Comprehensive LLM Self-Assessment Evaluation

Parameter	Details
Prompt	You are a deep learning engineer specializing in optimizing training performance. I'm running into performance issues with my CNN model training—specifically high CPU usage. Could you: Analyze potential reasons why my model is experiencing high CPU usage. Recommend strategies for migrating to GPU acceleration efficiently. Suggest debugging steps to identify potential hardware bottlenecks. I'm especially interested in understanding the trade-offs between different approaches so I can make an informed decision for my specific setup.
Prompt Type	Role-Based Prompt

Parameter	Details
Answer	To optimize the performance of your CNN model and address high CPU usage, here's an in-depth analysis and recommendations: ### **1. Analyzing Potential Reasons for High CPU Usage** High CPU usage during CNN training can occur due to several factors: #### **a. Data Preprocessing - Cause**: If your data is being preprocessed on the CPU while training happens on the GPU, the CPU can become a bottleneck, especially when working with large datasets or complex transformations (e.g., resizing, augmentation). - **Solution**: Move data preprocessing to the GPU or use multi-threading/multiprocessing techniques to offload the preprocessing to multiple CPU cores. **TensorFlow** and **PyTorch** allow you to parallelize data loading using `tf.data` (TensorFlow) or `DataLoader` (PyTorch) with `num_workers` and pin memory options. #### **b. Small Batch Sizes - Cause**: Small batch sizes reduce GPU memory utilization, which can lead to the CPU being responsible for most of the work. This can result in high CPU usage since it's handling both model computation and the data pipeline. - **Solution**: Increase batch size (within the limits of your GPU memory) to ensure the GPU is being used effectively. #### **c. Inadequate GPU Utilization - Cause**: If the GPU is not properly utilized (due to incorrect TensorFlow or PyTorch settings), the CPU can end up doing most of the work. This could happen due to improper device placement or small model sizes that don't fully leverage the GPU. - **Solution**: Ensure that operations are explicitly assigned to the GPU. Use device placement logging to check if your operations are running on the GPU. If not, use `tf.device('/GPU:0')` (for

Parameter	Details
Model Evaluated	ChatGPT
Evaluation Performed By	Claude

Core Self-Assessment Metrics

Metric	Score (1-10)	Interpretation	Key Evidence
Confidence-Performance Correlation	8	Excellent	Strong alignment between confident tone and accurate technical information. No explicit numerical confidence statements, but implicit confidence is well-aligned with factual accuracy.
Calibration Error	7	Good	Answer provides technically accurate information without hedging on well-established facts. No observable instances of overconfidence or underconfidence on technical content.
Task Difficulty Awareness	8	Very Good	Clear recognition of complexity in GPU migration. Thorough discussion of trade-offs shows awareness of implementation challenges.

Metric	**Score (1-10)**	**Interpretation**	**Key Evidence**
Error Recognition	6	Above Average	Notes potential issues (e.g., "numerical instability in mixed precision") but lacks explicit acknowledgment of knowledge boundaries.
Domain-Specific Variance	7	Good	Demonstrates strong domain knowledge across TensorFlow and PyTorch frameworks, with appropriate variation in confidence between established GPU practices and implementation specifics.
Prompt Sensitivity	N/A	N/A	Single prompt evaluation; cannot assess prompt sensitivity.
Weighted Self-Assessment Score	**7.3**	**Good**	WSAS = (8×0.25) + (7×0.25) + (8×0.15) + (6×0.15) + (7×0.1) + (N/A×0.1)

Technical Accuracy Assessment

Category	**Accuracy**	**Notes**
Factual Claims	95%	38/40 factual claims are correct. Minor imprecisions in TensorFlow profiler API syntax (current API differs slightly).

Category	Accuracy	Notes
Procedural Recommendations	100%	18/18 recommendations are valid approaches to solve the stated problem.
Inferences/Opinions	100%	12/12 inferences about trade-offs and advantages/disadvantages are reasonable and supported by standard ML engineering practices.
Overall Accuracy	97%	Exceptionally high accuracy across all technical content.

Self-Assessment Classification

Primary Classification	Expertly Calibrated
Secondary Classifications	Domain Sensitive: Shows appropriate depth in ML/DL engineering knowledgeComplexity Aware: Clearly articulates trade-offs and implementation challengesError Conscious: Mentions potential pitfalls in approachesReasoning Transparent: Clearly explains rationale for recommendations

Confidence Expression Analysis

Type	Count	Examples	Average Confidence Level
Explicit Confidence Statements	0	None present	N/A
Certainty Markers	12	“can”, “will”, “significantly”	80%

Type	Count	Examples	Average Confidence Level
Hedge Words	3	“may”, “could”, “can arise”	50%
Qualifying Phrases	7	“especially when”, “if you’re using”, “if not managed properly”	70%
Overall Estimated Confidence			**75%**

Metacognitive Strategies

Strategy	Presence	Effectiveness
Knowledge boundary articulation	Limited	Medium
Confidence calibration	Medium	High
Reasoning transparency	Strong	High
Alternative consideration	Strong	High
Information source qualification	None	N/A
Temporal qualification	None	N/A
Logical qualification	Medium	High
Uncertainty decomposition	Medium	Medium

Key Improvement Recommendations

1. Include explicit confidence statements where appropriate, especially for newer techniques or when discussing implementation challenges.
2. Articulate knowledge boundaries more clearly, particularly regarding hardware-specific optimizations or newer GPU architectures.
3. Incorporate information about the recency and reliability of recommendations, especially given the evolving nature of deep learning frameworks.
4. Add more specific guidance about diagnosing which of the potential causes is most likely in the user’s specific scenario.
5. Include brief mentions of the source or empirical basis for specific recommendations to enhance credibility.