Contents

1 GPU in UDF Implementation Summary

1.1 What Was Implemented

Added optional GPU support for RNN model predictions in Spark executor UDFs. By default, predictions run in CPU-only mode for maximum compatibility, but users can now enable GPU acceleration by setting

use_gpu_in_udf=True.

1.2 Changes Made

1.2.1 1. skol_classifier/rnn_model.py - Added use_gpu_in_udf Parameter

Modified __init__ signature:

```
def __init__(
    self,
    # ... existing parameters ...
    use_gpu_in_udf: bool = False  # NEW: Enable GPU in UDFs
):
```

Added parameter documentation (lines 440-450):

```
use_gpu_in_udf: Whether to allow GPU usage in Spark executor UDFs during predict
                Default: False (CPU-only mode for compatibility).
                Set to True to enable GPU in UDFs if:
                - Worker nodes have GPUs with proper CUDA/TensorFlow setup
                - GPU memory is sufficient for batch predictions
                - You've tested that CUDA initialization works in executors
                WARNING: GPU usage in UDFs can cause CUDA errors if:
                - Worker GPUs are incompatible with TensorFlow version
                - Multiple executors try to use the same GPU
                - GPU memory is insufficient
                Only enable if you've verified GPU availability on workers.
```

Stored parameter as instance variable (line 480):

```
self.use_gpu_in_udf = use_gpu_in_udf
```

1.2.2 2. skol_classifier/rnn_model.py - Added GPU Warning

Added warning message when GPU is enabled (lines 482-497):

```
# Warn if GPU is enabled in UDFs
if self.use_gpu_in_udf and self.verbosity >= 1:
    print("\n" + "="*70)
    print("WARNING: GPU enabled in Spark executor UDFs")
    print("="*70)
    print("GPU usage in distributed UDFs can cause issues:")
    print("  - CUDA initialization errors if GPUs are incompatible")
    print("  - Memory errors if GPU memory is insufficient")
    print("  - Conflicts if multiple executors share a GPU")
    print("")
    print("Only use this if you've verified:")
```

```
    print("  ✓ All worker nodes have compatible GPUs")
    print("  ✓ CUDA and TensorFlow are properly configured")
    print("  ✓ GPU memory is sufficient for batch predictions")
    print("  ✓ Executor GPU assignment is properly configured")
    print("="*70 + "\n")
```

1.2.3 3. skol_classifier/rnn_model.py - Pass Flag to UDF

Captured flag in UDF closure (line 1044):

```
use_gpu_in_udf = self.use_gpu_in_udf  # Capture for UDF closure
```

Added to debug logging (line 1034):

```
print(f"[RNN Predict Proba] GPU in UDF: {self.use_gpu_in_udf}")
```

1.2.4 4. skol_classifier/rnn_model.py - Conditional GPU Configuration in UDF

Replaced hardcoded CPU-only mode with conditional logic:

Before (lines 1053-1062):

```
# Force CPU-only mode in executors to prevent CUDA errors
os.environ['CUDA_VISIBLE_DEVICES'] = ''
log("[UDF PROBA] Set CUDA_VISIBLE_DEVICES to empty string")

# Import TensorFlow/Keras inside UDF after setting CPU mode
try:
    import tensorflow as tf
    from tensorflow import keras
    # Double-check GPU is disabled
    tf.config.set_visible_devices([], 'GPU')
    log("[UDF PROBA] TensorFlow imported and GPU disabled")
```

