Flowers classification using feedforward neural network

Machine Learning lectured by Prof. Pétia Georgieva - April 13, 2020

Bartosz Stoń
DETI
University of Aveiro
Wrocław University of Technology

Dominik Raška
DETI
University of Aveiro
VSB - Technical University of Ostrava

***Abstract*—This is a report of a project that aims to explore one of the most significant methods used in machine learning – classification. Our goal was to implement an artificial neural network consisting of regular densely-connected layers that would be able to recognise flowers species based on computer image data. The model uses a database containing 4214 pictures of flowers of 5 different species.**

***Index Terms*—machine learning, neural network, flower classification, multiclass classification, gradient descent**

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] We have decided to build an optimized model based on a feedforward neural network to solve the problem of flower classification. Following this paragraph, we describe the image data collection that was obtained and show how we extracted the features that was supplied to our classification model respectively. Section III describes the algorithms that were used for the training of the model. Part IV is about model optimization based on a number of hyper-parameters. After the selection of the optimal hyper-parameters, our model was retrained and subjected to the final evaluation. From there, we obtained the performance measures that are presented and discussed in the last two sections.

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

- Bartosz Stoń – data preprocessing, model fine-tuning, graph plotting, presentation
- Dominik Raška – k-fold validation, model fine-tuning, confusion matrix

II. Data

Our database that was obtained from Kaggle[1] contains 4323 JPEG photographs of 5 different flower species (classes). Total number of instances of each class are showed in Table IV. Raw images vary in their resolution and aspect ratio. Flowers in these photographs are placed in different backrounds, often with other objects that only increase the difficulty of our classification goal. Therefore, during the cleaning process, we have discarded several instances that could devolve the final performance of our model. Few examples that were kept are depicted in the Figure 1.

A. Preprocessing

All images were resized into a resolution 80×80 in order to represent uniform feature space with the final dimensionality given as follows:

$$80 \times 80 \times 3 = 19200 \tag{1}$$

Individual features that represent the values of RGB channels were divided by 255 to fall in the range $[0, 1]$. One-hot encoding was used to represent data classes as a suitable representation for neural network (NN) training. Finally, the entire dataset was shuffled and split into 2 parts:

1) Training set (80%) – used for training and evaluating of the model. A fraction of the training set (Cross-validation set) is utilized to evaluate the performance of the model over different selections of hyper-parameters.
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

[1]https://www.kaggle.com/alxmamaev/flowers-recognition

(a) Sunflower

(b) Tulip

Fig. 1: Data examples

III. Model implementation

For the sake of implementation efficiency we have decided to use Keras library that runs on top of widely used machine learning platform – TensorFlow. [2] The fixed structure of the model was built upon a seqentual neural network with one or very few densely-connected hidden layers. Parameters of any hidden layer are initilized using a uniform distribution within the range $[0, 1]$. Nodes in the hidden layer are activated using standard ReLU (Rectified linear unit) function. Nodes in the output layer corresponds to the probability of individual classes and their output is smoothened through softmax function. Cost function is then defined as a cross-entropy between the class probabilities obtained from the output layer and the true one-hot encoded values. The gradient of the cost fuction is backpropagated through the network while making use of mini-batch stochastic gradient descent (SGD) optimization algorithm.

IV. Optimalization

In order to find the model with the best hyper-parameters, several models with different initial parameters (number of hidden layers, number of nodes in hidden layers and regularization coefficient (λ) were initialy trained and validated. Resulting accuracies of each run are compared in the Table II. The appropriate learning rate (α) and number of epochs were pre-selected during the initial validation for each model. The number of epochs was determined for each model based on the minimum of the loss function. After that, it was assessed, whether the pre-selected learning rate maximized the accuracy by retraining the Model no. 1 with different learning rates – see Figure 2.

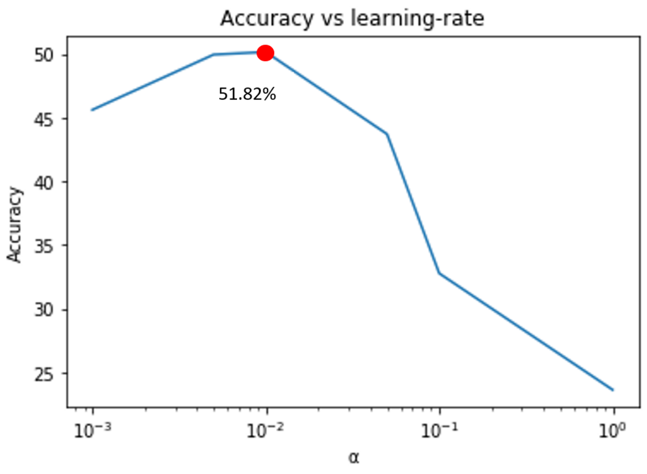
A. *Models 6-fold cross validation*

After initial selections of the learning rate and number of epochs, 6-fold cross validation was performed for each model. The accuracy of each model is shown in the Table II. It can be seen that the accuracy of the models decreases with the number of epochs, but there is no meaningful relationship between the number of nodes in the layers and the accuracy. Furthermore, the accuracy is higher for the regularization parameter $\lambda = 0.003$. Accuracy and loss for training and validation data of the best model (Model 1) are shown in the Figure 4. Significant overfitting can be seen, however, as the table shows, increasing the regularization parameter does not improve accuracy. Usually in case of overfitting reducing of features can be helpful, but there is no clear relationship between the number of nodes in hidden layer and accuracy.

