Report FastAPI

Report day: 21/02/2025

Name: Nguyễn Thanh Hòa

Team: 2 Team leader: Hòa

Team member: Nguyễn Thanh Hòa, Nguyễn Đình Vũ Hải, Nguyễn Quý Toàn

Team member

✅ **Task title 1: Design API to Send Inference Request via API and Call Triton**

▼ 1. Purpose

Mục tiêu là tạo một API sử dụng FastAPI, có thể nhận dữ liệu đầu vào từ người dùng, gửi dữ liệu đó đến Triton Inference Server để thực hiện dự đoán qua giao thức gRPC, và trả lại kết quả cho người dùng. Giao thức gRPC giúp tối ưu hóa tốc độ truyền tải dữ liệu và có thể được sử dụng thay cho HTTP trong các ứng dụng yêu cầu hiệu suất cao.

▼ 2. Action

Công việc được chia thành các bước cụ thể như sau:

Bước 1: Cài đặt FastAPI và Uvicorn

- Cài đặt FastAPI và Uvicorn làm server để chạy FastAPI.

```
pip install fastapi uvicorn
```

Bước 2: Tạo file Python cho ứng dụng FastAPI

- Tạo file `main.py` để định nghĩa ứng dụng FastAPI và thiết lập các route cho API.

Bước 3: Định nghĩa POST Route để nhận dữ liệu

- Định nghĩa route `predict/` để xử lý các yêu cầu POST.
- Route này nhận dữ liệu đầu vào, xử lý và gửi tới Triton Inference Server qua gRPC .

Bước 4: Cài đặt thư viện `grpcio` và `tritonclient.grpc` để gửi yêu cầu gRPC tới Triton

- Cài đặt thư viện `grpcio` và `tritonclient.grpc` để xử lý việc gửi yêu cầu gRPC tới Triton Inference Server.

```
pip install grpcio
```

```
pip install tritonclient
```

Bước 5: Thực hiện gửi yêu cầu tới Triton Inference thông qua gRPC

- Tạo một API để nhận dữ liệu đầu vào từ người dùng, gửi dữ liệu này tới Triton để thực hiện dự đoán qua giao thức gRPC, và trả lại kết quả cho người dùng.
- Sử dụng thư viện `tritonclient.grpc` để gửi yêu cầu gRPC tới Triton.

```
# Địa chỉ Triton Server qua gRPC
TRITON_GRPC_URL = "172.17.0.2:8001"  # Cập nhật URL của Tr
iton Server

# Tạo một route nhận request và gửi tới Triton qua gRPC
@app.post("/predict_grpc/")
def predict_grpc(data: dict):
    """
    Nhận dữ liệu từ người dùng, gửi đến Triton để inference qua
gRPC và trả về kết quả.
    """
    try:
        # Kết nối với Triton Server qua gRPC
        with grpcclient.InferenceServerClient(url=TRITON_GRPC_
URL) as client:
            # Tạo input tensor cho Triton
            input_tensor = InferInput("data_0", [1, 3, 224, 224], "FP
32")
            input_tensor.set_data_from_numpy(np.array(data["dat
a"], dtype=np.float32))

            # Gửi yêu cầu đến Triton
            result = client.infer(model_name="densenet_onnx", inp
uts=[input_tensor])

            # Lấy kết quả từ output của Triton
            output_data = result.as_numpy("output_0").tolist()
            return {"prediction": output_data}
```

```
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Bước 6: Chạy Server

- Chạy ứng dụng FastAPI bằng Uvicorn với tùy chọn `--reload` để tự động tải lại server khi thay đổi mã nguồn.

```
uvicorn main:app --reload
```

Bước 7: Kiểm tra API

- Kiểm tra ứng dụng FastAPI bằng cách gửi yêu cầu gRPC tới server Triton.
- Bạn có thể sử dụng Postman hoặc `curl` để gửi yêu cầu HTTP, nhưng trong trường hợp này, gRPC sẽ được xử lý trực tiếp qua thư viện Python.

Sample request:

```
json
CopyEdit
{
  "inputs": [
    {
      "name": "input_1",
      "shape": [1, 3, 224, 224],
      "datatype": "FP32",
      "data": [0.1, 0.2, 0.3, ...]
    }
  ]
}
```

Kết quả trả về từ API chứa kết quả inference từ Triton.

▼ 3. Result

Ứng dụng FastAPI đã được triển khai thành công với khả năng:

- Nhận dữ liệu qua yêu cầu POST.
- Gửi dữ liệu đến Triton Inference Server để thực hiện dự đoán qua gRPC.
- Trả về kết quả inference cho người dùng dưới dạng JSON.

▼ 4. Upcoming

Bước tiếp theo:

- Cải thiện khả năng xử lý lỗi khi gặp vấn đề với gRPC hoặc dữ liệu không hợp lệ.
- Kiểm tra và tối ưu hóa hiệu suất khi gửi dữ liệu lớn qua gRPC.
- Cấu hình Triton để chắc chắn mô hình hoạt động chính xác.
- Triển khai ứng dụng FastAPI lên môi trường cloud