Machine Learning
Università della Svizzera italiana

Assignment 2

Papageorgiou Marina

June 8, 2022

1 Structure of Assignment 2

Generally in my folder ml-21-22/`assignment2/`

1. deliverable/
    - `run_task1.py`
    - `run_task2.py`
    - `nn_task1.h5`
    - `nn_task2.h5`
    - `training_validation_accuracy1.png`
    - `training_validation_accuracy2.png`
2. src/
    - file1.py
    - file2.py
    - `utils.py`
3. `report_Papageorgiou_Marina.pdf`

2 Implementation of Task 1 (Convolutional Neural Network - CNN)

In this Task, I have to implement a Convolutional Neural Network (CNN) to classify images from `CIFAR-10` data set. `CIFAR-10` is a collection of 10 classes of images (airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck), which all of them consists of 60000 32x32 colour images, with 6000 images per class. Also, exist 50000 training images and 10000 test images. But, in this assignment it is asked for me to take into consideration only 3 classes (airplane, automobile and bird).

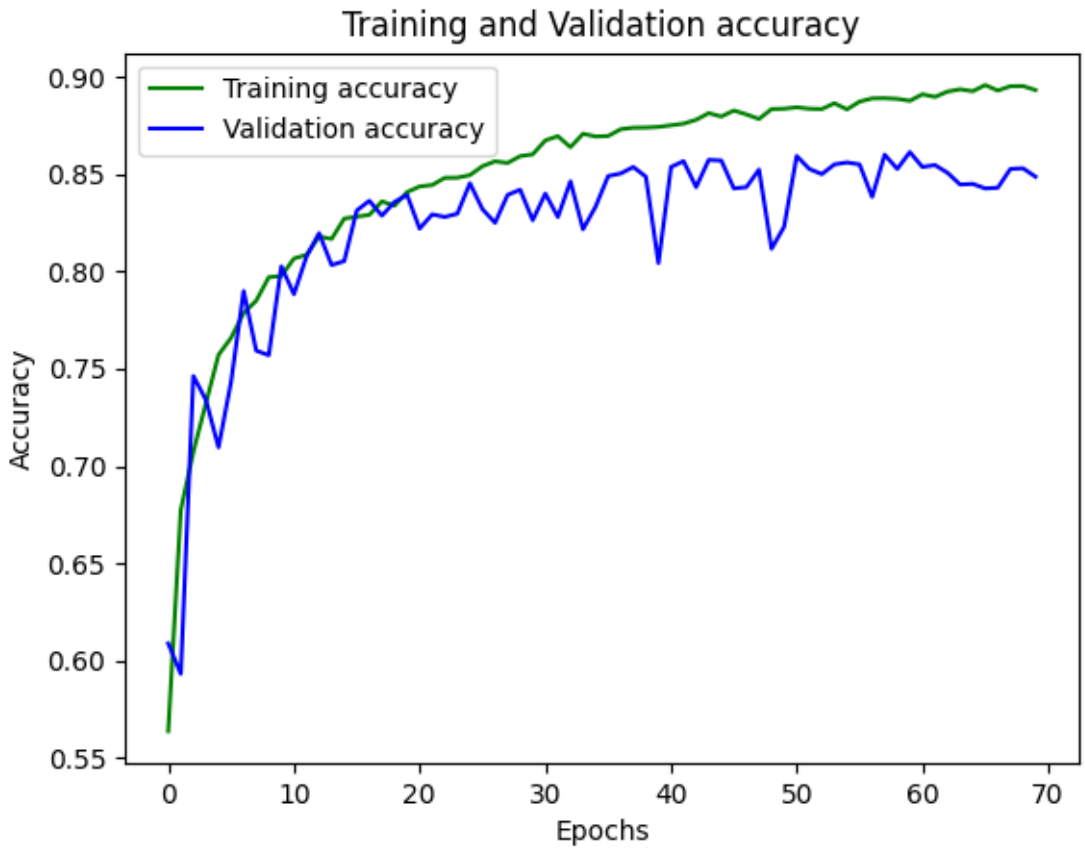
First of all, for the Task 1 I am implementing assignment2/src/file1.py. From that file, I am taking an image with the name `training_validation_accuracy1`, which image is a drawing plot with epochs on the x-axis and with two graphs: the train accuracy and the validation accuracy. After that, at the end of my code I am saving my data to the file `nn_task1.h5`.

