Serving HuggingFaceTransformer PyTorch models with a PyTorch prebuilt container on Vertex AI Model Registry and Endpoints

Author: Daniel Elias

Firstly, get familiar with the following resources on how to import any PyTorch model to Vertex AI Model Registry and how to deploy it to Vertex AI Online Prediction

- https://cloud.google.com/blog/products/ai-machine-learning/prebuilt-containers-with-pytorch-and-vertex-ai
- https://cloud.google.com/blog/topics/developers-practitioners/pytorch-google-cloud-how-deploy-pytorch-models-vertex-ai

The same steps apply for PyTorch Transformer models (e.g. roBERTa) obtained from HuggingFace.

In summary, there will be 3 steps you will need to follow:

1. Step 1 - Package your PyTorch model into a .mar file with PyTorch Model Archiver
2. Step 2 - Import the model to Vertex AI Model Registry with the pre-built PyTorch serving container image
3. Step 3 - Create a Vertex AI endpoint and deploy the PyTorch model

This guide will be mainly discussing the details on how to perform Step 1 which is crucial to import the model and deploy it.

- **Note**: even if Vertex AI is able to successfully import the PyTorch model or create the endpoint, this doesn't mean that the model archive was correctly set up or generated. If this is the case the endpoint will commonly throw 500 errors which come from inside the model code or configuration.

Step 1 - Package your PyTorch model into a .mar file with PyTorch Model Archiver

1. Locate the HuggingFace Transformer model you want to deploy to Vertex AI. We won't be downloading the files directly from this source
    a. For example, you can use the XLM-RoBERTa model: https://huggingface.co/xlm-roberta-base

2. To download the model files we will be using the following tool:
    a. **[A]** https://github.com/pytorch/serve/tree/master/examples/Huggingface_Transformers

b. We will follow the guide up until https://github.com/pytorch/serve/tree/master/examples/Huggingface_Transformers#create-model-archive-eager-mode to create the .mar file to be uploaded into GCS and then try to save the model to Vertex AI Model Registry

In my case I deployed the XLM-RoBERTa model. I worked inside a Vertex AI Workbench User managed notebook (JupyterLab). After creating my notebook I downloaded the files from [A]:

- Download_Transformer_models.py
- setup_config.json
- requirements.txt

I added some dependencies I was missing to this requirements.txt file so the file content is as follows:

```
Unset
transformers
optimum
torch-model-archiver
torchserve
```

Then in the notebook cell I ran:

```
Unset
!pip install requirements.txt
```

Now you need to change the values in setup_config.json to the model you would like to download. You can follow this section to understand how to do this. My setup_config.json was set as follows:

```
Unset
{
 "model_name":"xlm-roberta-base",
 "mode":"sequence_classification",
 "do_lower_case":true,
 "num_labels":"2",
 "save_mode":"pretrained",
 "max_length":"150",
 "captum_explanation":true,
```

```
 "embedding_name": "roberta",
 "FasterTransformer":false,
 "BetterTransformer":false,
 "model_parallel":false
}
```

After you have finished making changes you need to run the script. I ran it with the following command in a notebook cell:

```
Unset
! python Download_Transformer_models.py
```

This will automatically download the required files for the model into a directory called "Transformer_model"

After the script finished my "Transformer_model" directory has the following files:

- added_tokens.json
- config.json
- pytorch_model.bin
- sentencepiece.bpe.model
- special_tokens_map.json
- tokenizer_config.json
- tokenizer.json

In case you are missing a file you can always download it from the HuggingFace site. In my case the files are in: https://huggingface.co/xlm-roberta-base

> **! Important Note: Verify the size of the "pytorch_model.bin"** matches with the size of the same file in https://huggingface.co/xlm-roberta-base.
>
> - In my case, as I was using JupyterLab, the file took more than 30 minutes to download because of its size of around 1GB. You can always verify if the size of the file is still increasing (being uploaded) with linux commands. For example by listing the directory's files with:

```
Unset
! ls -la Transformer_model
```

Once all the files have finished downloading we will need to manually download 2 more files. To do so, we first need to create a new folder in the same directory where your "setup_config.json" is located. In my case I'm intending to do "Sequence Classification" I will name this directory "Seq_classification_artifacts"

```
Unset
! mkdir Seq_classification_artifacts
```

From here download the files: "index_to_name.json" and "sample_text_captum_input.txt" and save them into your "Seq_classification_artifacts" folder. More information on this can be found in this section

Finally we will also manually download another necessary file which is the handler "Transformer_handler_generalized.py". You can download this file from here. Save this file in the same directory where your "setup_config.json" is located.

