IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45AI/2025/4535025/
Vol. 2025 (20), Article ID 4535025

Artificial Intelligence Methods in Flower Species Recognition: Models Testing*

Jakub JANKOWICZ and Joanna WIŚNIEWSKA

Institute of Information Systems, Faculty of Cybernetics,
Military University of Technology, Gen. S. Kaliskiego 2, 00-908 Warsaw, Poland

Correspondence should be addressed to: Jakub JANKOWICZ, jankowicz.jakub@gmail.com

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

The following paper describes the process of designing and implementing machine learning models that allow for the recognition of flower species using images. This problem is significant mainly because of the artificial intelligence development and its capabilities. In this work, we briefly characterize the types of machine learning and the data set on which the experiments were performed. The computational technologies used in the experiment are also described to present the high degree of development of environments supporting programmers in using various machine learning models. Finally, two neural network models were created and tested - one based on the classifier and one on the convolutional neural network.

Keywords: machine learning, neural networks, Python, flowers

Introduction

Nowadays, we can observe more and more applications for artificial intelligence (AI). Currently, it supports us in many aspects of our lives, such as: medicine and helping doctors diagnose diseases or even recommendations when browsing online stores. AI allows us to facilitate our daily activities and sometimes we use it without even being aware that a given device or service uses it. Among lovers of walks in the forest, GPS-based applications for recording the route traveled and programs responsible for recognizing edible and poisonous mushrooms are becoming popular, which is precisely what is done using AI methods, for example in the form of a mobile application (see: Mushroom (2025)).

The aim of this work is to implement machine learning models that allow for the identification of flower species through image recognition. Such programs can be used in the future as commercial mobile applications that allow users to distinguish flower species.

The paper first addresses the issue of machine learning and its types. In the next section, the reader is introduced to the data set on which the experiments were performed. Then, the computational technologies used in the experiment are described, such as the Python language with its libraries and the user-friendly Google Colaboratory environment. Two neural network models were created and tested - based on the classifier and the convolutional neural network. Finally, the results of the obtained learning processes are described.

Cite this Article as: Jakub JANKOWICZ and Joanna WIŚNIEWSKA, Vol. 2025 (20) "Artificial Intelligence Methods in Flower Species Recognition: Models Testing" Communications of International Proceedings, Vol. 2025 (20), Article ID 4535025, https://doi.org/10.5171/2025.4535025

Machine Learning and Artificial Intelligence

Machine learning is a branch of artificial intelligence that deals with creating a system that, by operating on data, will learn and improve itself. The first model using machine learning was programmed by Arthur Samuel in 1952 (see: Mitchell (1997)) and was used to play checkers, or more precisely, to predict which side has a better chance of winning. Based on the achievements of Arthur Samuel and Donald Hebb's 1947 theory on neurons, described in the book by Hebb (1949), the perceptron was created, or more precisely, the machine called "Mark 1 Perceptron", authored by Frank Rosenblatt. Since the 1960s, work has been carried out on inventing algorithms that allow for self-improvement using the entered training data. In 1967, the k-nearest neighbors algorithm was created - responsible for classifying responses based on measures of distance between data samples (e.g. the Euclidean measure).

In the early 1980s, artificial neural networks began to be developed, among others, thanks to the discovery of backpropagation in the 1970s. The development of technology and theory allowed for the wider use of machine learning, for example in the field of board games - the chess-playing program "Deep Blue" played the first match against a grandmaster in 1996. The first duel was won by Garry Kasparov, while a year later in the rematch he had to admit the superiority of the machine with a result of 3½ : 2½ (see: Price (2020)). It was a historic moment when the world champion lost to the program for the first time. The beginning of the 21st century was a breakthrough in terms of the development of neural networks. In 2006, the term deep learning was coined, which uses many layers in neural networks to better recognize images and process natural language. Currently, deep learning is key in the development of artificial intelligence. Machine learning can be used, among others, for data analysis, image recognition, decision-making, prediction. They can be divided into 4 types: supervised, unsupervised, semi-supervised, and reinforcement learning (see: Machine Learning (2025)).

In unsupervised learning, the training dataset used does not contain the answers, so the program uses the data itself, between which it searches for patterns, relationships, and differences. A major advantage of unsupervised learning is the lack of the need to create labels. The biggest disadvantage of this solution is the difficulty of determining the correctness of the results of the trained model due to the lack of labels to which comparisons could be made. An example of a task supported by unsupervised learning is clustering, used for grouping based on similarities and relationships. The k-means algorithm is used here, which divides the data into groups with similar features.