At the beginning of my code, at `STEP_1` I am importing all the appropriate libraries that I am using in my code below. After that, I had to load my data and specifically the 3 first classes of `CIFAR-10` that I will use at that assignment (that's why I am writting `load_cifar10(3)`, having the 3 as a parameter). At `STEP_2`, I am normalizing each pixel of each channel so that the range is [0, 1]. But to do that, firstly I am converting from integers to floats my `train_norm` and my `test_norm`, because afterwards I will devide my `train_norm` and my `test_norm` with the float 255.0, which is the maximum value of each channel(Red, Green, Blue). At `STEP_3`, I am defining my model, which means that I am building a Convolutional Neural Network (CNN) with the architecture that the pdf is telling me. So I am doing that, adding each time at my model the order that I have to. Additionally, in `STEP_4` I am training the model on the training set using: the RMSprop optimization algorithm, categorical cross-entropy as a loss function and early stopping. The graph in `STEP_5`, which I mentioned before `training_validation_accuracy1.png`, it is a plot with epochs on the x-axis and with two graphs: the train accuracy and the validation accuracy. At the `STEP_6` for having assess the performances of the network on the test set loaded in point 1, and provide an estimate of the classification accuracy that I expect on new and unseen images, I have to evaluate my train test - running my model. After predicting some results with the test set, I can see an accuracy of X (the accuracy of current model is: 0.8543333411216736), which means we can expect this accuracy on unseen images as this set of images was not used for training, thus the model never saw it. At the last part, which is also the Bonus am having 2 hyper-parameters(4 models in total): learning rate = [0.01, 0.0001] and number of neurons = [16, 64]. I am taking them by two as couple with all the possible combinations(that's why 4 models in total). Inside 2 for loops i am trying to take all the possible combinations that I told above and after taking them I am fitting the new model again with my new requests. But as I can see below from my results the accuracy's scores are not better, so I am choosing the default model (because it's accuracy has bigger value than the accuracies of the combinations that I mentioned above). Again after that I am evaluating - running my train test and print the data.
After, all this procedure my code is printing in my terminal this:

```
Default Model:
94/94 [==============================] - 0s 3ms/step - loss: 0.3825 - accuracy: 0.8543
The accuracy of current model is: 0.8543333411216736
```

```
Learning Rate 0.01 and 16 Neurons:
94/94 [==============================] - 0s 3ms/step - loss: 0.4115 - accuracy: 0.8387
The accuracy of lr+0.01 and neurons=16 is: 0.8386666774749756
Learning Rate 0.01 and 64 Neurons:
94/94 [==============================] - 0s 3ms/step - loss: 0.4440 - accuracy: 0.8190
The accuracy of lr+0.01 and neurons=64 is: 0.8190000057220459
Learning Rate 0.0001 and 16 Neurons:
94/94 [==============================] - 0s 3ms/step - loss: 0.4673 - accuracy: 0.8123
The accuracy of lr+0.0001 and neurons=16 is: 0.812333345413208
Learning Rate 0.0001 and 64 Neurons:
94/94 [==============================] - 0s 4ms/step - loss: 0.4359 - accuracy: 0.8240
The accuracy of lr+0.0001 and neurons=64 is: 0.8240000009536743
```

My graph for the default model is the following one:



Finally, I am saving all those data in the file as I mentioned above `nn_task1.h5`.

3 Implementation of Task 2 (Feed Forward Neural Network - FFNN)

In this task, I am trying and building a classifier for the first 3 classes of the `CIFAR10` dataset. This time, however, I am not using a Convolutional Neural Network, but a classic Feed Forward Neural Network (FFNN) instead.

First of all, for the Task 2 I am implementing assignment2/src/file2.py. From that file, I am taking an image with the name `training_validation_accuracy2`, which image is a drawing plot with epochs on the x-axis and with two graphs: the train accuracy and the validation accuracy. After that, at the end of my code I am saving my data to the file `nn_task2.h5`.

At the beginning, I am adding all the appropriate libraries that I am using below to the code. After that, at `STEP_1` I am keeping the same `STEP_1` and `STEP_2` of Task 1. These steps are the use of `CIFAR10` dataset and the normalization of each pixel and the creating one-hot encoding of the labels, that I already explained above at Task 1. At `STEP_2` and at `STEP_3`, I am combining them (in my code), so I am changing the definition of the model by using the Flatten layer to my architecture and the activation of ReLU and Softmax (with some more restrictions). Next at `STEP_4` I am compiling the model and with the use of Early Stopping I am trying to fit my new model. Moreover, at the `STEP_5`, I am having a plot for my new model with epochs on the x-axis and with two graphs: the train accuracy and the validation accuracy and as a result I am taking the image `training_validation_accuracy2.png`. Also, at the next step I am evaluating - running my train test. Generally, I have obtained that in my Convolutional Neural Network (CNN), cifar10 dataset worked very efficient, but in my Feed Forward Neural Network (FFNN) was not very successful. So, I decided to make some changes to my values and to see if I will have better results. So, indeed after the changes that I did my results was much better that before. After all these changes, the image `training_validation_accuracy2.png`, which I am taking is not as good the results as I was expecting. So, before I had those lines of my code like this:

```
model.add(Dense(3, activation="relu"))
model.add(Dense(3, activation="relu"))
model.add(Dense(3, activation="softmax"))
```

but after trying many values at the first argument of the Dense function I ended up to the conclusion that the first argument of the Dense function must have this form:

```
model.add(Dense(64, activation="relu"))
model.add(Dense(16, activation="relu"))
model.add(Dense(3, activation="softmax"))
```

Like that my graph has better functionality, than before. Before the last value, my result was around 0.33 and now after the last value is 0.76, which is more than 0.75 and my approach is good.

My graph is the following one:

Finally, I am saving all those data in the file as I mentioned above `nn_task2.h5`.