

As shown in the diagram above, I will be deploying my model in a Docker image running inside of a Virtual Machine (VM) inside of Google Cloud Platform (GCP) Compute Engine. Once the Dockerfile is run, it creates a Docker image which contains a Python script that launches the model through Starlette (rather than Flask, which is said to be quite slow in deploying ML models through some basic research). The app contains a basic UI created with HTML that allows the user to upload an image (PNG, JPG) or a DICOM file as data, and also returns the predictions. This GUI is preferable to a command line interface (CLI) since uploading picture files is quite tedious that way.

The container is produced from a Docker file, so that the entire solution may be moved to a different machine by simply moving the ‘rsna-model’ folder and the Dockerfile. The ‘rsna-model’ folder contains the weights and structure of the trained RSNA RNN model as well as the base CNN feature extractor in TensorFlow model format. I decided to deploy onto a cloud platform as it is the easiest way to make my model available to everyone, as VMs are easily accessible through their external IP address. I also chose GCP since I am the most familiar with it, as compared to Paperspace or AWS. Also, I found it easier to create the Docker container directly within the VM, rather than creating it on a local machine and deploying to the VM: this mostly involves the use of a GPU to run the model, as my local laptop does not have a CUDA-enabled GPU.

I will not be able to retrain the model, as it is too costly to automate. It requires a powerful GPU and lots of memory to do so, which is outside of my budget. In fact, I will need a GPU just to pass data through the model for predictions, and the absolute cheapest GPU available is estimated to cost $0.28 hourly ($204.12 monthly). I would also need at least 4GB of memory to load the model into memory, load the input data, and make predictions. To monitor the system, it is easy enough to use GCP’s Logs and Monitoring to detect any potential errors. To fix any bugs that show up, a developer must SSH into the VM.

Pre-deployment checklist:

* Ensure the VM has sufficient processing power and memory to:
  + Hold the model in memory
  + Hold a reasonable amount of data in memory
  + Make a prediction in a reasonable amount of time.
* Ensure that requirements.txt are valid (a short test run should be sufficient)
* Ensure compatibility with both image files (JPG, PNG) and DICOM files.