Assignment 2

Surname Name

May 9, 2023

In this assignment you are asked to:

1. Implement a fully connected feed forward neural network to classify images from the **Cats of the Wild** dataset.
2. Implement a convolutional neural network to classify images of **Cats of the Wild** 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 Image Classification with Fully Connected Feed Forward Neural Networks (FFNN)

In this task, you will try and build a classifier for the provided dataset. This task, you will build a classic Feed Forward Neural Network.

1. Download and load the dataset using the following link. The dataset consists of 7 classes with a folder for each class image. The classes are 'CHEETAH', 'OCELOT', 'SNOW LEOPARD', 'CARACAL', 'LIONS', 'PUMA', 'TIGER'. Check 'src/utils.py' to find the ready and implemented function to load the dataset.
2. Preprocess the data: normalize each pixel of each channel so that the range is [0, 1].
3. 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.
4. 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.

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 and comment on the performances of the network on your test set, and provide an estimate of the classification accuracy that you expect on new and unseen images.

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 temporarely 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.

2 Image Classification with Convolutional Neural Networks (CNN)

Implement a multi-class classifier (CNN model) to identify the class of the images: 'CHEETAH' ,'OCELOT', 'SNOW LEOPARD', 'CARACAL', 'LIONS', 'PUMA', 'TIGER'.

1. Follow steps 1 and 2 from T1 to prepare the data.
2. Build a CNN of your choice, following these constraints:
   - use 3 convolutional layers.
   - use 3 pooling layers.
   - use 3 dense layers (output layer included).
3. Train and validate your model. Choose the right optimizer and loss function.
4. Follow steps 5 and 6 of T1 to assess performance.
5. Qualitatively and **statistically** 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.
6. Apply image manipulation/augmentation techniques in order to improve the performance of your models. Evaluate the performance of the model using the new images and compare the results with the previous evaluation performed in part 3. Provide your observations and insights.
7. **Bonus** (Optional) Tune the model hyper-parameters with a **grid search** to improve the performances (if feasible).
   - 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?