After (lines 1085-1117):

```
# Conditionally configure GPU based on use_gpu_in_udf flag
if not use_gpu_in_udf:
    # Force CPU-only mode in executors to prevent CUDA errors
    os.environ['CUDA_VISIBLE_DEVICES'] = ''
    log("[UDF PROBA] Set CUDA_VISIBLE_DEVICES to empty string (CPU-only mode)")
else:
    log("[UDF PROBA] GPU enabled - will attempt to use GPU if available")

# Import TensorFlow/Keras inside UDF
try:
    import tensorflow as tf
    from tensorflow import keras
```

```
    if not use_gpu_in_udf:
        # Double-check GPU is disabled for CPU-only mode
        tf.config.set_visible_devices([], 'GPU')
        log("[UDF PROBA] TensorFlow imported and GPU disabled")
    else:
        # Configure GPU with memory growth if available
        gpus = tf.config.list_physical_devices('GPU')
        if gpus:
            try:
                for gpu in gpus:
                    tf.config.experimental.set_memory_growth(gpu, True)
                log(f"[UDF PROBA] TensorFlow imported, {len(gpus)} GPU(s) availa
            except Exception as e:
                log(f"[UDF PROBA WARNING] GPU memory growth configuration failed
        else:
            log("[UDF PROBA] TensorFlow imported, no GPUs detected - will use CP
```

Key features: - Conditionally sets `CUDA_VISIBLE_DEVICES` based on flag - Detects available GPUs when enabled - Enables memory growth to prevent OOM errors - Gracefully falls back to CPU if no GPU available - Provides detailed logging for debugging

1.2.5 5. skol_classifier/model.py - Updated Factory Function

Added parameter passing in `create_model()` for RNN (lines 338-340):

```
# Add 'use_gpu_in_udf' parameter if provided
if 'use_gpu_in_udf' in model_params:
    rnn_params['use_gpu_in_udf'] = model_params['use_gpu_in_udf']
```

Updated docstring (lines 261, 279-282):

```
**model_params: Additional model parameters
                Can include:
                - 'class_weights': dict mapping label strings to weights
                - 'focal_labels': list of label strings for F1-based loss (RNN on
                - 'use_gpu_in_udf': bool to enable GPU in RNN UDFs (default: Fals
                # ...

GPU in RNN UDFs:
- Set 'use_gpu_in_udf=True' to enable GPU usage in Spark executor UDFs
- Only use if worker nodes have compatible GPUs with proper CUDA setup
- Default is False (CPU-only) for maximum compatibility
```

1.2.6 6. Documentation Created

1.2.6.1 docs/GPU_IN_UDF.md Comprehensive guide covering: - Quick start examples - When to use GPU in UDFs - Default behavior vs GPU-enabled mode - Requirements (hardware, software, Spark config) - Performance considerations - Troubleshooting common issues - Best practices - CPU vs GPU comparison table - Example hybrid deployment

1.2.6.2 examples/example_gpu_in_udf.py Example script with 3 scenarios: 1. CPU-only mode (default, always works) 2. GPU-enabled mode (requires GPU setup) 3. Performance comparison (CPU vs GPU)

1.3 How It Works

1.3.1 Default Behavior (CPU-Only Mode)

When `use_gpu_in_udf=False` (default):

1. **UDF initialization**: Sets `CUDA_VISIBLE_DEVICES=''`
2. **TensorFlow import**: Calls `tf.config.set_visible_devices([], 'GPU')`
3. **Prediction**: All inference runs on CPU
4. **Compatibility**: Works on any cluster (CPU or GPU)

1.3.2 GPU-Enabled Mode

When `use_gpu_in_udf=True`:

1. **UDF initialization**: Does NOT set `CUDA_VISIBLE_DEVICES`
2. **TensorFlow import**: Detects available GPUs
3. **GPU configuration**: Enables memory growth on each GPU
4. **Prediction**: Uses GPU if available, falls back to CPU otherwise
5. **Logging**: Reports GPU detection status

1.4 Usage Examples

1.4.1 Basic Usage

```
from skol_classifier.model import create_model

# Default: CPU-only mode (safe)
model_cpu = create_model(
    model_type='rnn',
    input_size=2000,
    use_gpu_in_udf=False  # or omit (default)
)
```

```
# GPU-enabled mode (requires GPU on workers)
model_gpu = create_model(
    model_type='rnn',
    input_size=2000,
    use_gpu_in_udf=True,  # Enable GPU
    prediction_batch_size=128  # Larger batch for GPU
)
```

