DiC Assignment – 2022

The goal of the assignment is to develop a data processing application using Docker, Python and Tensorflow and understanding the mechanisms behind computation and data offloading.

Assignment

The assignment consists of four parts:

Implementation of data processing application

Application should be implemented inside the `detection_loop` function of the `app.py` provided in the attached .zip archive. Students can choose one between two possible applications:

- **Object detection**: the application takes as input an image and returns the objects that has been detected on the image.
- **Speech recognition**: the application takes as input audio files and recognizes a set of keywords inside the audio files

Both input audio and images datasets are provided to the students. As reference for their implementation, students can use official tensorflow documentation. More precisely,

Object detection: https://www.tensorflow.org/hub/tutorials/object_detection

Speech recognition: https://www.tensorflow.org/tutorials/audio/simple_audio

Application should be at least commented on the main parts. We want to be sure that you understand what you are doing.

Dockerization of application

Developed application should be dockerized, i.e., deployed inside a Docker container. Students are encouraged to use the Dockerfile attached, which sets up the basics service needed for the execution on Docker. Please note that you might need to include more python packages, according to the design of your implementation. To add new packages, it is recommended to add them in the requirements.txt file. For additional information, we advise to read the attached Docker tutorial, which explains how your dockerized application is supposed to work. Additional resources can be found in the official Docker documentation.

Collect data on local and remote execution

Once you deployed your container with your application running, execute your application on the provided datasets and collect data about the **average inference time** of your implementation. By Inference time, we mean the time it takes for the application to perform the required object/speech recognition. If not already provided by tensorflow methods, you can obtain this value for a single image by using additional python modules, i.e., `time` or `timeit`. Average should be calculated on the whole images on the provided datasets. Finally, your containerized app should be deployed on the Hadoop TU Wien Cluster. Once it is deployed, students should collect data about (1) the time required to transfer each image/audio file in the dataset and (2) the average inference time on the cluster.

Comment on execution

Students are expected to provide a report on the developed application and on local and remote execution. In this final report, we expect students to comment these data, i.e., explaining whether it

is worthy to offload data to the remote cluster or not, and in which conditions it makes sense to execute locally or to offload. Conclusion should be supported by data you collected.

Evaluation

At the end of their assignment, students are expected to deliver an archive containing:

- Their Dockerfile and requirements.txt;
- Their app.py implementation including all additional files (if any) you implemented to make it work;
- A final Report, in .doc or .pdf format.

Evaluation will be mainly performed based on the quality of the provided report. The report should contain:

1. Information on how application has been developed, explaining the most important design choices and any modification to the Dockerfile or requirements.txt needed to make your implementation work;
2. Commands needed to deploy your container;
3. Explanation on how you calculate inference and transfer time;
4. Information on your experimental setup. i.e., CPU power, network bandwidth and any additional hardware (GPU) that you use to run your experiments;
5. Data about local/remote inference time and transfer time;
6. Comments on your execution: it is worth to offload execution on the remote cluster? If not, why? What would be needed to improve performance of remote and local execution? Can you think of a scenario where offloading improves performance?

Grading will be performed based on the following scheme:

Implementation	**40**
Dockerization	**10**
Local/Remote Execution	**20**
Report	**30**
TOTAL	**100**