided to use Keras library (v. 2.3.1) that runs on top of widely used machine learning platform – TensorFlow. [3] Throughout several network architectures we have used convolutional layers, featuring up to 512 filters, ReLU activation function, maxpooling, and in some cases – batch normalization to deal with high variance. At the end of a convolutional cascade, the output is flattened and fed to one or more densely connected layers with ReLU activation. Finally, the output layer is using softmax activation for its 5 nodes, where each node corresponds to one of the classes. While training, the cost function computed as a cross-entropy loss between true labels and predicted labels. The gradient of the cost function is backpropagated through the network while making use of Adam optimization algorithm. [4]

IV. Models

Although our reference model [5] is based on pre-trained convolutional neural network VGG16, we also decided to implement our own CNN models, to compare them. The structures of the proposed models (Models 1-7) are presented in Table II. They are based on models solving similar problems [6]–[8]. Models 1-3 determine three different networks consisting of convolutional and dense layers. Models 4-7 are a modification of previous models with dropout and batch normalization. For each model 6-fold cross validation was preformed giving accuracies presented in Table II. The training and validation accuracy for the best model so far (Model 2) were plotted in Fig. 2.

The reference model [5] consisting of frozen VGG16 [9] CNN and output dense layer is presented in table II as Model 8. The VGG16 is trained on ImageNet, which is a dataset of over 14 million images of everyday usage things [6]. The accuracy of Model 8 (83.12%) corresponds to the referenced one – 83.47%. We proposed two other structures (Models 9 and 10) in order to obtain better performance. Furthermore, we also validated models with Xception pre-trained NN, but due to low accuracy, results are not presented. Because the Model 9 has achieved the highest accuracy we will optimize it in the next section.

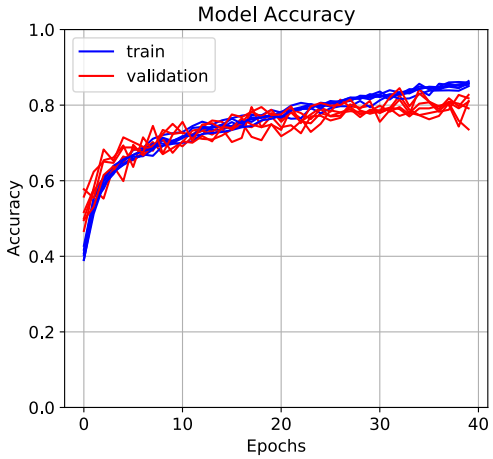


Fig. 2: Convergence of training and validation accuracy for the Model 2.

Model	1	2	3	4 (1)	5 (2)	6 (3)	7 (3)	8	**9**	10
1st Conv Dropout Normalization	32, (5, 5)	32, (3, 3)	64, (3, 3)	32, (5, 5) ✓	32, (3, 3) ✓	64, (3, 3) ✓	64, (3, 3) 0.2 ✓			
2nd Conv Dropout Normalization	64, (3, 3)	64, (3, 3)	128, (3,3)	64, (3, 3) ✓	64, (3, 3) ✓	128, (3,3) ✓	128, (3,3) 0.3 ✓			
3rd Conv Dropout Normalization	96, (3, 3)	128, (3, 3)	128, (3,3)	96, (3, 3) ✓	128, (3, 3) ✓	128, (3, 3) ✓	128, (3, 3) 0.3 ✓	VGG16	VGG16	VGG16
4th Conv Dropout Normalization	96, (3, 3)	128, (3, 3)	256, (3,3)	96, (3, 3) ✓	128, (3, 3) ✓	256, (3,3) ✓	256, (3,3) 0.2 ✓			
5th Conv Dropout Normalization			512, (3,3)			512, (3,3)	512, (3,3) 0.3 ✓			
Flatten Dropout	✓	✓	✓	✓ 0.5	✓ 0.5	✓ 0.5	✓	✓	✓	✓
1st Dense Normalization Dropout	512	512	1024	512 ✓	512 ✓	1024 ✓	1024 0.5		4096	512 0.5
2nd Dense Dropout										512 0.5
Output Dense	5	5	5	5	5	5	5	5	5	5
Accuracy [%]	78.34	79.88	78.01	78.72	77.80	77.56	77.33	83.12	**83.95**	82.76

