MPS NaN Error Fix - Documentation

Overview

This document describes the MPS (Apple Silicon GPU) NaN error issue and the automatic fallback solution implemented in Transcribe RO.

The Problem

What is the MPS NaN Error?

When using Whisper on Apple Silicon GPUs (M1, M2, M3, etc.) via the MPS (Metal Performance Shaders) backend, users may encounter numerical instability errors that produce NaN (Not a Number) values. The error typically looks like:

```
Error during transcription: Expected parameter logits (Tensor of shape (1, 51865))
of distribution Categorical(logits: torch.Size([1, 51865])) to satisfy the
constraint IndependentConstraint(Real(), 1), but found invalid values:
tensor([[nan, nan, nan, ..., nan, nan, nan]], device='mps:0')
```

Why Does This Happen?

This is a known issue with:
- **Whisper model on MPS**: The Whisper model uses certain operations that can cause numerical instability on MPS
- **FP16/FP32 precision**: While MPS works better with FP32, some internal operations may still produce NaN values
- **Hardware-specific**: More common on certain chip variants (e.g., M3 MAX)
- **Non-deterministic**: May work sometimes and fail other times with the same audio

The Solution

Automatic CPU Fallback

Transcribe RO now implements a **transparent automatic fallback** mechanism:

1. **Detection**: When transcription fails on MPS, the error message is analyzed to detect NaN-related errors
2. **Notification**: User is informed that a known MPS issue was detected
3. **Automatic Retry**: Model is automatically reloaded on CPU
4. **Seamless Continuation**: Transcription continues on CPU without user intervention
5. **Session Persistence**: Subsequent transcriptions in the same session will use CPU

Additional Safety Measures

1. **Environment Variables**: MPS-specific environment variables are set at startup:
   - `PYTORCH_ENABLE_MPS_FALLBACK=1`: Enables PyTorch's built-in MPS fallback
   - `PYTORCH_MPS_HIGH_WATERMARK_RATIO=0.0`: Helps prevent memory-related issues

2. **User Warnings**: When MPS is detected, users are warned about potential issues

3. **Manual Override**: New `--force-cpu` flag allows users to skip GPU entirely

Usage

CLI (Command Line Interface)

Automatic Fallback (Default)

Simply use the tool normally. If MPS fails, it will automatically retry on CPU:

```
python transcribe_ro.py audio.mp3
```

If MPS encounters a NaN error, you'll see:

```
================================================================================
⚠️  MPS NaN ERROR DETECTED
================================================================================
This is a known issue with Whisper on Apple Silicon GPUs.
The model encountered numerical instability (NaN values).
Automatically falling back to CPU for stable transcription...
================================================================================
Loading model on CPU device...
✓ Model successfully reloaded on CPU!
Retrying transcription on CPU...
================================================================================
✓ CPU FALLBACK SUCCESSFUL!
================================================================================
Transcription completed using CPU after MPS encountered errors.
Note: Future transcriptions in this session will use CPU.
================================================================================
```

Force CPU Mode

To avoid GPU entirely and run on CPU from the start:

```
python transcribe_ro.py audio.mp3 --force-cpu
```

This is useful if:
- You know your audio consistently triggers MPS errors
- You want to avoid the initial GPU attempt
- You're troubleshooting other issues

Specify Device

You can still manually specify the device:

```
# Try MPS (will auto-fallback if issues occur)
python transcribe_ro.py audio.mp3 --device mps

# Force CPU
python transcribe_ro.py audio.mp3 --device cpu

# Auto-detect (default)
python transcribe_ro.py audio.mp3 --device auto
```

GUI (Graphical Interface)

Automatic Fallback (Default)

The GUI automatically benefits from the fallback mechanism. Just use it normally:

1. Launch the GUI: `./run_gui.sh` (or `run_gui.bat` on Windows)
2. Select your audio file
3. Click “Start Transcription”

If MPS fails, the transcription will automatically retry on CPU.

Force CPU Option

A new checkbox is available in the Settings section:

```
☐ 🔧 Force CPU (bypass GPU issues)
⚠️ Check this if you experience MPS/GPU NaN errors.
   Automatic fallback is enabled by default.
```

Check this box to run on CPU from the start.

