NVIDIA-Certified Professional: Gen AI LLMs

NVIDIA-Certified Professional: Gen AI LLMs

Contents

This study guide provides an overview of each topic covered on the NVIDIA Certified Professional-Gen AI LLMs certification exam, recommended training, and suggested reading to prepare for the exam.

Job Description

Generative AI practitioners have hands-on experience in large language models (LLMs) and a solid understanding of model development, optimization, and deployment at scale. In this role, you'll design, train, and fine-tune cutting-edge LLMs, applying advanced distributed training techniques and optimization strategies to deliver high-performance AI solutions. You'll collaborate across research and production teams to ensure models are robust, efficient, and scalable.

Key Responsibilities

Model Development and Training

- Pretrain and fine-tune foundation models for both research and production use cases.
- Apply parameter-efficient fine-tuning (e.g., low-rank adaption, or LoRA) and optimization techniques such as knowledge distillation, pruning, quantization, and artifact simplification.
- Architect and implement distributed training strategies (data, model, tensor, and expert parallelism).

Evaluation and Optimization

- Develop and apply rigorous evaluation methods, combining quantitative metrics (BLEU, ROUGE, perplexity, and LLM-as-a-judge) with qualitative assessments (human-in-the-loop reviews, error analysis).
- Profile and optimize model and CUDA® kernel performance, diagnosing GPU bottlenecks and improving efficiency for both inference and training.
- Benchmark and troubleshoot large-scale deployments across multi-GPU, cloud, and on-premises systems.

Deployment and Scaling

- Deploy models in production using containerization (Docker) and orchestration (Kubernetes).
- Optimize inference for low latency, high throughput, and edge compatibility.
- Extend and maintain Python-based machine learning (ML) workflows, with targeted low-level optimizations in C++ when required.

Innovation and Research

- Design experiments to validate fine-tuning, evaluation, and optimization methods to ensure statistically sound results.
- Address real-world challenges such as CUDA memory allocation, kernel utilization, and distributed training scalability.
- Stay ahead of advances in generative AI, transformer architectures, and emerging NVIDIA technologies.

Recommended Qualifications and Experience

- 2–3 years of applied experience in AI/ML with a focus on LLMs
- Strong understanding of transformer-based architectures (self-attention, encoder-decoder, positional encoding)
- Familiarity with retrieval-augmented generation (RAG), hallucination mitigation, and advanced sampling techniques
- Proven experience with distributed parallelism and parameter-efficient fine-tuning workflows
- Proficiency in model evaluation metrics, performance profiling, and optimization
- Strong coding skills in Python, with the ability to implement performance-critical components in C++ a plus
- Experience with Docker and Kubernetes for scalable deployments
- Familiarity with NVIDIA's ecosystem (NGC™ catalog, Base Command™ Platform, DGX™ systems, AI Enterprise suite) is advantageous

Certification Topics and References

LLM Architecture :	Exam Weight 6%
Understanding and applying foundational LLM structures and mechanisms.	
1.1	Analyze encoder-decoder structures and their applications.
1.2	Describe transformer architectures including self-attention mechanisms.
1.3	Develop code to extract embeddings from both encoder and decoder models.
1.4	Implement advanced sampling techniques for text generation.
1.5	Understand output sampling techniques used in decoder-based language models.
1.6	Understand the concept of embeddings.

Recommended NVIDIA Courses and Suggested Readings

NVIDIA Courses

> Rapid Application Development With LLMs
> Building Transformer-Based NLP Applications

Suggested Readings

> Mastering LLM Techniques: Training | NVIDIA Blog
> Attention Is All You Need | arXiv

Prompt Engineering: Exam Weight 13%

Adapting LLMs to new domains, tasks, or data distributions via prompt engineering, chain-of-thought (CoT), domain adaptation, zero/one/few-shot learning, and output control.

2.1	Engineer effective prompts and templates, including chain-of-thought and prompt learning for small datasets or specialized domains.
2.2	Employ zero-shot, one-shot, and few-shot techniques to expand model adaptability.
2.3	Train decoder-based LLMs with causal language modeling as needed.
2.4	Design specialized LLM-wrapping modules with built-in validation and constrained decoding for improved consistency, reduced hallucinations, and better user experience.

