Image classification through neural networks

Team:
Riccardo Tavecchio 852211
Alfio Leanza 910235
Corso di Laurea Magistrale in Data Science

Anno 2023/2024
Progetto di Deep Learning

Abstract: In this project, we present a comprehensive strategy for the classification of images utilising Convolutional Neural Networks (CNNs). Our primary goal was to accurately categorise a dataset consisting of 3,670 images spanning five distinct flower categories. To achieve this, our project followed an iterative approach to model training, commencing with a baseline model and progressively advancing towards more sophisticated architectures by incorporating data augmentation techniques. This step-by-step progression aimed to optimise the model's performance metrics. To evaluate the model's ability to classify specific types of flowers and its overall capacity for generalisation, extensive testing was conducted at each final phase of every try.

Summary

1 Introduction

This report outlines the process of developing a neural network for flower species recognition, based on a dataset composed of 3,000 images divided into five distinct categories of flowers. The primary goal of the project is to construct a neural network model capable of accurately identifying the genus of a flower given as input, belonging to one of the categories analyzed during the training phase.

Interest in automatic flower species recognition has applications in various fields, from agriculture to botany, and even education and biodiversity conservation. In particular, a well-trained neural network can facilitate the rapid and accurate identification of plants, supporting experts and enthusiasts in the classification process.

The dataset used in this project was downloaded from a pre-existing source and contains images already categorized into their respective flower categories. This allowed us to focus directly on the subsequent phases of the project without having to undertake the complex and time-consuming process of data collection and labeling.

The report describes the following phases of the project:

1. **Dataset Preparation**: Cleaning and dividing the images into subsets for training, validation, and testing.
2. **Theory of Neural Network Architecture**: Theory of neural networks used.
3. **Designing the Neural Network Architecture**: Designing the neural network with a focus on the selection of layers and activation functions and calculation of the model training process, including the optimization and regularization techniques adopted.
4. **Inference**: Analyzing the model's performance based on metrics such as accuracy, precision, recall, and F1-score.

The adoption of machine learning techniques, particularly convolutional neural networks (CNNs), represents a strategic choice for image recognition, leveraging these networks' ability to automatically learn the salient features of images through multiple convolutional layers.

The expected result is a model with high accuracy in recognizing the categories of flowers present in the dataset, thereby contributing to the advancement of visual recognition technologies and their practical applications.

2 Data Preparation

In the data preparation phase, the dimensions of the images throughout the entire dataset were assessed. This step is crucial to understand the scale and variability of the images, which can inform subsequent preprocessing steps and model architecture choices.

Dimension Analysis:

The dimensions of each image within the dataset were examined to gain insights into the overall composition of the dataset. This involved traversing through the entire dataset and extracting the width and height of each image

Results:

- **Number of images in the dataset**: 3670
- **Number of unique image dimensions in the dataset**: 345
- **Average height of images in the dataset**: 271.79
- **Average width of images in the dataset**: 365.06

The most frequent dimensions in our images are:

1. (500, 333) with 635 images
2. (320, 240) with 618 images
3. (320, 213) with 331 images
4. (240, 240) with 208 images

The analysis revealed the scale and variety of images present in the dataset, laying the groundwork for subsequent data preprocessing and model development stages. This comprehensive understanding of the dataset's characteristics is essential for building an effective and robust flower species recognition model.

2.1 Training and Validation set:

As the last stage of data preparation, the dataset was divided into two parts: training (80 percent) and validation (20 percent). This split is essential to train the model on one subset of the data and validate it on another, ensuring that the model generalizes well to data not seen during training.

To achieve this, we set shuffle=True to ensure that the images were taken randomly, and we selected a batch size of 32.

3 Theory of Neural Network Architecture

This section will outline the main theoretical concepts that have been used for the current analysis.

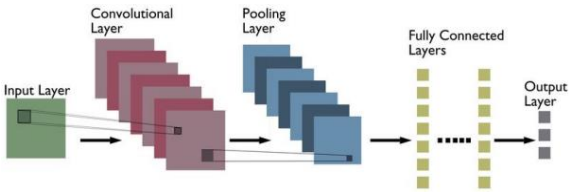
3.1 CNN

CNNs (also called ConvNets) are Neural Networks that use convolution in place of general matrix multiplication in at least one layer.

A typical CNN has 4 layers:

1. Input layer
2. Convolution layer
   - Affine transform (convolution)
   - Detector (Nonlinearity)
3. Pooling layer
4. Fully connected layer

Figure 3.1.1: Layers of a CNN



Source: Handouts from Milano Bicocca university (faculty: Data Science)

3.1.1 Input layer

The input layer in a convolutional neural network (CNN) is the starting point through which raw data is introduced into the model. In other words, this layer represents the first stage of the information processing pipeline.