Supervised learning uses a training dataset containing data along with responses. Based on them, the model searches for and learns the relationships between them in order to determine the most accurate response for the test dataset. In supervised learning, we mainly use regression (used to predict continuous values) and classification (used to assign records to previously defined labels).

Semi-supervised learning is characterized by the fact that it has the characteristics of supervised and unsupervised learning. In this case, the training set contains both data with responses and data without them. The model learns from the data with labels and uses them to study the data without labels. Semi-supervised learning is based on self-improvement - the program, having data with labels and without them, learns from the labeled data, then assigns labels to incomplete data, and then learns from the entire data set. An example of a method used in semi-supervised learning is pseudo-labeling, in which the model predicts and assigns pseudo-labels to unlabeled data, and then learns again using the labeled data and the data it has labeled.

Reinforcement learning uses a dataset to which it applies a provided set of rules and expected results. A system of "rewards" for solving problems is also used here - they can be positive and negative, and the model tries to collect as many positive ones as possible. Reinforcement learning is used iteratively - with each training cycle, the program updates its strategy for gaining "rewards" in order to improve. An example of an algorithm used in reinforcement learning is Q-Learning by Li (2023). It is based on adjusting the expected profit from performing actions in specific rules, which is called the Q value.

In this work, two types of neural networks (Aggarwal (2023)), i.e. supervised learning methods for recognizing flower species, were used. Multilayer Perceptron Classifier (MLPClassifier) from the scikit-learn library (see: MLPClassifier (2025)) is a machine learning model that implements an artificial neural network designed for data classification. It is one of the types of neural networks composed of at least one hidden layer, which makes it a so-called multilayer network. The most important features of MLPClassifier:

- a multilayer perceptron, which consists of input layer, one or more hidden layers and output layer,
- uses as activation function, most often ReLU (Rectified Linear Unit) or hyperbolic tangent, in hidden layers,
- learns using the backpropagation algorithm, using e.g. the adam or SGD optimizer (see: MLPClassifier (2025)),
- can be used for binary, multi-class and multi-label classification problems.

The second experiment was performed on a convolutional network from the Tensorflow library (see: Tensorflow (2025)). A Convolutional Neural Network (CNN) is a special type of artificial neural network, designed primarily for analyzing grid-structured data, so it is great for studying images. The most important features of CNN:

- Convolutional layer is a basic element of CNN. Instead of simply multiplying weights like in MLP, CNN uses filters (masks) that are moved over the image.
- After each convolutional layer, a nonlinear function, usually ReLU, is applied to introduce nonlinearity.
- Pooling layer is used to reduce the size of data without losing the most important information.
- Fully connected layers – at the end of the CNN there is usually a classic MLP (such as MLPClassifier), which makes a decision based on the features detected by the previous layers - e.g. classifies the image.

Data Set

According to data from 2016, there are 391 thousand species of vascular plants known in the world, of which 369 thousand are flowering plants by Butler (2023). The greatest plant species diversity is found in: Brazil, China, Colombia, Mexico and South Africa.

Today, according to Butler (2023), the number of known plant species may already be over 400,000, while it is estimated that there may be as many as 100,000 species yet to be discovered, probably most in the equatorial rainforests (Amazon and the forest in the Congo Basin), and in China and Australia.

The dataset used in this work contains images of selected flower species organized in subfolders, whose name represents the class in the model. All files are saved in .jpg format for better clarity and were downloaded from https://pixabay.com. This site is based on a free use license that allows users to download and share content such as images, videos and music for free. The creators can be supported by voluntary donations. The dataset created from the shared files contains 14 classes (species), and the images have different resolutions. The selected flower species are: chrysanthemum, dahlia, gerbera, lily, magnolia, poppy, marigold, narcissus, orchid, peony, rose, sunflower, daisy, tulip.

Utilized Technologies

To create machine learning models, the Google Colab (Colaboratory) environment was used, which is a browser-based document that allows writing and executing Python code provided by Google based on the cloud, see: Colab (2025). The file is divided into cells with code by default, which can be executed separately. The code is run on Google servers in the cloud, so the computer components are not used, and GPU (Graphics Processing Unit) and TPU (Tensor Processing Unit) resources from the cloud are used. The notebooks are ultimately saved to Google Drive, but they can also be exported to .py and .ipynb formats. Additionally, many Python libraries are already installed there, which saves time on preparing the work environment. This environment is most often used for machine learning, data analysis or visualization. It is also possible to share notebooks for real-time collaboration with other users.

