ARCathon

Submission Rules & Instructions

Submission Rules

Beginning on December 19, 2022, you are allowed to submit your solutions for evaluation twice per calendar week until December 31, 2022. Please submit your Docker image as a direct download link (e.g., WeTransfer, Google Drive, or your preferred platform) via the corresponding **submission form**: Submission Page. Please read this document carefully before starting your first submission.

- **General format**: The general format for submissions is Docker. Dockers have the advantage that they are portable and can be run on any virtual machine. It uses virtualization at the operating system level to deploy software in packages called containers. All code submitted to ARCathon must conform to the format below and be submitted as a Docker image!

- **Docker Image**: You must submit your solutions as a download link to a tarred Docker image. In the submission form you are required to provide a corresponding load as well as run command. Examples (see below for details):

    - Save docker image: “docker save IMAGE > /path/to/file.tar”
    - Load docker image: “docker load < file.tar”
    - Docker run command:
      “docker run --mount type=bind,source="$(pwd)"/secret_data,target=/data IMAGE”

- **Evaluation:** Your docker image will be evaluated on a Google Cloud server with a container-optimized operating system. This means that there are almost no restrictions on the Docker image if you are not using a GPU. If you are using a GPU, specify this in the submission and find out about the limitations below. New Google Cloud users get a free starter credit to check if the container runs correctly before submission.

- **Task input:** The secret test tasks are stored locally on the server and accessed by mounting the corresponding folder. Therefore, the execution command you provide should include the directory within your container where the data directory containing the evaluation tasks is to be mounted. The data directory contains the secret test set in the /secret_data/ folder. Here is an example of the mount part of the Docker run command:

  Command: `mount type=bind,source="$(pwd)"/secret_data, target=/data`

  In this example, the /secret_data/ directory that contains the secret evaluation tasks is mounted in the Docker container under /data/. The /secret_data/ directory contains an /evaluation/ folder with the secret evaluation tasks you want to solve and a /solution/ folder where you want to store your solution file (see below).

- **Solution output:** When running the Docker image, a JSON file named "solution_teamid.json" must be output in the /solution/ folder of the mounted folder. The file's name consists of "solution" followed by an underscore and your team ID (same as your team name) provided during registration.

When running the image, it should indicate whether it is still running by, for example, outputting how many tasks have already been solved, and by indicating when the program has finished (e.g., by displaying the message "Done!").

- **Structure of solution file:** The solution file must be a valid JSON file which you can test for validity on this website: Jsonlint. Furthermore, it should have the following structure:
  - It is an array of solutions for each task that are given as objects:
    [{object_task_1}, …., {object_task_n}]
  - The solution task objects in the list above have two key-value pairs:
    - "task_name": "filename"
      (The value is the filename of the task without ".json")
    - "test": [object_solution_1, object_solution_2]
      (The value is a list of the solutions to the tests of the task - some tasks have two inputs and two output solutions that have to be predicted)
    - The objects in the list with the key "test" have the following key-value pairs:
      - "output_id": 0 or 1
        (Depending on input number (first "test" task is 0, the second one is 1)
      - "number_of_predictions": 0, 1, 2 or 3
        (Max. 3 predictions per output are allowed)
      - "predictions": [object_prediction_1, object_prediction_2, object_prediction_3]
        (List of objects containing predictions – max. 3)
    - The objects in the list with key "predictions" have the following key-value pairs
      - "prediction_id": 0, 1, 2 or 3
        (Number of predictions, the order does not matter)
      - "output": [[…], …]
        (Your solution to the test input as an array)

- **Example file:** An example with two solutions for tasks of the public training set – spaces, tabs, etc., were used for improved readability – can be downloaded on the Competition Guide page or directly with this Link: Example File (solution_teamid.json).

Evaluation Boundaries

- **CPU and RAM:**
  - The solutions will be evaluated on a Google N1-standard-4 series server:
    - 4 vCPU's, 15GB RAM, 312 GB storage
    - Max. 1 Nvidia T4 GPU; 16 GB RAM
- **Maximal runtime:**
  - 24 hours without GPU.
  - 5 hours with GPU.
- **Internet and models:**
  - No internet access.
  - All external data – publicly available or not – including pre-trained models, are allowed.

Training and Public Test Sets

- The training and public evaluation sets can be downloaded on the ARCathon Website or with the following link: ARC (800 tasks). See the structure of ARC json files on our ARC Page.
- The secret evaluation set has the same form as the public one – except that the test outputs are replaced by trivial entries, like an array of zeros.

Docker basis container suggestions

- Python users can start with the official python docker container: https://hub.docker.com/_/python
- If a GPU is needed, one can start with the following Container suggested by Google: https://hub.docker.com/r/nvidia/cuda/tags/
- In general: If the Container runs on the above-mentioned Google VMs, they are accepted.

Further information

Please download the Python Docker Tutorial for more information about containerizing Python code with Docker and refer to arcathon@lab42.global in case you have technical or any other questions.