بسمه‌تعالی

راهنمای تمرین چهارم

- شرح فایل راهنما

اولین گام در استقرار مدل‌ها با استفاده از تریتون، ساخت یک repository است که مدل‌هایی را که ارائه می‌شوند و طرح configuration را در خود جای می‌دهد. برای این تمرین ما از یک مدل EAST برای Text Detection و یک مدل text recognition استفاده خواهیم کرد.

- مدل اول : **Text Detection**

مدل EAST را با دستور زیر دانلود و از حالت zip خارج کنید.

```
wget https://www.dropbox.com/s/r2ingd0l3zt8hxs/frozen_east_text_detection.tar.gz
tar -xvf frozen_east_text_detection.tar.gz
```

تبدیل مدل به فرمت ONNX

```
pip install -U tf2onnx
python -m tf2onnx.convert --input frozen_east_text_detection.pb --inputs "input_images:0" --outputs "feature_fusion/Conv_7/Sigmoid:0","feature_fusion/concat_3:0" --output detection.onnx
```

- مدل دوم : **Text Recognition**

با استفاده از دستور زیر وزن های مدل Text Recognition را دانلود کنید.

```
wget https://www.dropbox.com/sh/j3xmli4di1zuv3s/AABzCC1KGbIRe2wRwa3diWKwa/None-ResNet-None-CTC.pth
```

بسمه‌تعالی

راهنمای تمرین چهارم

با اجرای قطعه کد زیر مدل ها را به فرمت onnx. تبدیل کنید

```
import torch
from utils.model import STRModel
# Create PyTorch Model Object
model = STRModel(input_channels=1, output_channels=512, num_classes=37)
# Load model weights from external file
state = torch.load("None-ResNet-None-CTC.pth")
state = {key.replace("module.", ""): value for key, value in state.items()}
model.load_state_dict(state)
# Create ONNX file by tracing model
trace_input = torch.randn(1, 1, 32, 100)
torch.onnx.export(model, trace_input, "str.onnx", verbose=True)
```

بسمه‌تعالی

راهنمای تمرین چهارم

- آماده سازی **repository** مدل

ساختار repository مدل باید به صورت زیر باشد که در فایل ارسالی ساختار مورد نظر رعایت شده است.

```
# Example repository structure
<model-repository>/
  <model-name>/
    [config.pbtxt]
    [<output-labels-file> ...]
    <version>/
      <model-definition-file>
    <version>/
      <model-definition-file>
    ...
  <model-name>/
    [config.pbtxt]
    [<output-labels-file> ...]
    <version>/
      <model-definition-file>
    <version>/
      <model-definition-file>
    ...
```

بسمه‌تعالی

راهنمای تمرین چهارم

...

- **آماده سازی فایل های model.onnx**

هر دو فایل onnx. مربوط به دو مدل فوق باید با نام model.onnx در فولدرهای مربوطه در فولدر model_repository و سپس فولدر ۱ که شماره ورژن مدل است قرار بگیرند.(در فایل ارسالی مدل ها در فولدر مربوطه قرار گرفته اند.)

- **آماده سازی فایل های model configuration**

برای هر مدل باید یک فایل config.pbtxt ساخته شده و در فولدر مربوطه قرار بگیرد.(در فایل ارسالی فایل های config.pbtxt در فولدر مربوطه قرار گرفته اند.)

- **Launching the server**

قبل از ادامه لازم است موارد زیر را روی سیستم خود نصب داشته باشید.

- docker
- nvidia-container-toolkit
- nvidia-ctk runtime
- nvidia-docker2

برای نصب nvidia-container-toolkit و nvidia-ctk runtime می توانید سایت زیر استفاده کنید

https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html

بسمه‌تعالی

راهنمای تمرین چهارم

با ایجاد repository و configuration مدل، اکنون آماده راه اندازی سرور هستیم. در حالی که Triton Inference Server را می توان از source ساخت ولی، به دلیل تحریم ها و فیلتر ها این کار تنها با استفاده از فیلترشکن امکان پذیر است. که برای ساخت آن کافی است دستور زیر را اجرا کنید

```
docker run --gpus=all -it --shm-size=256m --rm -p8000:8000 -p8001:8001 -p8002:8002 -v $(pwd)/model_repository:/models nvcr.io/nvidia/tritonserver:<23.01>-py3
```

در صورت نداشتن فیلتر شکن می توانید از کانتینر Docker از پیش ساخته شده برای این تمرین استفاده کنید (ممکن است اجرای این دستور بیشتر از ۱۰ ساعت زمان ببرد)

```
docker run --gpus=all -it --shm-size=256m --rm -p8000:8000 -p8001:8001 -p8002:8002 -v $(pwd)/model_repository:/models ramintoosi/avir.co:tritonserver-23.01-py3
```

سپس دستور زیر را اجرا کنید.

```
tritonserver --model-repository=/models
```

با اجرای دستور فوق باید خروجی زیر را ببینید

```
I0712 16:37:18.246487 128 server.cc:626]
+------------------+---------+--------+
| Model            | Version | Status |
+------------------+---------+--------+
| text_detection   | 1       | READY  |
| text_recognition | 1       | READY  |
+------------------+---------+--------+

I0712 16:37:18.267625 128 metrics.cc:650] Collecting metrics for GPU 0: NVIDIA GeForce RTX 3090
I0712 16:37:18.268041 128 tritonserver.cc:2159]
+----------------------------------+------------------------------------------------------------------------------------------
| Option                           | Value
+----------------------------------+------------------------------------------------------------------------------------------
| server_id                        | triton
| server_version                   | 2.23.0
| server_extensions                | classification sequence model_repository model_repository(unload_dependents) sch
| model_repository_path[0]         | /models
| model_control_mode               | MODE_NONE
| strict_model_config              | 1
| rate_limit                       | OFF
| pinned_memory_pool_byte_size     | 268435456
| cuda_memory_pool_byte_size{0}    | 67108864
| response_cache_byte_size         | 0
| min_supported_compute_capability | 6.0
| strict_readiness                 | 1
| exit_timeout                     | 30
+----------------------------------+------------------------------------------------------------------------------------------

I0712 16:37:18.269464 128 grpc_server.cc:4587] Started GRPCInferenceService at 0.0.0.0:8001
I0712 16:37:18.269956 128 http_server.cc:3303] Started HTTPService at 0.0.0.0:8000
I0712 16:37:18.311686 128 http_server.cc:178] Started Metrics Service at 0.0.0.0:8002
```

بسمه‌تعالی

راهنمای تمرین چهارم

نکته : در صورت مشاهده ارور ۴۰۳ می توانید از دستورات زیر استفاده کنید :

محتوای فایل daemon.json را با دستورات زیر تغییر دهید

```
sudo nano /etc/docker/daemon.json
```

محتوای زیر زا در فایل daemon.json کپی و ذخیره کیند

```
{ "registry-mirrors": ["https://docker.iranrepo.ir", "https://docker.arvancloud.ir", "https://docker.iranserver.com"], "runtimes": { "nvidia": { "args": [], "path": "nvidia-container-runtime" } } }
```

سپس دستورات زیر اجرا کنید تا docker را مجددا راه اندازی کنید

```
systemctl daemon-reload
```

```
systemctl restart docker
```

- **Building a client application**

اکنون که سرور Triton ما راه اندازی شده است، می توانیم با اجرای کد client.py

که در فایل های ارسالی قرار دارد شروع به ارسال پیام به آن کنیم.

```
pip install tritonclient[http] opencv-python-headless
```

```
python client.py
```