> **Note:** This handler already has the necessary methods to "initialize" your model, "preprocess" your inputs, make an "inference" and finally make any further operations like "postprocess", among others.
>
> **! Important Note:** Pay close attention to the "preprocess" method because it handles your input. By trial and error I found out I was getting 503 errors when the model was already deployed because I was not sending the input as the "preprocess" method required.

Once all necessary files have been completely uploaded we will proceed to package the model into a .mar file using torch-model-archiver:

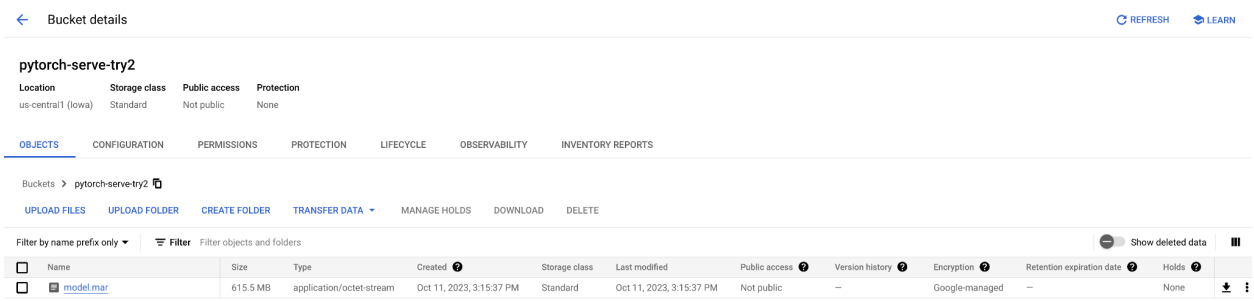
```
Unset
! torch-model-archiver --model-name model --version 1.0 --serialized-file
Transformer_model/pytorch_model.bin --handler
./Transformer_handler_generalized.py --extra-files
"Transformer_model/config.json,./setup_config.json,./Seq_classification_
artifacts/index_to_name.json"
```

> Note: Vertex AI Model Registry will look for a file named "model.mar" therefore the "--model-name" argument is set as "model",

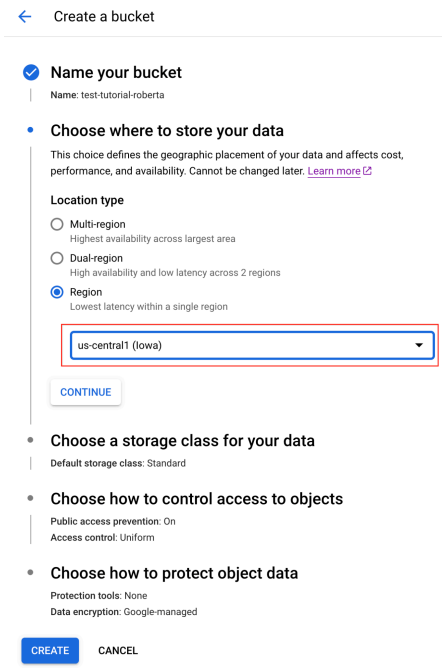
! Important Note: The “model.mar” will take some time to be generated. It will take approximately 5 - 10 minutes. You can always verify if its size is not changing as we have done before with the “ls -la” command

Step 2 - Import the model to Vertex AI Model Registry with the pre-built PyTorch serving container image

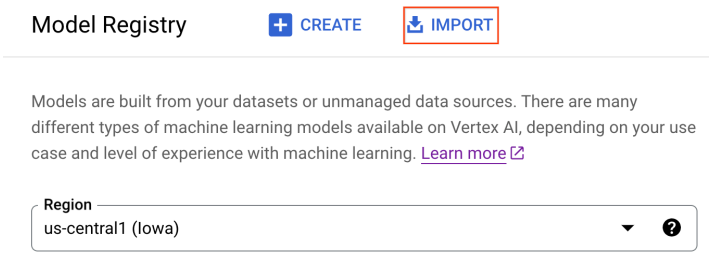
Firstly, download the .mar file. Again, this will take some time to finish due to the size of the file. Then import the .mar file to a Google Cloud Storage bucket. Make sure the file is named as “model.mar”. It also will take some minutes to get completely uploaded.



