

Triton POC Testing

```
# Using 23.02 (date:03-06-23)
docker run --gpus all --rm -p 8000:8000 -p 8001:8001 -p 8002:8002 -v ${PWD}/model_repository:/models nvcr.io/nvidia/tritonserver:23.02-py3 tritonserver --model-repository=/models

# Running the client server
docker run -it --rm --net=host -v ${PWD}:/workspace/ nvcr.io/nvidia/tritonserver:23.02-py3 -sdk bash
```

Triton Client Code

```
import tritonclient.http as httpclient

# Read in the image with pytorch to get the shape
https://pytorch.org/vision/main/generated/torchvision.io.read\_image.html

# Check if the server is alive
client.is_server_live()

transformed_img_shape = list(transformed_img.shape())

# Prep the data shape for inference
inputs = httpclient.InferInput("data_0", transformed_img_shape, datatype="FP32")

# Convert the data to numpy to send to inference server
import numpy as np
>>> inputs.set_data_from_numpy(np.float32(transformed_img.numpy()), binary_data=True)

inputs.set_data_from_numpy(transformed_img, binary_data=True)

outputs = httpclient.InferRequestedOutput("fc6_1", binary_data=True, class_count=1000)typ
```

```
# Use this to get the models on the server
client.get_model_repository_index()
[{'name': 'densenet_onnx', 'version': '1', 'state': 'READY'}]
```

```
>>> client.get_model_repository_index()
[{'name': 'densenet_onnx', 'version': '1', 'state': 'READY'}]
>>> client.get_model_config('densenet_onnx')
{'name': 'densenet_onnx', 'platform': 'onnxruntime_onnx', 'backend': 'onnxruntime', 'version_policy': {'latest': {'num_versions': 1}}, 'max_batch_size': 0, 'input': [{'name': 'data_0', 'data_type': 'TYPE_FP32', 'format': 'FORMAT_NONE', 'dims': [1, 3, 224, 224], 'is_shape_tensor': False, 'allow_ragged_batch': False, 'optional': False}], 'output': [{'name': 'fc6_1', 'data_type': 'TYPE_FP32', 'dims': [1, 1000, 1, 1], 'label_filename': '', 'is_shape_tensor': False}], 'batch_input': [], 'batch_output': [], 'optimization': {'priority': 'PRIORITY_DEFAULT', 'input_pinned_memory': {'enable': True}, 'output_pinned_memory': {'enable': True}, 'gather_kernel_buffer_threshold': 0, 'eager_batching': False}, 'instance_group': [{'name': 'densenet_onnx', 'kind': 'KIND_GPU', 'count': 1, 'gpus': [0], 'secondary_devices': [], 'profile': [], 'passive': False, 'host_policy': ''}], 'default_model_filename': 'model.onnx', 'cc_model_filenames': {}, 'metric_tags': {}, 'parameters': {}, 'model_warmup': []}
>>>
```

```
perf_analyzer -m densenet_onnx -b 1 --shape input.1:3,224,224 --concurrency-range 2:8:2 --percentile=95
```