Recommended NVIDIA Courses and Suggested Readings

NVIDIA Courses

> Building LLM Applications With Prompt Engineering
> Building RAG Agents With LLMs

Suggested Readings

> Mastering LLM Techniques: Inference Optimization | NVIDIA Technical Blog
> An Easy Introduction to LLM Reasoning, AI Agents, and Test-Time Scaling | NVIDIA Technical Blog
> Train a Reasoning-Capable LLM in One Weekend With NVIDIA NeMo™ | NVIDIA Technical Blog
> Multi-Turn Conversational Chat Bot NVIDIA Generative AI Examples 0.5.0 Documentation

Data Preparation: Exam Weight 9%

Preparing data for pretraining, fine-tuning, or inference by cleaning, curating, analyzing, and organizing datasets, tokenization, and vocabulary management.

3.1	Clean and curate data (handle missing, normalize, scale), and analyze class imbalances and feature distributions.
3.2	Organize datasets, ensure correct formats, and prepare data for modeling.
3.3	Select and train tokenizers and optimize tokenization strategies and vocabulary size (BPE and WordPiece) to fit tasks and resources.

Recommended NVIDIA Courses and Suggested Readings

NVIDIA Courses

> Adding New Knowledge to LLMs
> Building Transformer-Based NLP Applications

Suggested Readings

> NVIDIA/GenerativeAIExamples | GitHub
> NVIDIA AI Workbench Example Projects
> Tokenizers NVIDIA NeMo Framework User Guide
> NVIDIA /NeMo | GitHub
> Train Models Using a Distributed Training Workload
> Speed Up Data Exploration With NVIDIA RAPIDS™ cuDF | NVIDIA Technical Blog
> Accelerating Time- Series Forecasting With RAPIDS cuML | NVIDIA Technical Blog
> Model Fine-Tuning NVIDIA NeMo Framework User Guide 24.07
> NeMo Curator | NVIDIA Developer
> Faster Causal Inference on Large Datasets With NVIDIA RAPIDS | NVIDIA Technical Blog
> Fine-Tuning NVIDIA NeMo Framework User Guide

Model Optimization: Exam Weight 17%

Using model optimization strategies for large language models, such as pruning, quantization, and knowledge distillation, to reduce memory, accelerate inference, make models compatible with GPU acceleration, and deploy them efficiently.

4.1	Apply pruning, sparsity, and weight/activation quantization to reduce memory footprint and optimize model inference for hardware acceleration.
4.2	Choose and implement quantization strategies (post-training, quantization-aware, activation quantization) tailored for hardware and tasks (e.g., NVIDIA A100/H100 Tensor Core GPUs, FP16, INT8), and measure any accuracy trade-offs.
4.3	Implement knowledge distillation to create smaller, efficient models based on larger pretrained ones.
4.4	Conduct systematic hyperparameter tuning and distributed parameter search, including learning rate schedules and batch size adjustments.
4.5	Use advanced sampling (beam search, temperature scaling) and systematic ablation studies to evaluate model optimization impact.
4.6	Select and apply optimization methods (NVIDIA TensorRT™, sliding-window/streaming attention, key-value caching) based on architecture, task, and available resources.
4.7	Train encoder-based foundation LLMs with masked language modeling (MLM) and/or next sentence prediction, and understand quantization, distillation, and model pruning concepts.

Recommended NVIDIA Courses and Suggested Readings

NVIDIA Courses

> Building Transformer-Based NLP Applications
> Adding New Knowledge to LLMs
> Deploying RAG Pipelines for Production at Scale
> Model Parallelism: Building and Deploying Large Neural Networks

Suggested Readings

> Best Practices for TensorRT Performance | NVIDIA Documentation
> Quantization NVIDIA NeMo User Guide
> DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper, and Lighter | arXiv
> What Is Knowledge Distillation? | IBM
> Sparsity in INT8: Training Workflow and Best Practices for NVIDIA TensorRT Acceleration | NVIDIA Technical Blog
> Quantization and Calibration NVIDIA TensorRT Documentation
> Post-Training Quantization vs. Quantization-Aware Training | Fiveable
> Quantization-Aware Training (QAT) vs. Post-Training Quantization (PTQ) | Better ML (Medium)
> The Illustrated Transformer | Jay Alammar
> Understanding Data Types in AI and HPC: INT8, FP8, FP16, BF16, BF32, FP32, TF32, FP64, and Hardware Accelerators | It's About AI
> Quantization: What You Should Understand if You Want to Run LLMs | Pavan Mantha, LinkedIn
> Capabilities NVIDIA TensorRT Documentation
> LoRA: Low-Rank Adaptation of Large Language Models | OpenReview
> GPTQ: Accurate Post-Training Quantization for Generative Pretrained Transformers | arXiv

Fine-Tuning: Exam Weight 13%

Customizing pretrained LLMs for downstream tasks or domains using parameter-efficient methods, human feedback, contrastive learning, and robust evaluation.