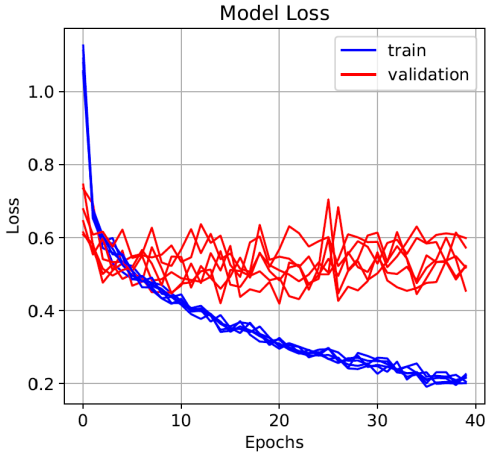
TABLE II: Structure and 6-fold cross validation accuracy of models. Every convolution layer consists of Conv2D with #filters, (kernel size) provided in the table and MaxPooling2D with size (2, 2). Following parameters are the same for all models: picture size 150×150, batch size 32, number of epochs 40, learning rate 0.001, optimizer Adam, activation functions for Conv2D and non-final dense layers: relu, output dense layer activation function: softmax. Normalization regularizes each batch by both mean and variance reference.

V. Optimization

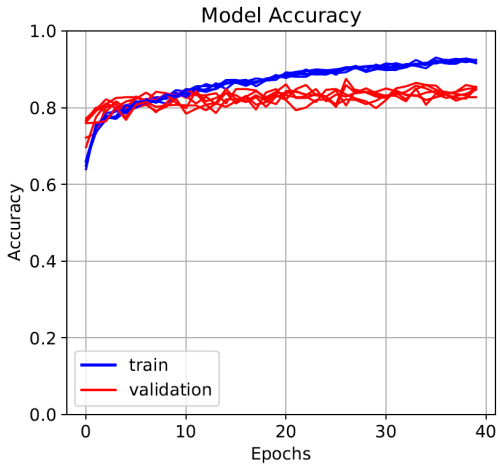
While keeping the structure of the Model 9, that features the pre-trained VGG16 network as a base, together with one additional densely connected layer as an interface into the final softmax layer, we have decided to optimize some of its hyper-parameters and training configurations, see Table III. When inspecting the table, we see, that adjusting the learning rate (α) by an order of magnitude lower or higher had no positive effect on the final cross-validation accuracy. Similarly, replacing the currently used optimizer (Adam) with Stochastic gradient descend (SGD) or Root Mean Square Propagation (RMSprop) did not help either. However, the validation results show a slight increase in accuracy after decreasing batch size from 32 to 16. The progression of the loss function and accuracy of the best model, marked as Model 9.4 is depicted in Fig. 3.

Model	9.1	9.2	9.3	**9.4**	9.5	9.6
α	0.001	0.0001	0.01	0.001	0.001	0.001
Batch size	32	32	32	16	32	32
Optimizer	Adam	Adam	Adam	Adam	SGD	RMSProp
Accuracy [%]	83.95	83.39	81.49	**84.73**	78.01	83.51

TABLE III: Fine-tuning of the best model (Model 9 in Tab. II)



(a) Loss



(b) Accuracy

Fig. 3: Training and validation (a) loss, (b) accuracy for the Model 9.4 (α=0.001, batch_size=16, optimizer: Adam)

VI. Results

Moving on, the selected model (see Tables II and III) was propagated through the whole training set (3371 samples), while applying the same data augmentations techniques as was described. After that, our model was evaluated using the test set – classifying 843 samples from our dataset. The classes were predicted correctly in 85.17% cases, which is a great increase when compared with the accuracy of NN based models – consult Tab. IV. The confusion matrix in Fig. 4 shows, that our model learnt to predict all classes quite well with few misclassified samples in each class. The confusion matrix shows that the tulip is very often mistakenly regarded as roses, and vice versa. This is probably due to the occurrence of many rose-like species of tulip in the database. The problem is also the pictures in which the distinction of the species is also heavy for humans. Two examples of wrong classified flower images are in Fig. 5.

