Contents

1 CUDA Debugging Scripts

Two standalone scripts to help diagnose CUDA initialization issues.

1.1 Scripts

1.1.1 1. test_cuda_init.py - Direct CUDA Driver API Test

Tests CUDA initialization using the raw CUDA driver API (via ctypes), bypassing TensorFlow entirely.

Run:

```
python3 test_cuda_init.py
```

What it does: - Loads `libcuda.so.1` directly - Calls `cuInit(0)` to initialize CUDA - Reports success/failure with error codes - If successful, enumerates GPU devices and compute capabilities - If failed, shows nvidia-smi output and device permissions

Exit codes: - `0` - cuInit() succeeded - `1` - cuInit() failed or library not found

This tests: Whether CUDA driver itself can initialize, independent of TensorFlow.

1.1.2 2. test_tf_cuda_init.py - TensorFlow CUDA Test (Verbose)

Tests CUDA initialization through TensorFlow with maximum logging enabled.

Run:

```
python3 test_tf_cuda_init.py
```

What it does: - Sets maximum TensorFlow verbosity (`TF_CPP_MIN_LOG_LEVEL=0`) - Enables CUDA-specific module logging (`TF_CPP_VMODULE=cuda_diagnostics=10,...`) - Imports TensorFlow and lists GPU devices - Attempts a simple GPU matrix multiplication - Provides diagnosis for common errors

Exit codes: - `0` - GPU operation succeeded - 1 - GPU initialization or operation failed

This tests: Whether TensorFlow can initialize and use CUDA/GPU.

1.2 Usage

1.2.1 First Test: Raw CUDA

Start with the direct CUDA test to see if the driver works at all:

```
cd /data/piggy/src/github.com/piggyatbaqaqi/skol
python3 test_cuda_init.py
```

Expected for RTX 5090: - ✓ Should succeed (cuInit works, driver is fine) - Reports compute capability 12.0

1.2.2 Second Test: TensorFlow

Then test TensorFlow specifically:

```
python3 test_tf_cuda_init.py 2>&1 | tee tf_cuda_test.log
```

The `2>&1 | tee` captures all output (including stderr) to a file for review.

Expected for RTX 5090: - ✗ Will likely fail with `CUDA_ERROR_INVALID_PTX` - Logs will show JIT compilation failure - Diagnosis will recommend CPU-only mode

1.3 Interpreting Results

1.3.1 Case 1: Both scripts succeed

✓ CUDA driver works ✓ TensorFlow can use GPU → **No issues**, GPU training will work

1.3.2 Case 2: test_cuda_init.py succeeds, test_tf_cuda_init.py fails

✓ CUDA driver works ✗ TensorFlow can't use GPU → **Compute capability issue** (like RTX 5090 with TF 2.21) → Use CPU-only mode or wait for TensorFlow update

1.3.3 Case 3: Both scripts fail

✗ CUDA driver doesn't work ✗ TensorFlow can't use GPU → **Driver/system issue** → Check: - `nvidia-smi` works? - `/dev/nvidia*` permissions? - Persistence mode: `sudo nvidia-smi -pm 1` - Need reboot?

1.4 Verbose Logging Explained

The `test_tf_cuda_init.py` script sets these environment variables:

- `TF_CPP_MIN_LOG_LEVEL=0` - Show all log levels (INFO, WARNING, ERROR)
- `TF_CPP_VMODULE=cuda_diagnostics=10,...` - Set module-specific verbosity to 10
- `CUDA_LAUNCH_BLOCKING=1` - Make CUDA launches synchronous for clearer errors

This produces extensive output showing: - CUDA initialization steps - Driver/kernel version checks - Device enumeration - PTX/kernel compilation attempts - Exact point of failure

1.5 Saving Output

To save full verbose output for analysis:

```
# TensorFlow test with full logs
python3 test_tf_cuda_init.py 2>&1 | tee tf_verbose.log

# Direct CUDA test
python3 test_cuda_init.py 2>&1 | tee cuda_direct.log
```

Review the logs to find the exact error message and stack trace.

1.6 What to Look For

In `test_tf_cuda_init.py` output, search for:

Success indicators: - `"Created device /job:localhost/replica:0/task:0/device:GPU:0"` - `"✓✓✓ GPU OPERATION SUCCESSFUL ✓✓✓"`

Failure indicators: - "`CUDA_ERROR_INVALID_PTX`" → Compute capability too new - "`CUDA_ERROR_INVALID_HANDLE`" → Usually follows INVALID_PTX - "`CUDA_ERROR_UNKNOWN`" → Generic init failure - "`failed call to cuInit`" → Driver initialization failed - "`JIT compiled from PTX`" followed by errors → Compilation failed

1.7 Next Steps

After running both scripts:

1. **If test_cuda_init.py succeeds but test_tf_cuda_init.py fails:**
    - Your hardware is fine
    - Use CPU-only mode: `os.environ['CUDA_VISIBLE_DEVICES'] = ''`
    - The auto-detection in `rnn_model.py` should handle this
2. **If both fail:**
    - Check NVIDIA driver installation
    - Try: `sudo nvidia-smi -pm 1`
    - Reboot and test again
    - Check kernel module: `lsmod | grep nvidia`
3. **If both succeed:**
    - Something else is wrong with your training setup
    - The RTX 5090 should work (surprisingly!)
    - Check TensorFlow version and build info