Performance Impact

MPS vs CPU Performance

- **MPS (when working)**: 3-5x faster than CPU
- **CPU**: Slower but 100% reliable
- **Automatic fallback**: Initial MPS attempt + model reload adds ~30-60 seconds overhead, then continues on CPU

Recommendations

1. **First-time users**: Try default settings (auto-detect with fallback)
2. **Known problematic audio**: Use `--force-cpu` or check the Force CPU box
3. **Batch processing**: If first file fails on MPS, use `--force-cpu` for remaining files

Technical Details

NaN Detection Logic

The error detection looks for:

- Explicit “nan” or “NaN” mentions (as whole words)
- “invalid values” in context of tensors
- Constraint-related errors with “found invalid”

This ensures we catch the specific MPS NaN errors without false positives.

Code Changes

CLI (`transcribe_ro.py`):
- Environment variables set at module import
- `_detect_nan_error()` method for error analysis
- Modified `transcribe_audio()` with retry logic

- New `--force-cpu` CLI flag
- Warning messages in device detection

GUI (`transcribe_ro_gui.py`):
- Environment variables set at module import
- New force_cpu checkbox option
- Device selection respects force_cpu setting
- Inherits automatic fallback from CLI's `AudioTranscriber` class

Environment Variables

```
os.environ['PYTORCH_ENABLE_MPS_FALLBACK'] = '1'
os.environ['PYTORCH_MPS_HIGH_WATERMARK_RATIO'] = '0.0'
```

These are set before PyTorch is loaded and help prevent some MPS issues at the PyTorch level.

Troubleshooting

Issue: MPS still fails even with environment variables

Solution: Use `--force-cpu` flag or check Force CPU in GUI

Issue: Want to see detailed fallback process

Solution: Use `--debug` flag:

```
python transcribe_ro.py audio.mp3 --debug
```

This shows:
- Device detection process
- Error analysis
- Model reloading steps
- Detailed timing information

Issue: CPU is too slow

Options:
1. Use a smaller model: `--model tiny` or `--model small`
2. Process shorter audio segments
3. Consider upgrading to a system with CUDA support (NVIDIA GPU)

Issue: Automatic fallback doesn't work

Check:
1. Make sure you're using the latest version
2. Run with `--debug` flag to see error details
3. Check if error is actually a NaN error (will be explicitly stated)

Known Limitations

1. **First attempt overhead**: If MPS fails, you pay the cost of:
   - Initial model load on MPS
   - Initial transcription attempt

   - Model reload on CPU
   - This adds ~30-60 seconds

2. **Session persistence**: Once fallback occurs, entire session uses CPU
3. **No automatic retry to MPS**: If MPS fails once, we don't attempt MPS again in the same session

Future Improvements

Potential enhancements being considered:

1. **Pre-detection**: Try to detect problematic audio before starting transcription
2. **Per-file retry**: Allow MPS retry on subsequent files
3. **Better MPS support**: Work with PyTorch team on upstream fixes
4. **Alternative backends**: Explore Core ML or other Apple-native options

Testing

A test suite is available to verify the implementation:

```
python test_mps_fallback.py
```

This tests:
- Environment variables
- NaN detection logic
- CLI flags
- GUI options
- Documentation completeness

Version History

Version 1.2.0 (2026-01-13)

- ✅ Added MPS environment variables for stability
- ✅ Implemented automatic NaN error detection
- ✅ Implemented automatic CPU fallback mechanism
- ✅ Added `--force-cpu` CLI flag
- ✅ Added Force CPU checkbox in GUI
- ✅ Added user warnings about MPS issues
- ✅ Created comprehensive test suite
- ✅ Updated documentation

Support

If you continue to experience issues:

1. Try `--force-cpu` mode
2. Run with `--debug` flag and share output
3. Report the issue with:
   - Your Mac model (M1/M2/M3)

- Audio file characteristics
- Full error message
- Debug output

Conclusion

The MPS NaN error is a known issue, but with automatic fallback, it should be **completely transparent** to users. The tool will handle the error automatically and complete your transcription successfully using CPU.

For users who want maximum reliability, the `--force-cpu` option provides a way to avoid GPU entirely.