In order to import the dataset and run programs in Google Colaboratory, Google Drive was used, which is a cloud space that allows you to store and organize files and data. To open a notebook in the environment, it must be placed in Google Drive along with the dataset folder.

The programming language chosen for implementing machine learning models is Python, see: Matthes (2016), a high-level, multi-paradigm programming language released in 1991. Its creator is Guido van Rossum. It is based on a free license. Python is characterized by clear, readable code that allows efficient writing and quick understanding even for people with little programming experience, and dynamic typing, i.e. assigning types of

variables during program execution. It is available on many operating systems, and offers a large collection of libraries used in various fields.

Several Python libraries were used to create the programs, supporting, among others, model training, image processing, loading the data set into the program, and support for model evaluation methods. The libraries used together with the available APIs are:

- Os – library that allows the application to interact with the operating system, such as file and directory operations. It allows, for example, file path management.
- Numpy – enables efficient array management. It works very well with other libraries such as Tensorflow or Pandas. It is characterized by: handling multidimensional arrays more efficiently than Python lists, support for mathematical functions (e.g. trigonometric, algebraic), support for filtering in arrays, and built-in operations performed on matrices.
- Pillow – is used for image processing. This library has functions such as: opening and saving images, image operations, modifying individual image pixels, applying filters to the image.
- Sklearn – one of the most popular Python libraries used for machine learning. It allows for creating supervised and unsupervised learning models, data operations (scaling, removing missing data), model evaluation.
- Google.colab – allows connecting a notebook in Google Colaboratory to Google Drive in order to load data from it into the model.
- Matplotlib – Python library used for creating graphs and data visualizations. It is one of the most popular tools in this language. It allows you to create 2D and 3D graphs and customize them by adjusting labels, titles, or styles. This library supports various data types coming from other libraries, such as Pandas or Numpy. Graphs can be exported to PDF or PNG formats and can be split into sub-graphs.
- Yellowbrick – used to visualize and evaluate machine learning models. It is designed to work well with Scikit-Learn. It allows you to create such model validation methods as: ROC curve, error matrix. It is easy to use, and the graphs it presents are easy to interpret.
- Tensorflow – serves for machine learning and deep learning. It is one of the most popular libraries for this purpose. It can be used for both simple and advanced ML models. It is scalable and highly efficient in terms of model training and computation. It works well with other libraries, such as Keras, Numpy or Pandas.
- Keras – Python library that is part of the Tensorflow library that allows you to create, train, and evaluate machine learning models. It is easy to use and allows you to quickly create models based on layers that can be added, removed, and modified. It supports various data types: numeric, text, images.

The above tools allowed us to create two machine learning models for flower recognition through image recognition. The first one uses a classifier based on an artificial neural network from the Scikit-Learn library (Model 1). The second model was created on a convolutional neural network used from the Keras library (Model 2) associated with Tensorflow.

Description of the Experiment – Model 1

In the first experiment, the MLPClassifier (Multilayer Perceptron Classifier) neural network from the Sklearn library was used. A set of flowers was loaded as images, and then normalization was performed by dividing the X array by 255, due to the default pixel value of the image, which is from 0 to 255, coming from the format in which the image was saved - RGB. In the next stage, label binarization is performed (LabelBinarizer function). This is to make it easier for the algorithm to recognize and classify classes. For example, having three classes: rose, tulip and daisy, they will be saved as binary numbers. In the presented case, for a rose it would be [1,0,0], for a tulip [0,1,0], and for a daisy [0,0,1]. Then the set was divided into a training and test set in the proportions of 80% and 20%.

Fig. 1 shows the obtained image grid from the training set of the neural network.