Model	**1**	2	3	4	5	6
α	0.01	0.01	0.01	0.01	0.01	0.01
λ	0.003	0.003	0.003	0.003	0.003	0.003
Hidden layers	1200	2400	2400 300	4800 600	2400 600 300	4800 1200 300
Accuracy [%]	**51.82**	51.53	49.48	51.73	47.22	49.36

Model	7	8	9	10	11	12
α	0.01	0.01	0.01	0.01	0.01	0.01
λ	0.01	0.01	0.01	0.01	0.01	0.01
Hidden layers	1200	2400	2400 300	4800 600	2400 600 300	4800 1200 300
Accuracy [%]	48.29	47.69	43.71	46.68	34.44	36.84

TABLE II: Models accuracy with respect to changing its hyper-parameters (α – learning rate, λ – regularization parameter, hidden layers structure). Picture size 80, batch size 128, initial weights - uniform distribution.



(a)

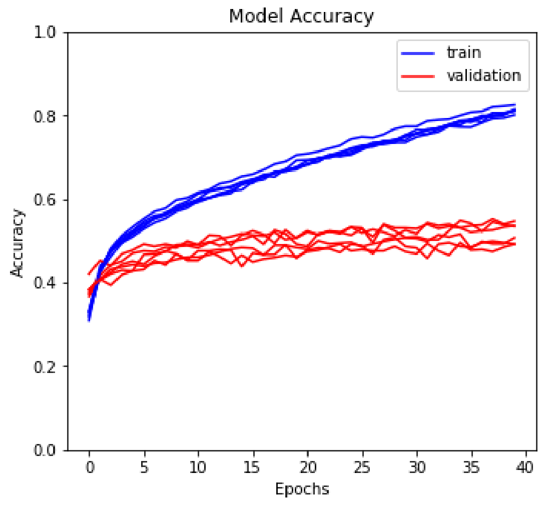
Fig. 2: Accuracy while adjusting learning rate

B. *Aditional models 6-fold cross validation*

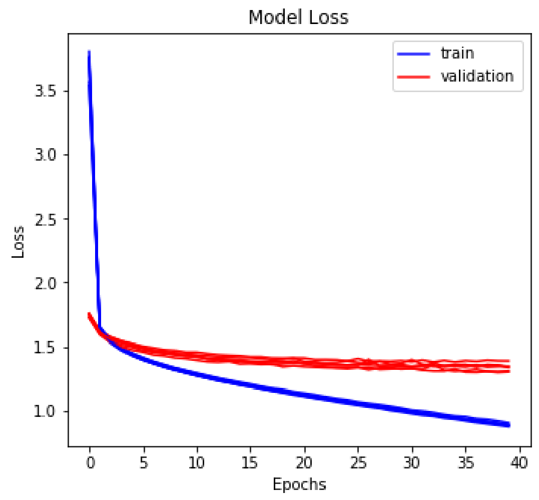
Due to the low accuracy of the previous models, an additional four models were built based on the best of them (Model 1). In each model one parameter was changed in relation to the initial Model 1, see Table III. Model 13 does not contain regularization, Model 14 due to overfiting number of input features – image size was reduced, in Model 15 initializer of initial weights was changed to default in the keras package, while in Model 16 the batch size was reduced. The accuracy of none of the presented models exceeds the accuracy of Model 1.

Model	13	14	15	16
λ	0	0.003	0.003	0.003
Picture size	80	50	80	80
Kernel initialization	uniform	uniform	Glorot-normal	uniform
Batch size	128	128	128	32
Accuracy [%]	46.12	51.52	49.38	51.67

TABLE III: Models accuracy with specific changes. Parameters not listed in this table are the same as in Model 1 from Table II.



(a) Accuracy

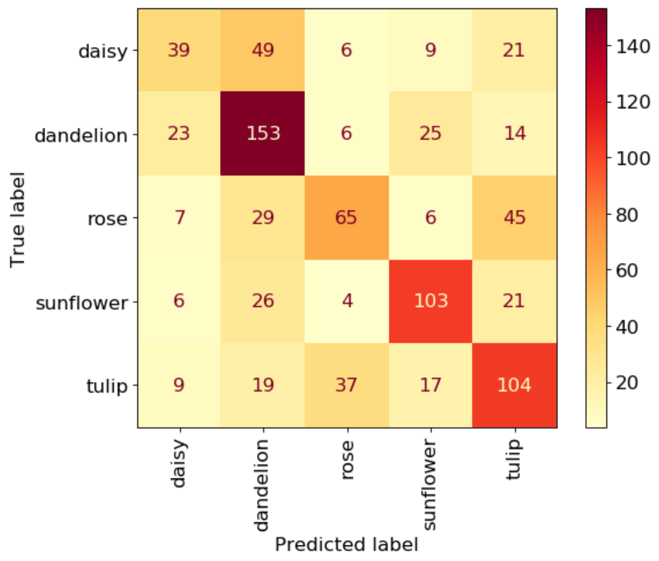


(b) Loss

Fig. 3: Convergence of the best training - Model 1.