5.1	Align models with intent via supervised fine-tuning or reinforcement learning from human feedback, including methods like direct preference optimization (DPO) or group relative policy optimization (GRPO).
5.2	Apply contrastive loss for embeddings and use parameter-efficient techniques (LoRA, adapters, P-tuning).
5.3	Implement early stopping to prevent overfitting and select performance metrics for all phases.
5.4	Mitigate hallucinations, assess fine-tuning impact, and perform parameter-efficient updates for LLMs.

Recommended NVIDIA Course and Suggested Readings

NVIDIA Course

> Adding New Knowledge to LLMs

Suggested Readings

- Deploy Diverse AI Apps With Multi-LoRA Support on NVIDIA RTX™ AI PCs and Workstations | NVIDIA Blog
- Selecting Large Language Model Customization Techniques | NVIDIA Technical Blog
- LoRA: Low-Rank Adaptation of Large Language Models. | arXiv
- Parameter-Efficient Fine-Tuning for LLMs With NVIDIA NeMo | NVIDIA Technical Blog
- Prevent LLM Hallucinations With the Cleanlab Trustworthy Language Model in NVIDIA NeMo Guardrails | NVIDIA Technical Blog
- Chapter 7, Regularization for Deep Learning— Deep Learning: An MIT Press Book (Goodfellow, Bengio, Courville)
- LLM Evaluation Metrics: BLEU, ROUGE, and METEOR Explained | Medium

Evaluation: Exam Weight 7%

Assessing LLMs via quantitative and qualitative metrics, framework design, benchmarking, error analysis, and scalable evaluation.

6.1	Analyze benchmark results, conduct human-in-the-loop and LLM-as-a-judge evaluations, and assess model quality using key metrics (BLEU, ROUGE, Perplexity).
6.2	Diagnose LLM failure modes and perform systematic error analysis to identify common behavioral and output patterns.
6.3	Benchmark and compare LLM deployments across various platforms (on-prem DGX, cloud GPUs) using standardized evaluation metrics.
6.4	Design and implement comprehensive evaluation frameworks integrating all the above practices for robust and scalable model assessment.

Recommended NVIDIA Course and Suggested Readings

NVIDIA Course

> Deploying RAG Pipelines in Production at Scale

Suggested Readings

- NVIDIA Metrics | Ragas
- Retrieval-Augmented Generation (RAG) Pipeline | NVIDIA Docs
- Evaluating Medical RAG With NVIDIA AI Endpoints and Ragas | NVIDIA Technical Blog
- Integrations | Ragas
- NVIDIA-AI-Blueprints/RAG | GitHub
- NeMo Evaluator | NVIDIA Developer
- Introduction—NVIDIA Autonomous Vehicles Safety Report
- Performance Analysis Tools | NVIDIA Developer
- Deployment Best Practices—NVIDIA RTX vWS: Sizing and GPU Selection Guide for Virtualized Workloads
- Troubleshoot NVIDIA NIM for LLMs
- How the DGX H100 Accelerates AI Workloads | CUDO Compute
- Masked Language Model Scoring | arXiv
- Perplexity of Fixed-Length Models | Hugging Face
- The Importance of Starting With Error Analysis in LLM Applications | Shekhar Gulati

GPU Acceleration and Optimization: Exam Weight 14%

Scaling and optimizing LLM training and inference on GPU hardware. Involves multi-GPU/distributed setups, parallelism techniques, troubleshooting, memory and batch optimization, and performance profiling.

7.1	Configure multi-GPU and distributed training setups (DDP, FSDP, model, pipeline, tensor, data, sequence, and expert parallelism).
7.2	Apply Tensor Core and mixed-precision optimizations and batch/memory management for efficient throughput.
7.3	Distribute and optimize self-attention head general matrix multiplication (GEMM) operations and implement gradient accumulation for large models or limited GPU memory.
7.4	Identify and address bottlenecks using CUDA profiling and troubleshoot memory and kernel efficiency issues.