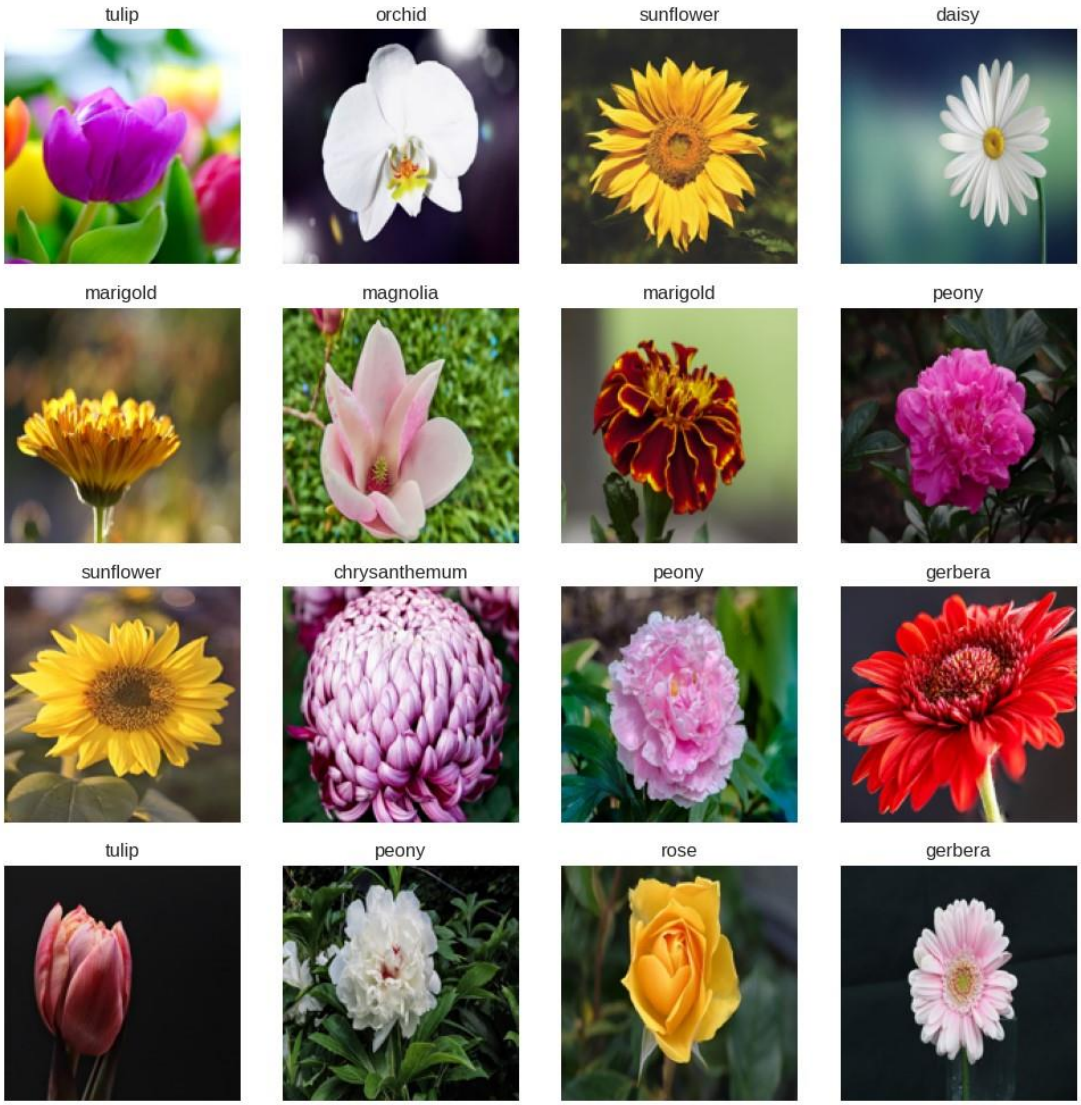

Fig. 1. Model 1 image grid

The model parameter adjustments started with 40 iterations of the training process and a single hidden layer of 512 neurons, and to select the number in the layers successive powers of 2 were used. Initially, the neural network operated on default settings. The first results were very unsatisfactory, so by trial and error, parameters were adjusted, added and removed. The number of hidden layers was gradually increased, and the number of neurons in the first layer was reduced to 256. The maximum number of iterations from the initial one was increased and operated on values between 50 and 100, so that the model could learn correctly while avoiding the effect of overfitting. The "sgd" method was used to update the network weights, but attempts were also made to use the "adam" method. The advice generated by the artificial intelligence built into the environment from the Gemini API was used to adjust the classifier parameters. The random state was set to 42 in order to have a constant network result after each code execution.

The accuracy of the model on the training data is 1.0, which means that the model is doing well with the training itself or the overfitting effect occurs. In the case of removing the "random_state" parameter in the code fragment that defines the structure of the neural network, this accuracy is from about 0.9 to 1.0. It can therefore be concluded that the training process after 60 declared iterations on the training data went correctly. The problem occurs in the case of the test data - in this case, for the "random_state" parameter equal to 42, the accuracy is equal to 0.14, and for random states it is in the range of 0.9 to 0.14.

Another problem of this model is the loss curve, shown in Fig. 2, which does not efficiently approach 0, and at times there are increases in loss, but at the next iteration it drops again. This may indicate that at some moments the neural network encounters problems that make training difficult.