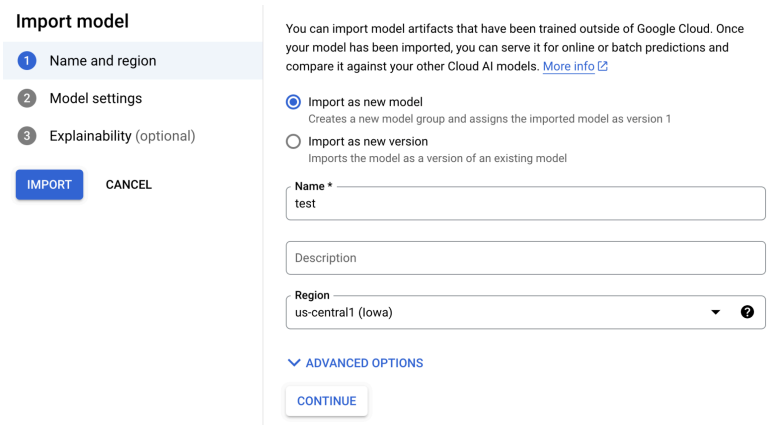
! Important note: Make sure all your resources (Google Cloud Storage bucket, Model Registry and Endpoint) are in the same region. In my case I used “us-central1”



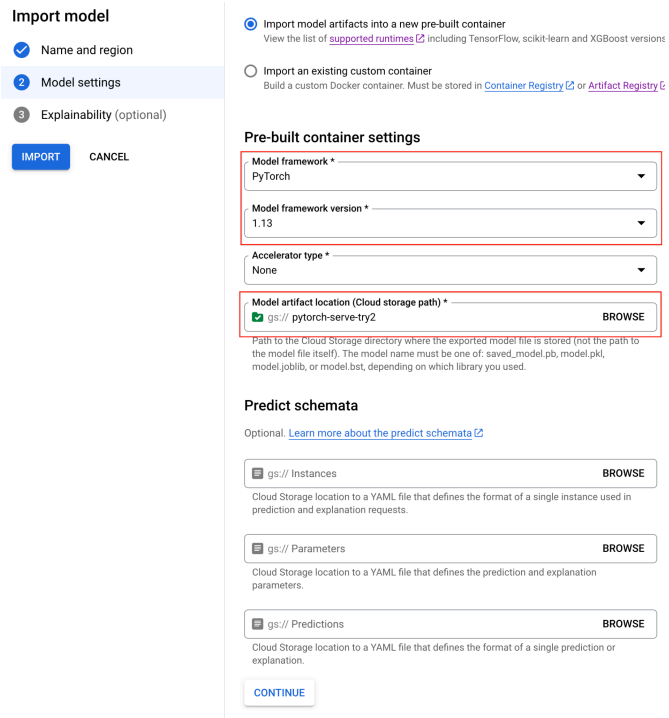
Then, import the model into Vertex AI Model Registry:



Set up the name of the model



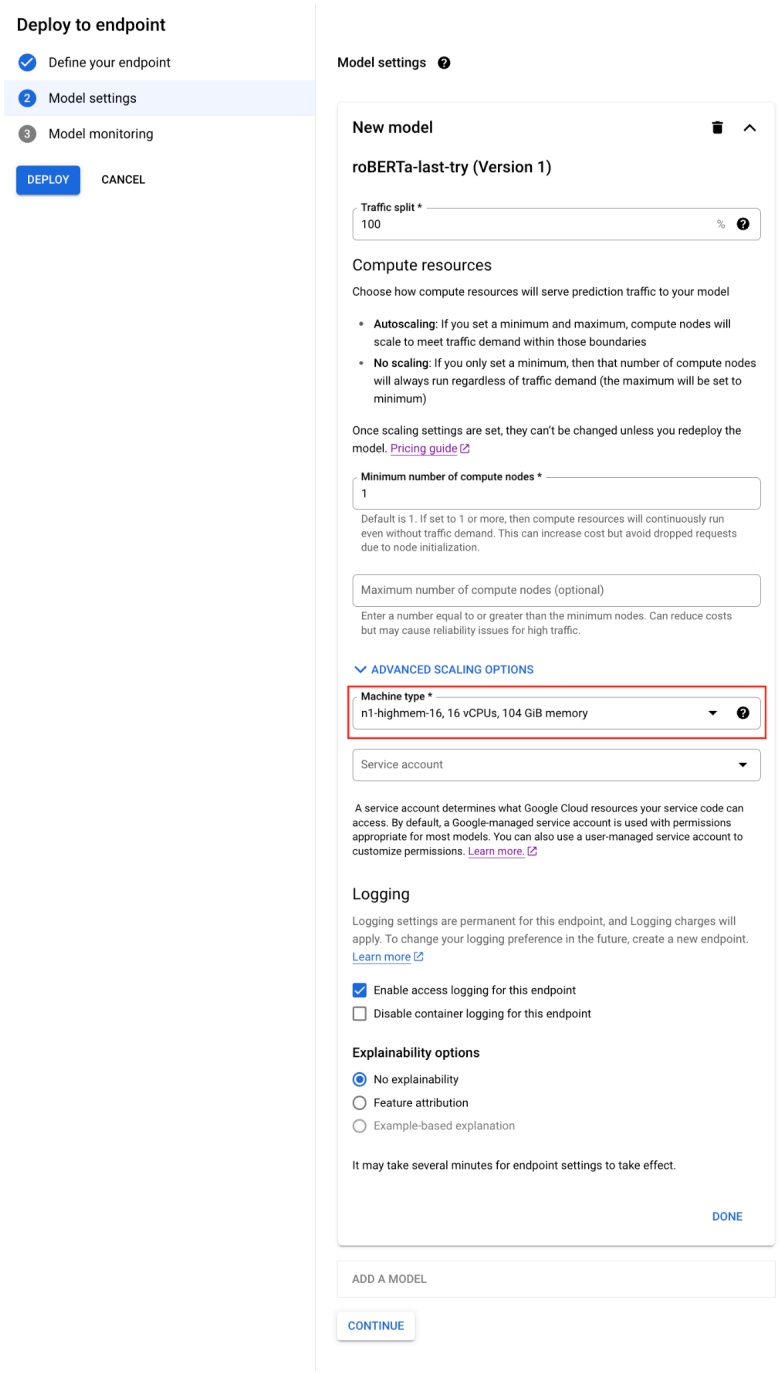
Click on continue, then configure the model settings as follows. Add the name of the bucket where your “model.mar” is located. Then click on import.