Recommended NVIDIA Courses and Suggested Readings

NVIDIA Courses

> Optimizing CUDA Machine Learning Codes With NVIDIA Nsight™ Profiling Tools

> Model Parallelism: Building and Deploying Large Neural Networks

Suggested Readings

> Distributed Data Parallel in PyTorch

> CUDA C++ Best Practices Guide 13.0

> Assess, Parallelize, Optimize, Deploy | NVIDIA Blog

> Parallelisms — NVIDIA NeMo User Guide

> Batching — NVIDIA NeMo Developer Docs

> PyTorch Lightning: Gradient Accumulation | GitHub

> Accelerate Usage Guides: Gradient Accumulation | Hugging Face

Model Deployment: Exam Weight 9%

Deploying LLMs in production via containerized pipelines, scalable orchestration, efficient batch and model serving, and real-time monitoring.

8.1	Analyze computational tradeoffs for model types (encoder, decoder, encoder-decoder) and optimize for memory and latency.
8.2	Build containerized inference pipelines, use dynamic batching, and deploy with NVIDIA Dynamo-Triton.
8.3	Configure and manage serving (Kubernetes, ensemble workflows), implement live monitoring, and run models in Docker.

Recommended NVIDIA Courses and Suggested Readings

NVIDIA Courses

> Deploying RAG Pipelines in Production at Scale

> Building RAG Agents With LLMs

Suggested Readings

> NVIDIA NIM Microservices for Accelerated AI Inference

> Overview of NVIDIA NIM for Large Language Models (LLMs)

> Power Your AI Projects With New NVIDIA NIMs for Mistral and Mixtral Models | NVIDIA Blog

> Concurrent Model Execution—Dynamo-Triton (Previously NVIDIA Triton™ Inference Server) User Guide

> Model Configuration—Dynamo-Triton (Previously Triton Inference Server) User Guide

> Schedulers—Dynamo-Triton (Previously Triton Inference Server) User Guide

> Batchers—Dynamo-Triton (Previously Triton Inference Server) User Guide

Production Monitoring and Reliability: Exam Weight 7%

Establishing monitoring dashboards and reliability metrics while tracking logs and anomalies for root-cause analysis. Evaluating benchmarking agents against prior versions. Implementing automated tuning, retraining, and versioning to ensure continuous uptime, transparency, and trust in production deployments.

9.1	Define monitoring dashboards and reliability metrics.
9.2	Track logs, errors, and anomalies for root-cause diagnosis.
9.3	Continuously benchmark deployed agents against prior versions.
9.4	Implement automated tuning, retraining, and versioning in production.
9.5	Ensure continuous uptime, transparency, and trust in live deployments.

Recommended NVIDIA Courses and Suggested Readings

NVIDIA Course

> Deploying RAG Pipelines in Production at Scale

Suggested Readings

> Attention I All You Need—arXiv

> BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding—arXiv

> Improving Language Understanding by Generative Pretraining (Radford, Narasimhan, Salimans, Sutskever) | OpenAI

> Masked Language Modeling Guide | Hugging Face

> Causal Language Modeling Guide | Hugging Face

Safety, Ethics, and Compliance: Exam Weight 5%

Practicing responsible AI practices throughout the LLM lifecycle. Includes auditing for bias and fairness, implementing guardrails, configuring monitoring for ethical compliance, and applying bias detection and mitigation strategies to ensure responsible deployment and use of LLMs.

10.1 Apply responsible AI practices to model deployment.

10.2 Audit LLMs for bias and fairness.

10.3 Configure monitoring systems for production LLMs.

10.4 Implement bias detection and mitigation strategies.

10.5 Implement guardrails to restrict undesired LLM responses.

Recommended NVIDIA Courses and Suggested Readings

NVIDIA Courses

> Building RAG Agents With LLMs
> Building LLM Applications With Prompt Engineering
> Deploying RAG Pipelines in Production at Scale

Suggested Readings

> Build an Enterprise RAG Pipeline Blueprint | build.nvidia.com
> RAG 101: Demystifying Retrieval-Augmented Generation Pipelines | NVIDIA Technical Blog
> Full-Stack Observability for NVIDIA Blackwell and NIM-Based AI | Dynatrace
> Large-Scale Production Deployment of RAG Pipelines | NVIDIA Deep Learning Institute
> Observability Tool NVIDIA Generative AI Examples 0.5.0 | GitHub
> NVIDIA-AI-Blueprints/RAG | GitHub
> Measuring the Effectiveness and Performance of AI Guardrails in Generative AI Applications | NVIDIA Technical Blog

Questions?

Contact us here.