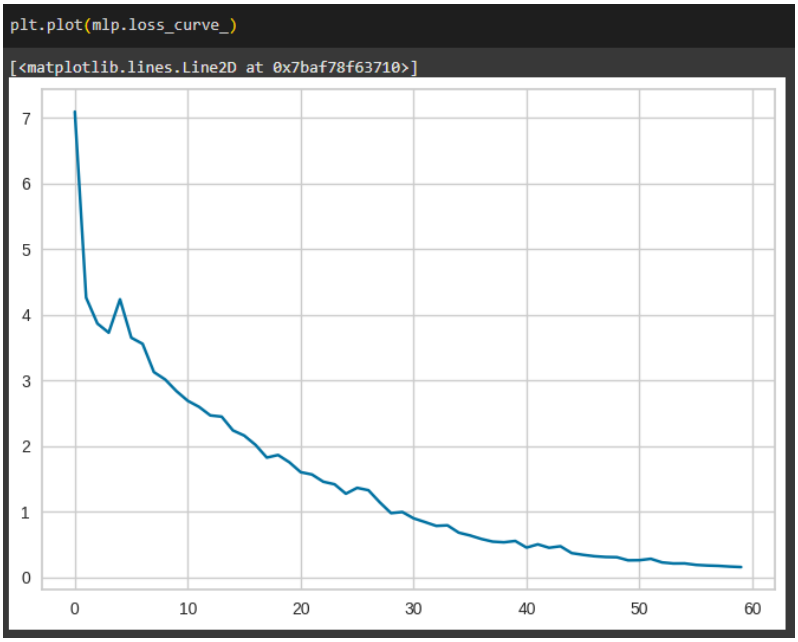

Fig. 2. Loss curve for model 1

The last measure used to evaluate the first program is the confusion matrix. Since the accuracy for the training data is 1.0, the matrix in this case also came out without errors and all labels were correctly assigned. This can be observed by reading the first matrix in Fig. 3. Values higher than zero appear only on its diagonal, which means that no case was misclassified. The second matrix shows the predictions for the test data and only 8 cases were correctly assigned labels.

```
print(confusion_matrix(y_train_single, pred_train_single))
print('\n')
print(confusion_matrix(y_test_single, pred_test_single))
```

```
[[12  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 11  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 10  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 11  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 13  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 14  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  9  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 14  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 10  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 13  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 12  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 13  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 14  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 12]]


[[2 0 0 0 0 0 0 0 0 0 1 0 0 0]
 [3 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [5 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [3 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [2 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [2 0 0 0 0 1 1 0 0 0 0 2 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [3 1 0 0 0 0 0 0 1 0 0 0 0 0]
 [2 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [2 0 0 0 0 0 0 0 0 0 1 0 0 0]
 [1 0 0 0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 1 0]
 [3 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```

Fig. 3. Confusion matrix for model 1

Finally, the network was tested on a test image of a rose. Unfortunately, the program incorrectly predicted the class and assigned the label "poppy". To sum up the program with the classifier, we can say that the neural network is able to learn on training data, but it has trouble distinguishing between test data. This may be due to too small a dataset or too detailed images, which cause the model problems.

Description of the Experiment – Model 2

The second model for image recognition was implemented using a convolutional neural network using the tools provided by the Tensorflow library.

During data loading, labels were encoded as integers, and images were loaded as color (RGB). The training set is 75% of the data set, and the test set is 25%, i.e. in the training set we have 158 images beloning to 14 classes and in the test set 52 images beloning to 14 classes. In the next step, a grid of images from the training set was created to additionally confirm the correct execution of the previous operations (Fig. 4).

