import tritonclient.http as httpclient

import numpy as np

# הכתובת של השרת Triton שלך

triton\_url = "your-triton-server-url:8000" # שנה את הכתובת

client = httpclient.InferenceServerClient(url=triton\_url)

# יצירת קלט (למשל תמונה בגודל 28x28, מתאימה למודל MNIST)

input\_data = np.random.rand(1, 1, 28, 28).astype(np.float32)

# הגדרת הקלט והפלט

input\_tensor = httpclient.InferInput("INPUT\_\_0", input\_data.shape, "FP32")

input\_tensor.set\_data\_from\_numpy(input\_data)

output\_tensor = httpclient.InferRequestedOutput("OUTPUT\_\_0")

# שליחת הבקשה למודל

response = client.infer(model\_name="mnist", inputs=[input\_tensor], outputs=[output\_tensor])

# הדפסת התוצאה (חיזוי)

print(response.as\_numpy("OUTPUT\_\_0"))

import ipywidgets as widgets

import numpy as np

import tritonclient.http as httpclient

import matplotlib.pyplot as plt

from PIL import Image

from io import BytesIO

# יצירת רכיב העלאת קובץ

upload\_widget = widgets.FileUpload(accept='.png,.jpg,.jpeg', multiple=False)

# פונקציה להציג את התמונה שהועלתה

def show\_uploaded\_image(change):

uploaded\_file = next(iter(upload\_widget.value.values())) # קובץ הועלה

img\_data = uploaded\_file['content'] # התוכן של הקובץ

# המרת התמונה לפורמט שניתן להציג

img = Image.open(BytesIO(img\_data))

plt.imshow(img)

plt.axis('off') # להסתיר את הצירים

plt.show()

upload\_widget.observe(show\_uploaded\_image, names='value')

display(upload\_widget)

# פונקציה לשליחת התמונה לשרת Triton לצורך חיזוי

def predict\_image(image\_data):

# יצירת לקוח לשרת Triton

triton\_url = "your-triton-server-url:8000" # עדכן את הכתובת של השרת שלך

client = httpclient.InferenceServerClient(url=triton\_url)

# המרת התמונה למערך numpy בגודל (1, 1, 28, 28) עבור MNIST

img = Image.open(BytesIO(image\_data))

img = img.convert('L') # המרת התמונה לשחור-לבן

img = img.resize((28, 28)) # שינוי גודל התמונה ל-28x28

img\_data = np.array(img).astype(np.float32)

# קלט ופלט Triton

img\_data = np.expand\_dims(img\_data, axis=0) # הוספת ממד נוסף עבור batch size

img\_data = np.expand\_dims(img\_data, axis=1) # הוספת ממד נוסף עבור ערוץ

input\_tensor = httpclient.InferInput("INPUT\_\_0", img\_data.shape, "FP32")

input\_tensor.set\_data\_from\_numpy(img\_data)

output\_tensor = httpclient.InferRequestedOutput("OUTPUT\_\_0")

# שליחת בקשה לשרת Triton

response = client.infer(model\_name="mnist", inputs=[input\_tensor], outputs=[output\_tensor])

# הצגת התוצאה

result = response.as\_numpy("OUTPUT\_\_0")

predicted\_class = np.argmax(result)

print(f"Predicted class: {predicted\_class}")

return predicted\_class

# פונקציה להריץ אחרי העלאת תמונה

def on\_upload(change):

uploaded\_file = next(iter(upload\_widget.value.values())) # קובץ הועלה

image\_data = uploaded\_file['content']

predicted\_class = predict\_image(image\_data)

print(f"The model predicted the number: {predicted\_class}")

upload\_widget.observe(on\_upload, names='value')

model = torch.load("mnist-model.pt") # טוען את המודל המאומן

model.eval() # מעביר את המודל למצב הערכה (Inference)

example\_input = torch.randn(1, 1, 28, 28) # דוגמה לקלט כדי לעזור לטרייסר להבין את המבנה

traced\_model = torch.jit.trace(model, example\_input) # מעביר את המודל לפורמט TorchScript

traced\_model.save("model\_repository/mnist/1/model.pt") # שומר בפורמט הנכון

### **משתני סביבה רלוונטיים ל-S3 ב-Triton**

1. **AWS\_ENDPOINT\_URL** – כתובת ה-**endpoint** של ה-S3 הארגוני.
2. **AWS\_ACCESS\_KEY\_ID** – מפתח הגישה שלך ל-S3.
3. **AWS\_SECRET\_ACCESS\_KEY** – המפתח הסודי של הגישה ל-S3.
4. **AWS\_DEFAULT\_REGION** – האזור (region) שבו נמצא ה-S3 שלך (אם רלוונטי).
5. **TRITON\_MODEL\_REPOSITORY** – הנתיב של מאגר המודלים, בפורמט S3.