The input layer receives data in their raw form, usually images. These images are represented as arrays of pixels. For example, a grayscale image of size 28x28 pixels is represented as a matrix of size 28x28, where each element of the matrix corresponds to the intensity of one pixel (from 0 for black to 255 for white).

In the case of color images, the data is represented by three channels (red, green and blue), so the image is represented as a tensor of dimensions height x width x depth. For example, an RGB image of size 32x32 pixels is represented as a tensor of size 32x32x3.

The input layer does not perform operations on the data, but simply prepares it to be processed by subsequent layers. The typical format of the input data is a vector or tensor, which is compatible with the convolution and pooling operations that occur in subsequent layers of the network.

In summary, the input layer of a CNN is responsible for receiving and correctly representing the raw data in a form that can be further processed by subsequent convolutional and pooling layers. This layer is crucial because it ensures that the data is in the correct format for processing within the neural network.

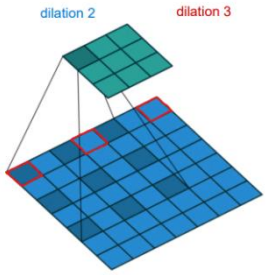
3.1.2 Convolutional layer

The **convolution** operation is composed of multiple filters (kernels), the filters for 2D images are also two-dimensional (2D).

Parameters:

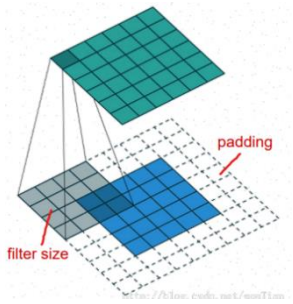
- Filter size: larger filters to be avoided. Odd sized filters are preferred to even sized filters
- Padding: to not alter the spatial dimensions of the input. We pad input in every direction with 0's before applying filter.
- Stride: the number of pixels we move filter to the right/down
- Dilation: to have a larger receptive field

Figure 3.1.2.1: graphical display of dilation parameter



Source: Handouts from Milano Bicocca university (faculty: Data Science)

Figure 3.1.2.2: graphical display of filter size and padding parameter



Source: Handouts from Milano Bicocca university (faculty: Data Science)

Activation function introduce non-linearity after filter is applied to the whole feature map.

Preferred activation function in CNN is ReLU that leaves outputs with positive values as is, replaces negative values with 0.

3.1.3 Pooling layer

Pooling is a key-step in convolutional based systems that reduces the dimensionality of the feature maps. It combines a set of values into a smaller number of values, i.e., the reduction in the dimensionality of the feature map.

3.1.4 Fully connected layer

Neurons in a fully connected layer have full connections to all activations in the previous layer. Their activations can hence be computed with a matrix multiplication followed by a bias offset. The last fully-connected layer holds the output, such as the class scores.

Any time we wish to represent a probability distribution over a discrete variable with n possible values, we may use the softmax function. Softmax functions are most often used as the output of a classifier, to represent the probability distribution over n different classes

3.1.5 Input normalization

Batch Normalization

Batch Normalization (BN) is a technique used to improve the training of deep neural networks by normalizing the inputs of each layer to have zero mean and unit variance. This helps stabilize and accelerate the training process.

3.2 Regularization

Many strategies used in machine learning are explicitly designed to reduce the test error, possibly at the expense of increased training error. The strategies are known collectively as regularization.

3.2.1 Weight decay

L2 regularization strategy drives the weights closer to the origin by adding a regularization term to the objective function.

$$\Omega(\vartheta) = \frac{1}{2}||w||_2^2$$

L1 regularization (also known as Lasso) on the model parameter w is defined as:

$$\Omega(\vartheta) = ||w||_1 = \sum_i |w_i|$$

that is, as the sum of absolute values of the individual parameters.

3.2.3 Early stopping

Monitors the model's performance on the validation set during training and stops training when the performance stops improving. This helps to avoid overfitting by stopping at the point of optimal generalization.

3.2.4 Evaluation

Performance measures calculated from the confusion matrix entries:

- Accuracy: Proportion of correct forecasts to total forecasts made
- Sensitivity: true positive rate
- Specificity: true negative rate
- F-measure: A measure that combines precision and recall is the harmonic mean of precision and recall. It reaches its best value at 1 (perfect precision and recall) and worst at 0. Note, however, that the F-measures do not take the true negatives into account

3.3 Data Augmentation

To build effective Deep Learning models, it is essential that the validation error continues to decrease along with the training error. Data augmentation is a very powerful technique to achieve this goal.