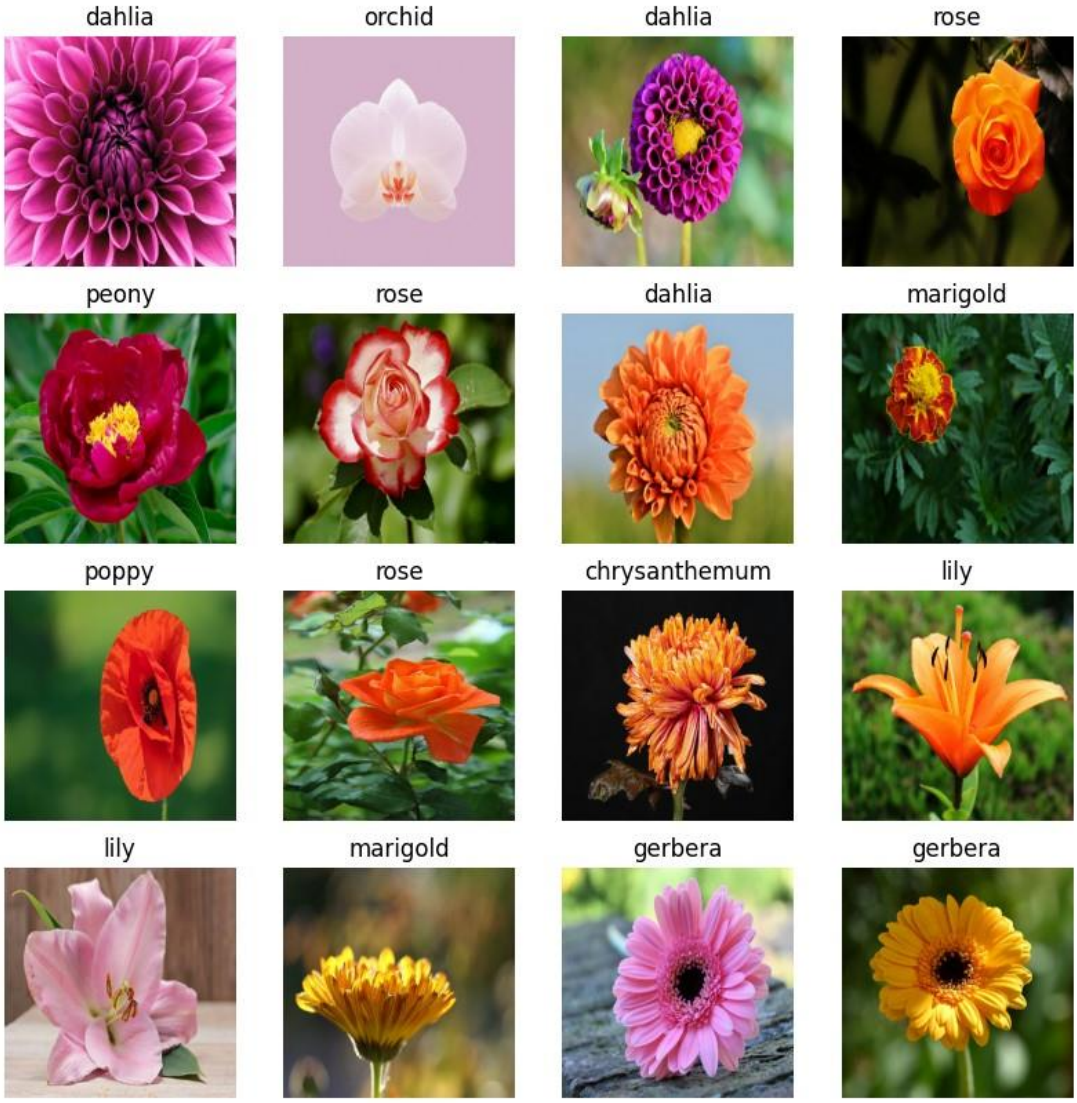

Fig. 4. Model 2 image grid

The next step is to normalize the image data to prepare it for input into the model. Normalization is done by dividing by 255, because pixel values in RGB images usually range from 0 to 255. At the end of this code snippet, the value for the first image is checked. It should be between 0.0 and 1.0. But in this case, it ranges from 0.0 to 0.9978, so the maximum value is slightly lower than expected, but such a small difference is acceptable and it may be the result of imprecise calculations or there is no pixel with the maximum value in this particular image.

Next, a model of a convolutional neural network responsible for image recognition is defined by using the Sequential() function from the Tensorflow library. In this code fragment, subsequent layers are added to the model. Initially, there were few layers and the results were far from satisfactory, but with subsequent tests of the program, they were added and expanded to improve the network's results. The number of neurons in the layers was also adjusted. To determine in which direction the changes to the parameters and layers should be introduced, the Gemini API provided in Google Colaboratory based on artificial intelligence was used. Ultimately, it was possible to achieve results that allowed the network to correctly assign a class to the test image also used in the first model. In this way, it is possible to assess which program performs better. By using the softmax activation function on the output dense layer, the model returns a probability distribution for each class.

Then the model is configured by specifying the optimizer as “adam”, the loss function used for multiclass classification (its parameter from_logits=False means that the classes will be represented as probabilities), and the metrics that have been selected to evaluate the model is the accuracy.

In Fig. 5, a summary of the model architecture generated is presented. We can see columns reflecting: the type of layer, the shape of the output generated by the layers allowing to read how the data in the model changes, and the number of parameters in each layer. Below this table, the numbers of all parameters, trained parameters, and untrained parameters are counted.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
conv2d_3 (Conv2D)	(None, 28, 28, 128)	147,584
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 256)	6,422,784
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 14)	3,598

Total params: 6,667,214 (25.43 MB)
Trainable params: 6,667,214 (25.43 MB)
Non-trainable params: 0 (0.00 B)

Fig. 5. Summary of the model 2 architecture

Then, the model defined and verified in this way was trained. Before training, the number of epochs was defined. Each epoch is a full pass through the entire data set. In this case, 20 epochs were declared. The model was trained on normalized training data. The training process is visible in Fig. 6.

