DSAID Video Analytics (VA)

DSAID-VA develops and deploys AI-enabled video analytics products and solutions, to address agency use cases that require images and videos. The work involves AI model development, training and operationalization, software development and developing CI/CD pipelines.

Assessment Format: This take-home assessment involves successfully containerizing a model, serving it as an API, and writing a simple frontend to interact with the API.

Assessment Details

Background: As an AI engineer working in the VA team, you are expected to know how to make the best use of open-source models. For convenience, we will be using the following GitHub repo: https://github.com/CASIA-IVA-Lab/FastSAM.

First, clone the repo and get the code working, either on your local machine or any online service of your choice. You may refer to Inference.py for this part, but the model should be hotloaded for the next portion, i.e. it API should not call Inference.py for each call, but rather the model should be loaded once the server is live.

Next, wrap the model as an API function. You can use either Flask or FastAPI to implement the API functions. You should implement 2 API functions exposed through **port 4000**:

a. **ping** (to ping if the service is alive and running)
   - the response should be in JSON serialised format with key = 'message' and value = 'pong'
b. **infer** (to post the image for inference and receive the inference result)
   - The image should be passed to the call using the ‘files’ parameter in a POST request as a binary with key = ‘image’ and value = <binary value of the image file> . The binary value of the image file can be obtained by function *open(test_image_path.jpg, “rb”)* or any other equivalent functions
   - The mode of inference (everything, box, text, points) should be passed to the call using the ‘data’ parameter in a POST request with key = ‘mode’ and value = one of [everything, box, text, points]
     - If mode = everything, no additional parameters are needed
     - If mode = box, provide the box in the ‘data’ parameter with key = box_prompt and value = [x_top_left, y_top_left, x_bottom_right, y_bottom_right] of the bounding box, with values normalized to between [0, 1] and top left of the image being (0,0)
     - If mode = text, provide the prompt in the ‘data’ parameter with key = text_prompt and value = ‘the black dog’ for e.g.
     - If mode = points, provide the points in the ‘data’ parameter with two key:value pairs
       - key = point_prompt and value = [[x,y],[x,y]], with values normalized to between [0, 1] and top left of the image being (0,0)
       - key = point_label and value = [1,0] a list of labels matching the list of points above, with the first item in the point_label_list being the label for the first point in the point_prompt_list
   - Input parameters should be validated using tools such as Pydantic

Bonus 1: Write unit test for the API using test framework like Pytest.

Bonus 2: Optimise the model (e.g. hotload, TensorRT) to improve latency.

Then, containerize the API service using Docker container. **The Docker file along with the python code is to be zipped and submitted for evaluation**.

Finally, write a simple frontend that allows for image upload (displayed upon successful upload), mode selection, and the relevant information needed:

- Box: A text field allowing 4 comma-separated integers as input. When 4 integers are detected, draw the bounding box on the uploaded image that was previously displayed.
- Text: A text field allowing any valid string
- Points: Two text fields per point, the first for the point (2 comma-separated integers) and the second for the label. Upon both fields receiving valid input, draw the point on the uploaded image that was previously displayed. Have a button below the two text fields to add more points. Points which have the same label should be of the same colour.

When the frontend receives the result, it should display the mask on the original image.

How to submit your answers:

- Submit the link to the Google Drive/One Drive/DropBox or any other online storage service that holds the following submissions:
    1. Zip file of the model weights (.pth, .pt, .h5 or .hdf5) and any relevant artefacts required to run the model
    2. Requirement.txt dependency file using *pip freeze > requirements.txt*
    3. Python code for the API functions
    4. Docker file to run the model API service
    5. Text file to show the command line to run the Docker container
    6. Python code for the frontend
    7. Text file to show the command line to run the frontend

Please submit the link to the above-mentioned artefacts when you complete your assessment via email to howard@dsaid.gov.sg.

Deadline:

- Please complete the assessment **within 7 calendar days** upon receiving this email. Thereafter, you might be invited for a chat with our tech assessors to discuss your submission.

Grading Criteria / Matric:

You will be assessed on

- Good coding practices
- Readability and code cleanliness

Additional Notes to Candidates:

- For questions relating to the assessment, please contact howard@dsaid.gov.sg.