Augmented data creates a more complete set of possible data points, thereby minimizing the distance between the training set and the validation set, as well as any future test set. Augmented data addresses overfitting at the root of the problem.

The idea is that, through data augmentation, more information can be extracted from the original data set.

4 Designing the Neural Network Architecture

This section will present the analyses conducted in order to answer the research question posed initially.

4.1 First Model

We start from a simple architecture to see the trend and then we'll adjust it to reach better results. First, we define the input layer. Our network accepts images of size 271x365 pixels, each with three colour channels (red, green, blue). This input dimension is specified at the beginning of the model.

The first layer of the network is a convolutional layer that applies 32 filters, each with a size of 3x3. The padding is set to ‘same’ to ensure that the spatial dimensions of the output are the same as the input. This convolutional layer is followed by a ReLU activation function, which introduces non-linearity into the model, allowing more complex patterns to be captured in the images.

Next, we apply a Max Pooling layer with a 3x3 window and a stride of 3. This layer reduces the spatial dimensions of the images, thus decreasing the number of parameters and computational complexity. Additionally, Max Pooling makes the network more robust to variations in the position of objects in the images.

After the first pooling layer, we add a second convolutional layer with 64 filters, also of size 3x3 and with "same" padding. This layer is also followed by a ReLU activation. Increasing the number of filters enables the network to extract more sophisticated and abstract features from the images.

To further reduce the data dimensions without losing important information, we use a Global Max Pooling layer. This layer reduces each feature map to a single value, retaining the most relevant feature for each map.

Finally, we reach the output layer, which is a dense layer with a number of neurons equal to the number of classes in our classification problem. We use a softmax activation function to obtain a probability distribution over the classes, allowing the model to make probabilistic predictions for each class.

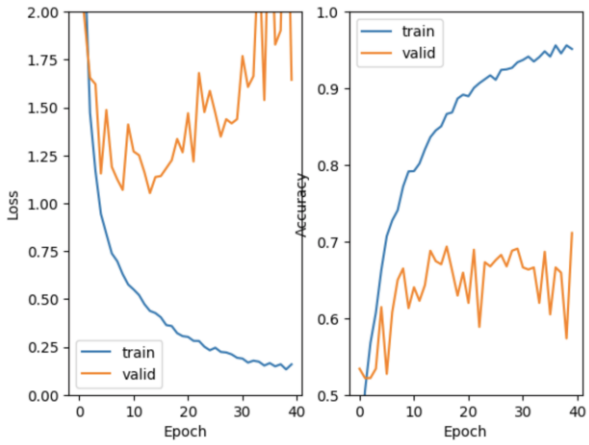
Once the model architecture is defined, we proceed with its compilation. We use the categorical crossentropy loss function, ideal for multi-class classification problems. For optimization, we choose the RMSprop algorithm with a learning rate of 0.001, known for its effectiveness in training deep neural networks. Lastly, accuracy is used as a metric to evaluate the model's performance during training.

The model is trained using a training dataset for 40 epochs. During training, we monitor the loss and accuracy on both the training and validation datasets, allowing us to assess the model's progress and prevent overfitting.

After the training part we proceed with the evaluation to see if the model is able to generalise well. The validation loss indicates how far the model deviates from the actual class values, while the validation accuracy represents the percentage of correctly classified images.

Model 1	
Validation Loss	1.6443
Validation Accuracy	0.7112

Figure 4.1.1: Model 1 loss vs epoch and accuracy vs epoch.



Validation loss is significantly higher than training loss and validation accuracy is lower than training accuracy, this could be a sign of overfitting. Additional regularisation techniques, as l2 regularization may be necessary.

4.2 Enhanced Model 1

In this model, based on the first one, we'll add regularization and batch normalization to avoid overfitting.

In particular we add L2 regularization with a factor of 0.1 to the kernel to penalize large weights and prevent overfitting. Next, we add a Batch Normalization layer, which normalizes the output of the previous convolutional layer, improving stability and convergence speed during training.

In the second convolutional layer, again applying L2 regularization with a factor of 0.1 to the kernel.