```
Epoch 1/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 49s 8s/step - accuracy: 0.0825 - loss: 2.7580 - val_accuracy: 0.0385 - val_loss: 28.3943
Epoch 2/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 66s 6s/step - accuracy: 0.0948 - loss: 2.6180 - val_accuracy: 0.0962 - val_loss: 73.5355
Epoch 3/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 41s 6s/step - accuracy: 0.1232 - loss: 2.5173 - val_accuracy: 0.1538 - val_loss: 121.5298
Epoch 4/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 29s 6s/step - accuracy: 0.1857 - loss: 2.4046 - val_accuracy: 0.1731 - val_loss: 118.0276
Epoch 5/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 40s 6s/step - accuracy: 0.2534 - loss: 2.2795 - val_accuracy: 0.2115 - val_loss: 118.6239
Epoch 6/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 29s 6s/step - accuracy: 0.3480 - loss: 2.0823 - val_accuracy: 0.1923 - val_loss: 197.8794
Epoch 7/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 42s 6s/step - accuracy: 0.3206 - loss: 2.0518 - val_accuracy: 0.2115 - val_loss: 213.7591
Epoch 8/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 41s 6s/step - accuracy: 0.5513 - loss: 1.4773 - val_accuracy: 0.3269 - val_loss: 264.1113
Epoch 9/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 41s 6s/step - accuracy: 0.6031 - loss: 1.2647 - val_accuracy: 0.3077 - val_loss: 252.0804
Epoch 10/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 40s 6s/step - accuracy: 0.5426 - loss: 1.2942 - val_accuracy: 0.2692 - val_loss: 314.7900
Epoch 11/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 28s 5s/step - accuracy: 0.7507 - loss: 0.8344 - val_accuracy: 0.3462 - val_loss: 490.9810
Epoch 12/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 34s 7s/step - accuracy: 0.7819 - loss: 0.6428 - val_accuracy: 0.3077 - val_loss: 489.0455
Epoch 13/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 30s 6s/step - accuracy: 0.8695 - loss: 0.4884 - val_accuracy: 0.3077 - val_loss: 512.5804
Epoch 14/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 30s 6s/step - accuracy: 0.9210 - loss: 0.2800 - val_accuracy: 0.3654 - val_loss: 610.3901
Epoch 15/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 41s 6s/step - accuracy: 0.9193 - loss: 0.3520 - val_accuracy: 0.3462 - val_loss: 655.1988
Epoch 16/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 30s 6s/step - accuracy: 0.9519 - loss: 0.1839 - val_accuracy: 0.3077 - val_loss: 714.2148
Epoch 17/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 42s 6s/step - accuracy: 0.9375 - loss: 0.2195 - val_accuracy: 0.3846 - val_loss: 742.1302
Epoch 18/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 31s 6s/step - accuracy: 0.9584 - loss: 0.1300 - val_accuracy: 0.2885 - val_loss: 1060.1331
Epoch 19/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 39s 6s/step - accuracy: 0.9534 - loss: 0.1493 - val_accuracy: 0.3077 - val_loss: 877.6271
Epoch 20/20
5/5 ━━━━━━━━━━━━━━━━━━━━ 30s 6s/step - accuracy: 0.9372 - loss: 0.1921 - val_accuracy: 0.2885 - val_loss: 1064.6838
```

Fig. 6. Model 2 training process

The learning effects are visible in the graphs in Fig. 7.

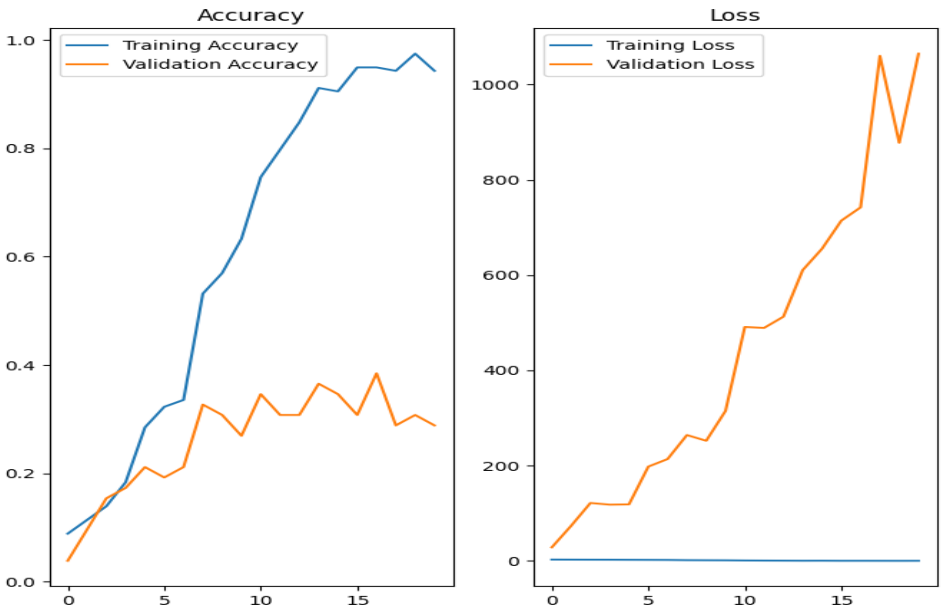

Fig. 7. Accuracy and loss charts