V. Results

The best model (Model 1) was retrained on the full training set and then used to classify our test data that contains 843 data instances. The final model accuracy is 55.04% – metrics are provided in the Table IV. This accuracy is higher than the accuracy obtained by cross validation (51.82%), which confirms that the model is overfitted and probably more examples are needed. For a more thorough examination of the result of the models, the confusion matrix is shown in the Fig 4. Except for a slight overrepresentation of dandelions, all 5 categories are approximately equal in size. Analyzing the matrix, it can be seen that the largest values follows the left diagonal of the matrix, which is the desired result. An exception to this is daisy, which was incorrectly classified as a dandelion in most cases. It can be seen that every other flower is also often considered a dandelion (column 2). This may be due to the two forms the dandelion has – a fresh or overblown flower. Very often, tulip gets confused with rose and vice versa – this is probably due to the great similarity of these flowers. Examples of correct and wrong classifications are given in the attachment and in the Fig 5. However, the interpretation of why the given images have been classified into a given category seems often impossible and contradictory to predictions.



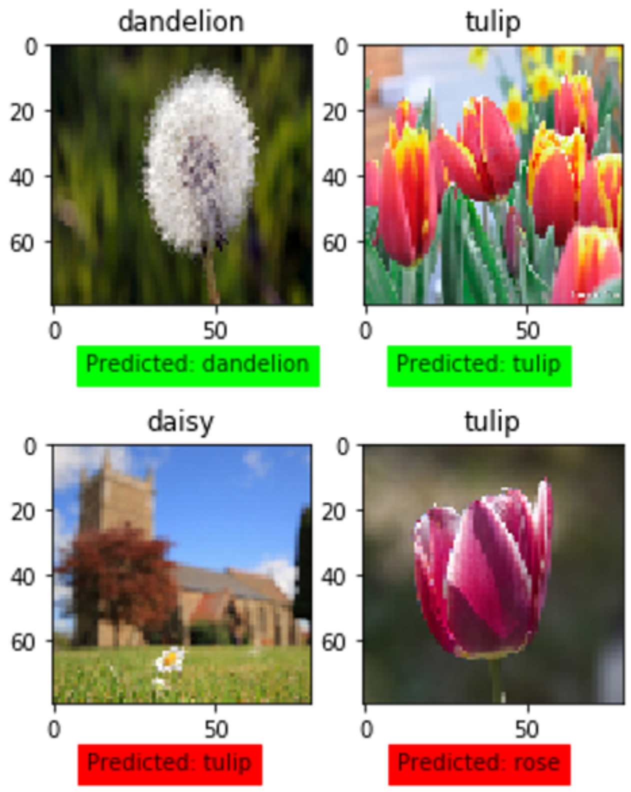
True label \ Predicted label	daisy	dandelion	rose	sunflower	tulip
daisy	39	49	6	9	21
dandelion	23	153	6	25	14
rose	7	29	65	6	45
sunflower	6	26	4	103	21
tulip	9	19	37	17	104

(a)

Fig. 4: Confusion matrix

TABLE IV: Metrics of the final model

Accuracy	Precision	Recall	F1 score
55.04%	52.74%	54.41%	52.95%



(a)

Fig. 5: Examples of correct and wrong classification

VI. Conclusion

Due to the huge number of species, the rarity of some species and the un-unified way of taking pictures, there are no publications containing simple NN models that would deal with this problem. On the other hand, deep learning models, especially convolutional neural networks (CNN), are widely used. [3] Due to a complicated problem, it is probably not possible to obtain better accuracy using only densely-connected NN layers trained on this database. Due to overfitting, accuracy can likely be increased by finding more examples or by data augmentation. Using both augmentation and CNN for this database, it is possible to achieve an accuracy of more than 80% [4], using additionally pretrained model, it is possible to increase the accuracy of the model to more than 94% [5]. As such, the goal of the next project lies in the development of a more complex deep learning model and in applying techniques of data augmentation.

References

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

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

[3] H. Hiary, H Saadeh, and M Saadeh. Flower classification using deep convolutional neural networks. iet computer vision. page 855–862, 2018.

[4] R. Mehrora. https://www.kaggle.com/rajmehra03/flower-recognition-cnn-keras. 13.04.2020.

[5] S. Wadhwa. https://www.kaggle.com/sominwadhwa/flower-recognition-fastai-94-accuracy. 13.04.2020.