TABLE IV: Metrics of the final model.

	Accuracy	Precision	Recall	F1 score
Project 1: NN	55.04%	52.74%	54.41%	52.95%
Project 2: CNN	**85.17**%	**85.28**%	**85.30**%	**85.11**%

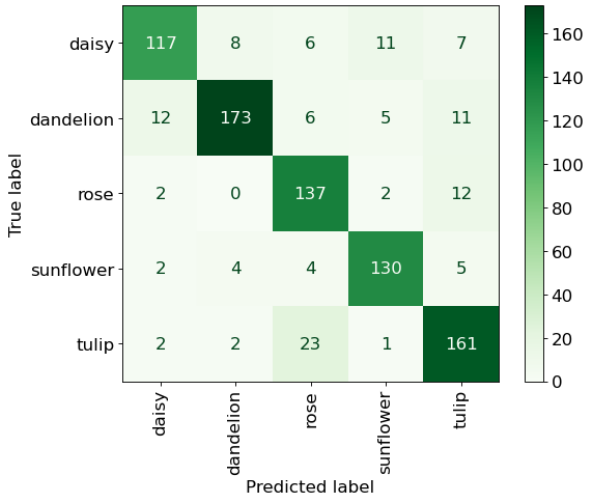


Fig. 4: Confusion matrix.

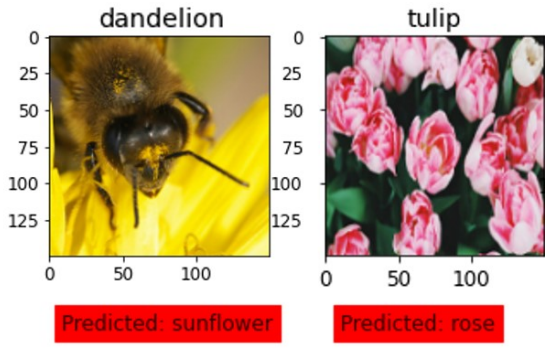


Fig. 5: Examples of wrong classification.

VII. Conclusion

After modelling various CNN for our problem of flower classification, we can state with confidence, that they fulfil the initial assumption and are much more powerful than regular neural networks for image classification as the state of the art suggests. [2] Without data augmentation, we have registered an increase in the overall classification accuracy by more than 25%, whilst with augmentation, we were able to push accuracy even further. We also showed that pre-trained models can be used quite effectively and VGG16 in particular served this purpose very well.

It is worth noting, that the networks used for transfer learning are constantly evolving in the last years. As one recent study suggests, with the right balance of the 3 network dimensions, we can achieve the same or better accuracy much more efficiently than with the most commonly used models for image classification. [10] This observation suggests a good direction and potential expansion of this study in the future.

References

[1] I. Gogul and S. Kumar. Flower species recognition system using convolution neural networks and transfer learning. pages 1–6, 03 2017.

[2] Farhana Sultana, Abu Sufian, and Paramartha Dutta. Advancements in image classification using convolutional neural network. *CoRR*, abs/1905.03288, 2019.

[3] Martin Abadi and et al. Tensorflow : Large-scale machine learning on heterogeneous distributed systems. 01 2015.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[5] M Mathur. CNN's using VGG16 (83% accuracy). 30.04.2020. www.kaggle.com/madz2000/cnn-s-using-vgg16-83-accuracy.

[6] F. Chollet. *Deep Learning with Python*. Manning Publications; 1st edition, 2017.

[7] R. Mehrotra. Flower recognition CNN keras. 21.09.2019. www.kaggle.com/rajmehra03/flower-recognition-cnn-keras.

[8] O. F. Gurbuz. Flowers recognition with custom CNN. 21.09.2019. www.kaggle.com/epiktroll/flowers-recognition-with-custom-cnn.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[10] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.