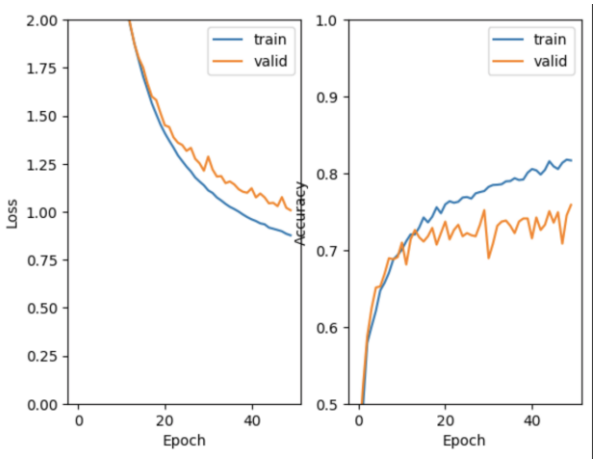
Also in the output layer we apply L2 regularization with a factor of 0.001.

To have more stability on the learning history we reduce the learning rate to 0.0001

After training, we evaluate the model on the validation dataset to compare its performance with the previous model.

Model 2	
Validation Loss	1.0074
Validation Accuracy	0.7589

Figure 4.2.1: Model 2 loss vs epoch and accuracy vs epoch.



4.3 Enhanced Model 2

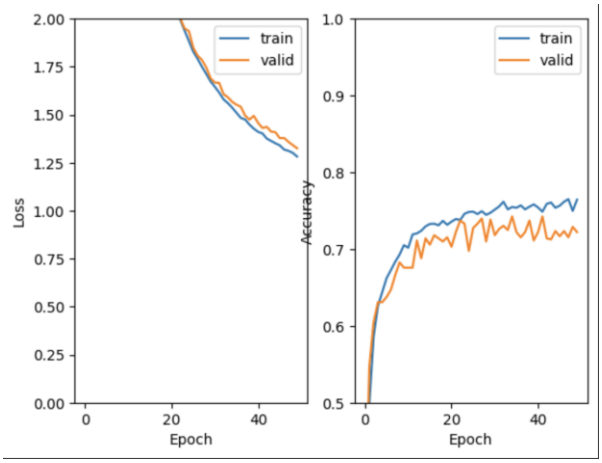
Seeing the results before, we decide to construct a deeper model in order to try to gain the maximum that our observations can give.

To further improve the model's ability to extract features, compared to the previous ones we add a third convolutional layer with 64 filters of size 3x3, applying L2 regularisation and a ReLU activation function, but without Batch Normalisation in this layer.

In addition, we add the early stopping callback to monitor the validation loss.

Model 3	
Validation Loss	1.325
Validation Accuracy	0.7221

Figure 4.3.1: Model 3 loss vs epoch and accuracy vs epoch.



As we see from the values of the evaluation and from the plots, adding more layers lead us to worse results. So this mean that we have few observations and construct deeper model lead to worse model, thus we try to add images by using data augmentation technique.

4.4 Model with data augmentation

This model builds on previous architectures, integrating data augmentation techniques to improve the robustness and generalisation capability of the model.

To begin, we initialize a different training dataset with a size of 290x390, because later we'll apply a random crop layer to return to the original size.

The input layer is defined to accept images of any size, allowing flexibility in processing augmented images. Data augmentation layers are applied directly to the input images, including a random crop to reduce the images to a size of 271x365 pixels, a random horizontal flip with a 50% probability, and a random contrast adjustment within a range of ±25%.

Following data augmentation, the network architecture consists of several convolutional layers with increasing filter sizes and L2 regularization to prevent overfitting. Thus we have a model with four convolutional blocks and different neurons. After these

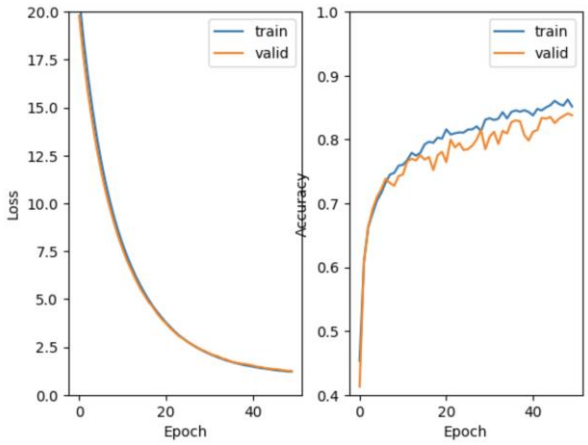
convolutional blocks, a global max pooling layer reduces each feature map to a single value, followed by a dense layer.

A 5-epoch early stopping callback is also used here. The model is trained on the augmented dataset for 50 epochs.

Model evaluation:

Model 4	
Validation Loss	1.2456
Validation Accuracy	0.8379

Figure 4.4.1: Model 4 loss vs epoch and accuracy vs epoch



To follow the goal of our project, we utilised data augmentation to greatly improve our outcomes. The structure of our model evolved to include deep pooling layers, which played a crucial role in capturing the most important characteristics for our classification objective.