Model 2 based on the convolutional network achieves more satisfactory results compared to the previous one, but some elements of this solution are also problematic. It does not have a steady state, so each run of the program achieves different results. The accuracy of this model on training data is usually around 0.95. On the other hand,

the accuracy for test data remains in the range of around 0.23 to over 0.33, which is a better result than in the case of the first model. The losses during training of the CNN model for training data are low and starting from around 3 they drop to 0, including a slight increase at the very end of the training process. The losses for test data are very high, sometimes approaching 1000.

When testing the model on a test image that was also used to test the first model, it was noticed that in some runs it was possible to correctly predict the class for rose. However, when it assigns an incorrect label, it usually gives one of the four classes: poppy, dahlia, peony or marigold. Again, as in the case of the first model, the problem may be that the dataset is too small or the images in it are too detailed.

In order to check what result the implemented neural networks will achieve when the images in the dataset are less detailed, their quality was reduced from the basic value of 92 to 40. To do this, the free program GIMP was used. The result achieved by the convolutional neural network was similar, but the accuracy of the classifier dropped to 0.02.

Conclusions

This paper describes the implementation and testing process of two neural network models using popular libraries. Both programs train on the dataset of flowers photos. Unfortunately, the first classification model does not cope well with predicting flower species for the test dataset, which may be due to the dataset being too small or the images contained in it being too detailed. The images themselves are probably correctly loaded into the models and processed. The second program does better both with training on the training data and with predicting the flower species for the test image. Incorrect predictions do occur in this model, and the accuracy and losses for the test set are far from ideal. Despite these problems, both neural networks seem to process and learn correctly from the input data.

References

- Aggarwal, C.C. (2023) ‘Neural Networks and Deep Learning: A Textbook’, Springer.
- Butler, R.A. (2023), [Online], [Retrieved May 10, 2025], Total number of plant species by country, https://worldrainforests.com/03plants.htm
- Colab (2025), [Online], [Retrieved May 10, 2025], https://colab.research.google.com/
- Hebb, D.O. (1949) ‘The organization of behavior. A neuropsychological theory’, McGill University.
- Li, S. (2023) ‘Reinforcement Learning for Sequential Decision and Optimal Control’, Springer Verlag, Singapore.
- Machine Learning (2025), [Online], [Retrieved May 10, 2025], https://www.geeksforgeeks.org/machine-learning/
- Matthes, E. (2016) ‘Python Crash Course’, No Starch Press, San Francisco.
- Mitchell, T. (1997) ‘Machine learning’, McGraw-Hill Companies, Inc.
- MLPClassifier (2025), [Online], [Retrieved May 10, 2025], https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
- Mushroom (2025), [Online], [Retrieved May 10, 2025], https://play.google.com/store/apps/details?id=com.pingou.champignouf&hl=pl
- Price, B. (2020) ‘The history of chess in fifty moves’, Quid.
- Tensorflow (2025), [Online], [Retrieved May 10, 2025], https://www.tensorflow.org/tutorials/images/cnn