

# Assignment 2

---

Surname Name

April 25, 2022

In this assignment you are asked to:

1. Implement a neural network to classify images from the CIFAR10 dataset;
2. Implement a fully connected feed forward neural network to classify images from the CIFAR10 dataset.

Both requests are very similar to what we have seen during the labs. However, you are required to follow **exactly** the assignment's specifications.

## 1 FOLLOW OUR RECIPE

Implement a multi-class classifier to identify the subject of the images from **CIFAR-10** data set. To simplify the problem, we restrict the classes to 3: airplane, automobile and bird.

1. Download and load CIFAR-10 dataset using the following **function**, and consider only the first three classes. Check `src/utils.py`, there is already a function for this!
2. Preprocess the data:
  - Normalize each pixel of each channel so that the range is  $[0, 1]$ ;
  - Create one-hot encoding of the labels.
3. Build a neural network with the following architecture:
  - Convolutional layer, with 8 filters of size  $5 \times 5$ , stride of  $1 \times 1$ , and ReLU activation;
  - Max pooling layer, with pooling size of  $2 \times 2$ ;
  - Convolutional layer, with 16 filters of size  $3 \times 3$ , stride of  $2 \times 2$ , and ReLU activation;

- Average pooling layer, with pooling size of  $2 \times 2$ ;
  - Layer to convert the 2D feature maps to vectors (Flatten layer);
  - Dense layer with 8 neurons and tanh activation;
  - Dense output layer with softmax activation;
4. Train the model on the training set from point 1 for 500 epochs:
    - Use the RMSprop optimization algorithm, with a learning rate of 0.003 and a batch size of 128;
    - Use categorical cross-entropy as a loss function;
    - Implement early stopping, monitoring the validation accuracy of the model with a patience of 10 epochs and use 20% of the training data as validation set;
    - When early stopping kicks in, and the training procedure stops, restore the best model found during training.
  5. Draw a plot with epochs on the  $x$ -axis and with two graphs: the train accuracy and the validation accuracy (remember to add a legend to distinguish the two graphs!).
  6. Assess the performances of the network on the test set loaded in point 1, and provide an estimate of the classification accuracy that you expect on new and unseen images.
  7. **Bonus** (Optional) Tune the learning rate and the number of neurons in the last dense hidden layer with a **grid search** to improve the performances (if feasible).
    - Consider the following options for the two hyper-parameters (4 models in total):
      - learning rate: [0.01, 0.0001]
      - number of neurons: [16, 64]
    - Keep all the other hyper-parameters as in point 3.
    - Perform a grid search on the chosen ranges based on hold-out cross-validation in the training set and identify the most promising hyper-parameter setup.
    - Compare the accuracy on the test set achieved by the most promising configuration with that of the model obtained in point 4. Are the accuracy levels statistically different?

## 2 IMAGE CLASSIFICATION WITH FULLY CONNECTED FEED FORWARD NEURAL NETWORKS

In this task, we will try and build a classifier for the first 3 classes of the CIFAR10 dataset. This time, however, we will not use a Convolutional Neural Network, but a classic Feed Forward Neural Network instead.

1. Follow steps 1 and 2 from T1 to prepare the data.
2. Flatten the images into 1D vectors. You can achieve that by using `tf.reshape` or by prepending a **Flatten layer** to your architecture; if you follow this approach this layer will not count for the rules at point 3.
3. Build a Feed Forward Neural Network of your choice, following these constraints:
  - Use only Dense layers.
  - Use no more than 3 layers, considering also the output one.
  - Use ReLU activation for all layers other than the output one.
  - Use Softmax activation for the output layer.
4. Follow step 4 of T1 to train the model.
5. Follow steps 5 and 6 of T1 to assess performance.
6. Qualitatively compare the results obtained in T1 with the ones obtained in T2. Explain what you think the motivations for the difference in performance may be.
7. **Bonus** (Optional) Train your architecture of choice (you are allowed to change the input layer dimensionality!) following the same procedure as above, but, instead of the flattened images, use any feature of your choice as input. You can think of these extracted features as a conceptual equivalent of the Polynomial Features you saw in Regression problems, where the input data were 1D vectors. Remember that images are just 3D tensors (HxWxC) where the first two dimensions are the Height and Width of the image and the last dimension represents the channels (usually 3 for RGB images, one for red, one for green and one for blue). You can compute functions of these data as you would for any multi-dimensional array. A few examples of features that can be extracted from images are:
  - Mean and variance over the whole image.
  - Mean and variance for each channel.
  - Max and min values over the whole image.
  - Max and min values for each channel.
  - Ratios between statistics of different channels (e.g. Max Red / Max Blue)
  - **Image Histogram** (Can be compute directly on **TF Tensors** or by temporarily converting to numpy arrays and using `np.histogram`)

But you can use anything that you think may carry useful information to classify an image.

**N.B.** If you carry out point 7 also consider the obtained model and results in the discussion of point 6.