One of the most significant enhancements we made was a modification to the activation function. We made the switch from utilising "relu" to "selu" due to the former's susceptibility to the vanishing gradient problem. In particular the activation function “selu” enjoys a self-normalising capability.

As a result, we observed a substantial improvement in the model's performance, characterised by faster convergence and a more dependable training process.

5 Inference

Below are all the performance values of the models used:

Table 5.1: Inference on model 1

Classification report Model 1:				
	P	R	F	S
1	0.16	0.15	0.15	125
2	0.25	0.26	0.26	197
3	0.15	0.16	0.15	107
4	0.19	0.20	0.19	138
5	0.20	0.17	0.19	167
Accuracy			0.20	734
macro avg	0.19	0.19	0.19	734
weighted avg	0.20	0.20	0.20	734

P=precison R=recall F=f1-score S=support

Table 5.2: Inference on model 2

Classification report Model 2:				
	P	R	F	S
1	0.18	0.16	0.17	125
2	0.27	0.29	0.28	197
3	0.18	0.19	0.18	107
4	0.16	0.16	0.16	138
5	0.27	0.26	0.26	167
Accuracy			0.22	734
macro avg	0.21	0.21	0.21	734
weighted avg	0.22	0.22	0.22	734

P=precison R=recall F=f1-score S=support

Table 5.3: Inference on model 3

Classification report Model 3:				
	P	R	F	S
1	0.20	0.17	0.18	125
2	0.26	0.28	0.27	197
3	0.18	0.16	0.17	107
4	0.14	0.16	0.15	138
5	0.23	0.23	0.23	167
Accuracy			0.21	734

macro avg	0.20	0.20	0.20	734
weighted avg	0.21	0.21	0.21	734

P=precison R=recall F=f1-score S=support

Table 5.4: Inference on model 4

Classification report Model 4:				
	P	R	F	S
1	0.19	0.17	0.18	125
2	0.27	0.27	0.27	197
3	0.12	0.14	0.13	107
4	0.18	0.20	0.19	138
5	0.30	0.27	0.28	167
Accuracy			0.22	734
macro avg	0.21	0.21	0.21	734
weighted avg	0.22	0.22	0.22	734

P=precison R=recall F=f1-score S=support

During the initial three reports, data augmentation was not implemented in the training phase. As a result, these models demonstrate poor precision, recall, and f1-scores for each flower category. The accuracy of these models varies from 20% to 22%, suggesting a moderate ability to accurately classify different types of flowers. Both the macro and weighted averages for precision, recall, and f1-score consistently hover around 20%, indicating a consistently low performance across all classes.

It is worth mentioning that the second and third reports exhibit a slight enhancement in performance measurements when compared to the initial report. Specifically, the second model demonstrates a slight boost in recall and f1-score, particularly for class 1 and class 4. This pattern implies that making slight adjustments or fine-tuning the neural network's structure or hyperparameters may have been employed, resulting in slightly improved outcomes.

The fourth classification report presents the performance of a neural network model trained with data augmentation. This model exhibits a more prominent enhancement in its performance metrics. When compared to the first three models, there is a slight elevation in precision, recall, and f1-score for the majority of the classes. While the overall accuracy remains at 22%, the individual class metrics showcase a more equitable performance. Notably, Class 4 displays a significant improvement in precision and f1-score, indicating that the utilization of data augmentation has contributed to the model's improved ability to generalize, specifically for certain types of flowers.

Upon comparing these reports, it becomes clear that the inclusion of data augmentation has a discernible effect on the performance of the model, albeit to a moderate extent. The introduction of greater variability in the training data is likely to assist the model in effectively capturing the various characteristics present in images of flowers, resulting in enhanced accuracy in classification and more equitable metrics across all classes.

This comparison underscores the importance of data augmentation in training robust neural networks for image classification tasks, particularly when dealing with limited datasets.

5.1 Conclusion

To recap, although the classification of flower types proved to be a challenge for all four models, the incorporation of data augmentation in the fourth model resulted in a modest yet significant enhancement in the evaluation metrics. This finding holds great importance in terms of refining model training techniques and enhancing the precision and applicability of neural network classifiers in forthcoming studies.

Moving forward, it is crucial to concentrate on refining data augmentation methods, delving into more sophisticated neural network structures, and potentially expanding the size and variety of the training dataset to attain

greater accuracy and more consistent classification outcomes. This investigation underscores the significance of robust training strategies in the development of successful neural network models for image classification assignments.