It will take 1 or 2 minutes for the model to get imported. Once it is imported we will proceed to deploy to a Vertex AI endpoint

Step 3 - Create a Vertex AI endpoint and deploy the PyTorch model

Click on the model imported in “Model Registry”. Set a name for your endpoint. Click on continue and the only necessary thing to set will be the Machine type. My model required a high memory machine. Then, just click on the deploy button and this step is done. It will take 10-20 minutes to be deployed.



Test your endpoint

You can now test your endpoint using the Vertex AI Python SDK (aiplatform library) as follows:

```python
from google.cloud import aiplatform

aiplatform.init(project=PROJECT_ID, location=REGION)

endpoint = aiplatform.Endpoint(
endpoint_name="projects/PROJECT-NUMBER/locations/REGION/endpoints/ENDPOINT-ID" )
instances = [
  {'data': '{"text":"Bloomberg has decided to publish a new report on the global
economy.", "target":1}'},
  {'data': '{"text":"The cat climbed the wall", "target":1}'}
]
response = endpoint.predict(instances=instances)
print(response.predictions)
```

> **! Important note:** The way in which you send your input depends on how the "preprocess" method in your "Transformer_handler_generalized.py" or the handler you have used processes the request. One certain thing is you will need to still format it as "instances = [{your-data},{your-data}]"

Troubleshooting

If you get any error when testing your endpoint you can see the corresponding logs with the following filter in Cloud Logging:

```
resource.type="aiplatform.googleapis.com/Endpoint"
resource.labels.endpoint_id="ENDPOINT-ID"
```

The error could either be caused by a badly formatted prediction input or some bad configuration or corruption of your model.mar file. **Even if Vertex AI says the model was successfully deployed, this doesn't mean the .mar file is correct.**

Resources:

- [1] https://cloud.google.com/blog/products/ai-machine-learning/prebuilt-containers-with-pytorch-and-vertex-ai

- [2] https://cloud.google.com/blog/topics/developers-practitioners/pytorch-google-cloud-how-deploy-pytorch-models-vertex-ai
- [3] https://github.com/GoogleCloudPlatform/vertex-ai-samples/blob/main/notebooks/official/prediction/pytorch_image_classification_with_prebuilt_serving_containers.ipynb

Other Resources:

- https://www.googlecloudcommunity.com/gc/AI-ML/Error-while-deploying-hugging-pytorch-model-ROBERTA-to-Vertex-AI/m-p/595498
- https://github.com/pytorch/serve/tree/master/examples/Huggingface_Transformers
- https://github.com/GoogleCloudPlatform/vertex-ai-samples/blob/fcee9bf738277bd45b80843120f50a56d8502f18/community-content/pytorch_text_classification_using_vertex_sdk_and_gcloud/pytorch-text-classification-vertex-ai-pipelines.ipynb
- https://github.com/GoogleCloudPlatform/vertex-ai-samples/blob/main/notebooks/official/prediction/pytorch_image_classification_with_prebuilt_serving_containers.ipynb