ROLV Validation Test

Official Hardware Verification & Benchmarking Kit

1. Purpose

This test establishes a performance and energy baseline on your specific hardware. ROLV uses the data generated here to create a "Us vs. Them" report, showing exactly how much faster and more efficient your workload becomes when processed via ROLV Sparse primitives.

2. The Role of SHA-256 Hashes (Mathematical Legitimacy)

To prove our results are legitimate, we use SHA-256 hashing:

- **The Baseline:** Your hardware generates a unique "fingerprint" (hash) of the calculation result.
- **The Match:** When ROLV processes the same data, we must produce the **exact same hash**.
- **The Proof:** This guarantees that ROLV is not "skipping" math or reducing precision. We deliver the exact same high-fidelity results as your current vendor, just significantly faster and with less power.

---

3. Python Verification Script

Save the code below as rolv-verifier.py or paste it into a Jupyter Notebook cell.

Python

```
#!/usr/bin/env python3

import torch

import numpy as np

import time

import hashlib

import json

import argparse

import os

import sys

import platform

import psutil

import re
```

```
# ================================================================================
# [ CONFIGURATION BLOCK ] - YOU CAN CHANGE ANY OF THESE PARAMETERS
# ================================================================================
# ROLV will use these exact dimensions to duplicate your test.
# --------------------------------------------------------------------------------
TEST_ROWS      = 20000    # Matrix Rows (M)
TEST_COLS      = 20000    # Matrix Columns (K)
TEST_BATCH     = 5000     # Batch Size / Tokens (N)
TEST_SPARSITY  = 0.70     # Percentage of zeros (0.0 to 1.0)
TEST_ITERS     = 1000     # Number of benchmark iterations

# MODE: What are you currently testing?
# Options: "baseline_dense", "vendor_dense", "vendor_sparse", "vendor_mkl", "vendor_cublas"
TEST_MODE      = "baseline_dense"
# ================================================================================


def get_full_env():
    """Captures every variable of the hardware and software environment."""
    env = {
        "processor_model": platform.processor(),
        "cpu_architecture": platform.machine(),
        "cpu_cores_physical": psutil.cpu_count(logical=False),
        "cpu_cores_logical": psutil.cpu_count(logical=True),
        "total_system_ram_gb": round(psutil.virtual_memory().total / (1024**3), 2),
        "os_version": f"{platform.system()} {platform.release()}",
        "python_version": sys.version.split()[0],
        "torch_version": torch.__version__,
        "compute_backend": torch.__config__.show(),
    }
```

```
    if torch.cuda.is_available():
        env["accelerator"] = torch.cuda.get_device_name(0)
        env["vram_gb"] = round(torch.cuda.get_device_properties(0).total_memory / (1024**3), 2)
    elif hasattr(torch.backends, 'mps') and torch.backends.mps.is_available():
        env["accelerator"] = "Apple Silicon Integrated GPU"
    return env

def run_rolv_test(user_email, rows, cols, batch, zeros, iters):
    device = torch.device("cuda" if torch.cuda.is_available() else ("mps" if
torch.backends.mps.is_available() else "cpu"))
    env_specs = get_full_env()

    print(f"\n[*] Initializing Data on {device}...")
    torch.manual_seed(12345)
    density = 1.0 - zeros
    A = (torch.rand(rows, cols, device=device) * (torch.rand(rows, cols, device=device) < density))
    V = torch.rand(cols, batch, device=device)

    # Warmup
    for _ in range(10): torch.matmul(A, V)
    if device.type == 'cuda': torch.cuda.synchronize()

    # Benchmark execution
    print(f"[*] Benchmarking {iters} iterations...")
    start_time = time.perf_counter()
    for _ in range(iters):
        Y = torch.matmul(A, V)
    if device.type == 'cuda': torch.cuda.synchronize()
    end_time = time.perf_counter()
```

```
# Metrics
duration = end_time - start_time
avg_sec = duration / iters
tflops = ((2 * rows * cols * batch) / avg_sec) / 1e12
tokens_per_sec = (batch * iters) / duration

# Energy Metrics (Joules and Watts)
wattage_estimate = 150.0
total_joules = wattage_estimate * duration

# Create the Full Data Package
return {
   "user_info": {"email": user_email},
   "metadata": {"mode": TEST_MODE, "timestamp": time.strftime("%Y-%m-%d %H:%M:%S")},
   "environment": env_specs,
   "parameters": {
      "rows": rows, "cols": cols, "batch": batch,
      "sparsity": zeros, "iterations": iters
   },
   "metrics": {
      "latency_ms_per_iter": avg_sec * 1000,
      "throughput_tokens_sec": tokens_per_sec,
      "compute_tflops": tflops,
      "total_joules_consumed": total_joules,
      "recorded_watts": wattage_estimate,
      "sha256_result_hash": hashlib.sha256(Y.cpu().numpy().tobytes()[:4000000]).hexdigest()
   }
}
```

```
if __name__ == "__main__":
    print("\n" + "="*60)
    print(" ROLV VALIDATION: HARDWARE IDENTIFICATION")
    print("="*60)

    # Capture user email for file naming and identification
    user_email = input("Please enter your work email: ").strip()
    if not user_email: user_email = "anonymous_user"

    # Clean email for filename
    clean_email = re.sub(r'[^a-zA-Z0-9]', '_', user_email)
    filename = f"rolv_results_{clean_email}_{time.strftime('%Y%m%d_%H%M')}.json"

    full_results = run_rolv_test(user_email, TEST_ROWS, TEST_COLS, TEST_BATCH, TEST_SPARSITY,
TEST_ITERS)

    # Print the full JSON output to terminal
    print("\n[ GENERATED DATA PACKAGE ]")
    print(json.dumps(full_results, indent=4))

    with open(filename, "w") as f:
        json.dump(full_results, f, indent=4)

    print("\n" + "="*60)
    print(f" SUCCESS: Results saved to {filename}")
    print(f" Performance: {full_results['metrics']['throughput_tokens_sec']:.2f} tokens/s")
    print(f" Energy: {full_results['metrics']['total_joules_consumed']:.2f} Joules")
    print("="*60)
```

```
print(f"ACTION: Email '{filename}' (or copy/paste the JSON above) to rolv@rolv.ai.")
```

---

4. How to Run & Submit

For Advanced Users:

1. Save the code as rolv-verifier.py.
2. Run via terminal: python rolv-verifier.py.
3. Email the generated .json file to **rolv@rolv.ai**.

For Novice Users (Recommended):

1. Download **Anaconda** (which includes Jupyter Notebook) from anaconda.com/download.
2. Open a new **Jupyter Notebook**.
3. Paste the code above into a cell and press **Shift + Enter**.
4. Enter your email when prompted.
5. **Submission:** You can either email the .json file created in your folder **OR** simply copy the text between [GENERATED DATA PACKAGE] and the final } and paste it into an email to **rolv@rolv.ai**.