As part of the Machine Learning module, a neural network for object recognition has been developed. During this presentation, as described in the slide, it will be presented the process of developing a CNN model, including data loading and preprocessing, its architecture, the summary of the model, and the compilation and training processes. Following the results of the model's training, the model evaluation process will be described and the relevant results. Finally, conclusions and references used for this assignment will be presented.

It is obvious that more and more artificial intelligence technologies are embedded in our daily life. Lately, computer vision and especially deep vision models are on the spotlight, given their multiple uses. Examples where such models are widely used include Google image search, YouTube, and robotics.

Now we will go through the process of developing a neural network for object recognition. For this analysis, we used and loaded the CIFAR-10 dataset that includes 60.000 32x32 colour images in 10 different classes, with 6.000 images per class. Normalisation of data was applied to streamline the scale of the dataset. That process was done by dividing each pixel value of the images by 255 and it turn each image to range from 0 to 1. It was also applied one-hot encoding so as the training and evaluation metrices to be understandable and interpretable. During this process the categorical class labels were transformed to a binary matrix representation. Finally, as part of the data preprocessing, the data was split for training, evaluation, and testing, to ensure the effectiveness of the testing approach.

Our CNN architecture was designed for image classification and in practice what it does is to extract features from images, reduce their dimensionality, and perform classification using fully connected layers. More specifically, our CNN has 2 convolutional layers which apply a set of learnable filters to the input images. Each filter extracts features such as textures, lines, edges from the image and then it creates feature maps. In our model, the first layer has 32 filters, and it was applied to the input image, and the second layer has 64 filters and it was applied to the output of the first layer. To introduce non-linearity in our model, after each convolutional layer a ReLU is used as activation function. This activation function was selected because it is a well-known function for complex pattern learning, avoiding at the same time gradient problem. In addition, to reduce the spatial dimensions of the feature maps created, we apply 2 max pooling layers of 2x2. Such approach contributes to down sample the data and in turn increases the network's effectiveness and robustness. Then, flattening was applied to transform the multi-dimensional feature maps to a single vector. Following this step, dense was applied to perform the final classification. During this process, 2 fully connected layers were used in order the model to learn the relationships between different features and at the end assign probabilities to each class. The first layer has 64 neurons and as in case of convolutional layers, ReLU activation was used, and the second has 10 neurons linked to the 10 classes of our dataset and softmax activation function was used. Softmax was selected on the basis that it is an activation function that it is suitable for multi-class classification problems. Below the process followed is depicted.

In this slide, a layer-by-layer breakdown of the model is depicted. Very briefly, our model has 502.688 parameters in total. Out of these, 167.562 parameters are trainable, and the rest are non-trainable. On top of that, the optimizer used during training has 335.126 parameters.

Going in more details, after having developed the architecture of the model and before training, a compilation of the model was performed to configure the learning process. During this process, "Adam" optimiser was selected to manage how the model updates its weights based on the data it sees each time. The selection of this optimiser was done since it automatically adjusts the learning rate for each parameter and its computational efficiency. Additionally, to measure the model's predictions against the actual target values, "categorical_crossentropy" was selected as loss function considering that this loss function is appropriate for such multiclass-classification problems. Finally, to assess the model's performance, accuracy was selected as a metric. Precision and recall were not added as performance metrics, as it was considered that accuracy would be an adequate metric by itself.

Following the model's compilation, the training process began. During this learning process, 10 epochs were used, meaning that the model was trained to the entire training dataset 10 times. The number of epochs was considered adequate to avoid either underfitting or overfitting. In addition, during its epoch, 1.250 samples were processed before the model updates its weights. As it is depicted in the graphs, the training accuracy of the model was 78.45%.

Following the training process, the model was evaluated on the held-out test dataset. As it is depicted in the graph, its evaluation accuracy was 69.14%.

To summarize, during this assignment, a CNN model for image classification was developed, trained, and evaluated. Following all the steps of such process the model was able to classify accurately more than 69% of the cases when tested in an unseen dataset revealing its generalization capabilities. Perhaps, more advanced techniques, such as data augmentation or use of pre-trained models could enhance the accuracy of the model and could be considered for further analysis.

References

Chollet, F. (2021). *Deep Learning with Python .* Manning Publications.

Gupta, A. (2023) *A comprehensive guide on Optimizers in deep learning*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/ (Accessed: 16 January 2025).

Sharma, S. (2022) *Activation functions in neural networks*, *Medium*. Available at: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (Accessed:16 January 2025).

VanderPlas, J. (2022). *Python Data Science handbook.* O'Reilly Media, Inc.