1.4.2 Via SkolClassifierV2

```
from skol_classifier.classifier_v2 import SkolClassifierV2

classifier = SkolClassifierV2(
    spark=spark,
    input_source='files',
    file_paths=['data/*.txt'],
    model_type='rnn',

    # RNN parameters
    input_size=2000,
    hidden_size=256,
    window_size=50,

    # Enable GPU in prediction UDFs
    use_gpu_in_udf=True,
    prediction_batch_size=128,

    verbosity=2
)

predictions = classifier.predict()
```

1.4.3 Hybrid Deployment

```
# Training: CPU mode (runs on driver)
train_classifier = SkolClassifierV2(
    spark=spark,
    file_paths=['data/train/*.ann'],
    model_type='rnn',
    use_gpu_in_udf=False,  # CPU for training
    verbosity=2
)
train_classifier.fit_and_save_model()
```

```
# Inference: GPU mode (runs on workers)
pred_classifier = SkolClassifierV2(
    spark=spark,
    file_paths=['data/new/*.txt'],
    model_type='rnn',
    auto_load_model=True,
    use_gpu_in_udf=True,  # GPU for inference
    prediction_batch_size=128,
    verbosity=2
)
predictions = pred_classifier.predict()
```

1.5 Performance Impact

1.5.1 Expected Speedup

Based on model complexity: - **Small models** (1-2 layers, 64-128 hidden): 2-5x faster - **Medium models** (2-3 layers, 128-256 hidden): 5-10x faster - **Large models** (3+ layers, 256+ hidden): 10-20x faster

1.5.2 Memory Considerations

GPU memory usage = Model size + (Batch size × Window size × Input size × 4 bytes)

Example: - Model: 256 hidden, 3 layers ≈ 10 MB - Batch: 128 windows - Window: 50 lines - Input: 2000 features - Total ≈ 10 MB + 128 × 50 × 2000 × 4 bytes ≈ 60 MB per batch

Recommendation: Keep total GPU memory usage under 80% of available memory.

1.6 Backward Compatibility

✅ **Fully backward compatible** - Default behavior unchanged (CPU-only mode) - Existing code works without modifications - `use_gpu_in_udf=False` is the default - No breaking changes to existing APIs

1.7 Testing

The implementation includes:

1. **Automatic GPU detection**: Checks for GPUs and logs availability
2. **Graceful degradation**: Falls back to CPU if GPU unavailable

3. **Memory growth**: Prevents OOM errors on GPU
4. **Comprehensive logging**: Detailed messages at verbosity >= 2
5. **Example scripts**: Demonstrates CPU and GPU modes

1.8 Requirements for GPU Mode

1.8.1 Hardware

- NVIDIA GPU on worker nodes
- Compatible CUDA compute capability
- Sufficient GPU memory (4GB+ recommended)

1.8.2 Software

- CUDA toolkit (version compatible with TensorFlow)
- cuDNN library
- TensorFlow GPU version

1.8.3 Spark Configuration

```
spark = SparkSession.builder \
    .config("spark.executor.resource.gpu.amount", "1") \
    .config("spark.task.resource.gpu.amount", "1") \
    .getOrCreate()
```

1.9 Common Issues and Solutions

1.9.1 Issue 1: CUDA Error

Symptom: `CUDA_ERROR_INVALID_HANDLE` **Solution**: Verify GPU compatibility, check CUDA version, or use CPU mode

1.9.2 Issue 2: Out of Memory

Symptom: `Resource exhausted: OOM when allocating tensor` **Solution**: Reduce `prediction_batch_size` or use smaller model

1.9.3 Issue 3: No GPU Detected

Symptom: Logs show "no GPUs detected" **Solution**: Check `nvidia-smi`, verify CUDA setup, ensure GPU resources allocated to executor

1.10 See Also

- **GPU in UDF User Guide** - Complete usage guide
- **RNN Model Documentation** - RNN implementation
- **Example Script** - Working examples
- **TensorFlow GPU Guide** - TensorFlow GPU setup

Implementation Date: 2025-12-20 **Files Modified**: - skol_classifier/rnn_model.py - skol_classifier/model.py

Files Created: - docs/GPU_IN_UDF.md - docs/GPU_UDF_IMPLEMENTATION_SUMMARY.md - examples/example_gpu_in_udf.py

Status: ✅ Complete and documented