GPU Functional/Load Tests - Sample PyTorch script

Sample PyTorch script, executed from a Jupyter Notebook in OpenShift AI. Script should return some basic information about the GPUs and compares CPU vs. GPU times.

```
!nvidia-smi
```

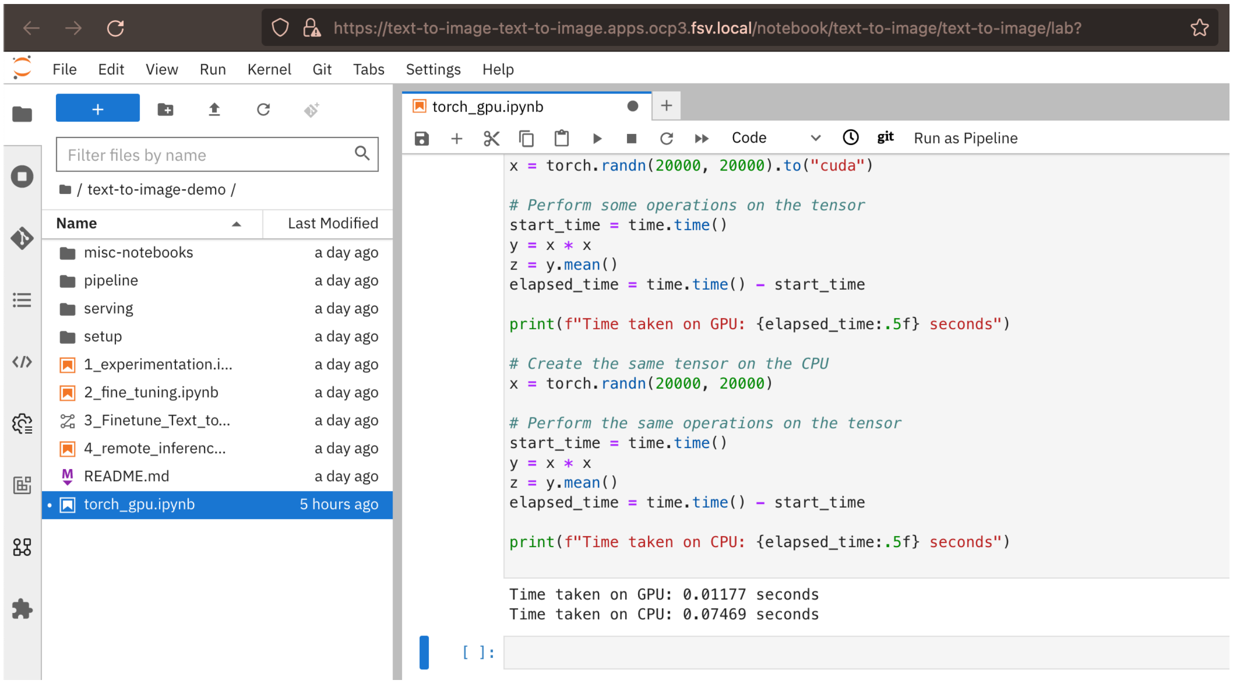
```
import torch

# Check how many GPUs are available
num_of_gpus = torch.cuda.device_count()
print(f"Number of available GPUs: {num_of_gpus}")

# Get a list of all the currently available GPUs
available_gpus = [torch.cuda.device(i) for i in range(num_of_gpus)]
print(f"List of available GPUs: {available_gpus}")
```

```
import torch

# Get a list of all the currently available GPUs
gpu_names = [torch.cuda.get_device_properties(i).name for i in
range(torch.cuda.device_count())]
print(f"List of available GPUs: {gpu_names}")
```

```
import torch

# Check if CUDA is available
if torch.cuda.is_available():
# Set the device to the first available GPU
device = torch.device("cuda:0")
print(f"Using {torch.cuda.get_device_name(device)}")
```

```
# Create a tensor on the GPU
x = torch.randn(3, 3).to(device)

# Perform some operations on the tensor
y = x + x

# Move the tensor back to the CPU
z = y.to("cpu")

# Print the result
print(z)
else:
print("CUDA is not available")
```

```
import torch
import time


if torch.cuda.is_available():
# Create a large tensor on the GPU
x = torch.randn(20000, 20000).to("cuda")

# Perform some operations on the tensor
start_time = time.time()
y = x * x
z = y.mean()
elapsed_time = time.time() - start_time

print(f"Time taken on GPU: {elapsed_time:.5f} seconds")
else:
print("CUDA is not available")
# Create the same tensor on the CPU
x = torch.randn(20000, 20000)

# Perform the same operations on the tensor
start_time = time.time()
y = x * x
z = y.mean()
elapsed_time = time.time() - start_time

print(f"Time taken on CPU: {elapsed_time:.5f} seconds")
```



Name	Last Modified
misc-notebooks	a day ago
pipeline	a day ago
serving	a day ago
setup	a day ago
1_experimentation.i...	a day ago
2_fine_tuning.ipynb	a day ago
3_Finetune_Text_to...	a day ago
4_remote_inferenc...	a day ago
README.md	a day ago
torch_gpu.ipynb	5 hours ago

```
x = torch.randn(20000, 20000).to("cuda")

# Perform some operations on the tensor
start_time = time.time()
y = x * x
z = y.mean()
elapsed_time = time.time() - start_time

print(f"Time taken on GPU: {elapsed_time:.5f} seconds")

# Create the same tensor on the CPU
x = torch.randn(20000, 20000)

# Perform the same operations on the tensor
start_time = time.time()
y = x * x
z = y.mean()
elapsed_time = time.time() - start_time

print(f"Time taken on CPU: {elapsed_time:.5f} seconds")
```

```
Time taken on GPU: 0.01177 seconds
Time taken on CPU: 0.07469 seconds
```

```
[ ]:
```