

Seminar Report

LLM Inference Engines: A Study of NVIDIA Nemotron 3 Architecture

Praveen babu Geddada

MatrNr: 22774253

Supervisor: Jonathan Decker

Georg-August-Universität Göttingen
Institute of Computer Science

September 30, 2026

Abstract

Large Language Models (LLMs) have become increasingly large, making efficient inference an important challenge. NVIDIA Nemotron 3 Nano addresses this challenge by combining Mamba-2, Transformer attention, and Mixture-of-Experts (MoE) in a hybrid architecture. This report studies how these components work together and how they are intended to improve inference efficiency.

The main focus is on the theoretical design of Nemotron 3 Nano, with particular attention to Mamba-2 and its State Space Duality (SSD) formulation, the use of attention in selected layers, and sparse expert routing in the MoE layers. The report also discusses how these design choices affect computation, memory usage, and long-context processing.

As a practical case study, Nemotron 3 Nano is evaluated against Qwen3-30B-A3B-Instruct-2507, which has a similar total and active parameter count. Both models are tested on an NVIDIA A100 80GB GPU using the same inference setup. The measurements include generation throughput, generation latency, peak GPU memory, and model loading time.

The results show that measured inference performance depends on the model architecture, hardware, and workload. The benchmark is used to support the theoretical analysis rather than to establish a general performance ranking. Overall, the report examines how the combination of Mamba-2, attention, and MoE forms the main architectural approach behind Nemotron 3 Nano and its focus on efficient inference.

I hereby declare that all uses of tools and AI as well as assistance from third parties in this work have been fully and accurately disclosed in the Transparency Report in Appendix Section ??.

Missing or incorrect information may be considered an attempt to cheat.

Contents

List of Tables	iv
List of Figures	iv
List of Abbreviations	v
1 Introduction	1
2 Background	2
2.1 Transformer Architecture	2
2.2 Self-Attention	2
2.3 Language Model Inference	2
2.4 Key-Value Cache	3
2.5 State Space Models	3
2.6 Mamba	3
2.7 Mamba-2 and State Space Duality	4
3 Mixture-of-Experts	4
3.1 Introduction to Mixture-of-Experts	4
3.2 Sparse Expert Routing	4
3.3 Total and Active Parameters	5
3.4 Advantages and Limitations	5
4 Nemotron 3 Nano Architecture	5
4.1 Architecture Overview	5
4.2 Hybrid Mamba-2 and Attention	5
4.3 Mamba-2 Layers	6
4.4 Attention and Grouped-Query Attention	6
4.5 MoE Layers	6
4.6 Long-Context Processing	6
4.7 Why the Architecture Can Help Inference	7
5 Comparison with Qwen3-30B-A3B	7
5.1 Reason for the Comparison	7
5.2 Architectural Differences	7
6 Experimental Methodology	8
6.1 Hardware	8
6.2 Software	8
6.3 Models	8
6.4 Benchmark Prompts	8
6.5 Evaluation Metrics	8
6.6 Benchmark Procedure	9

7	Experimental Results	9
7.1	Overall Results	9
7.2	Generation Throughput	10
7.3	Generation Latency	10
7.4	GPU Memory	11
7.5	Model Loading Time	11
7.6	Long-Context Evaluation	12
8	Architecture-Performance Discussion	13
8.1	Effect of the Workload	14
8.2	Architecture and Implementation	14
8.3	Comparison with Published Results	14
8.4	Role of Mamba-2, MoE, and Attention	14
8.5	Discussion	15
9	Limitations and Future Work	15
10	Conclusion	16
	References	17
A	Transparency Report	A1
A.1	Literature and External Sources	A1
A.2	Software and Tools	A1
A.3	Benchmark Data	A1
A.4	AI Assistance	A1
A.5	Use of Supervisor Feedback	A1
A.6	Limitations of AI Assistance	A1

List of Tables

1	Main architectural characteristics of the two models.	7
2	Average results from the five-prompt benchmark.	9
3	Nemotron 3 Nano long-context inference results.	12

List of Figures

1	Generation throughput for the five benchmark prompts.	10
2	Generation latency for the five benchmark prompts.	10
3	Peak GPU memory used during inference.	11
4	Generation latency of Nemotron 3 Nano across increasing input context lengths.	12
5	Generation throughput of Nemotron 3 Nano across increasing input context lengths.	13
6	Peak GPU memory usage of Nemotron 3 Nano as a function of input context length.	13

List of Abbreviations

AI	Artificial Intelligence
LLM	Large Language Model
NLP	Natural Language Processing
SSM	State Space Model
SSD	State Space Duality
MoE	Mixture-of-Experts
KV	Key-Value
GQA	Grouped-Query Attention
GPU	Graphics Processing Unit
HPC	High-Performance Computing
CUDA	Compute Unified Device Architecture
BF16	Brain Floating Point 16

1 Introduction

Large Language Models (LLMs) have become an important part of modern artificial intelligence and are used in applications such as text generation, question answering, coding, and summarization [Vas+17]. The capabilities of these models have improved with increasing model size, but larger models also require more computational resources during inference [NVI25]. As a result, reducing inference cost while maintaining model capacity has become an important research problem.

Most recent LLMs are based on the Transformer architecture, which uses self-attention to model relationships between tokens [Vas+17]. Self-attention has been highly successful for language modeling, but its computational and memory requirements become more important as the context length increases. During autoregressive generation, the key-value cache must also grow with the sequence length. These limitations have motivated research into alternative sequence modeling methods that can process long sequences more efficiently.

State Space Models (SSMs) provide one such alternative. Mamba introduced selective state spaces that allow the model to control how information is stored and propagated through a sequence [GD24]. Mamba-2 extended this approach through State Space Duality (SSD), establishing a closer connection between state-space models and attention-based sequence processing [DG24]. These developments have made SSM-based architectures an important direction for efficient sequence modeling.

NVIDIA Nemotron 3 Nano builds on these ideas by combining Mamba-2 layers with Transformer attention and Mixture-of-Experts (MoE) layers [NVI25]. This hybrid design is different from architectures that rely mainly on a single type of sequence-processing layer. Mamba-2 can process sequence information using a compact recurrent state, while attention provides direct interactions between tokens. The MoE component further increases the model’s total capacity while activating only a subset of experts for each token. Understanding how these components work together is important for understanding the design of Nemotron 3 Nano.

This work focuses on the architectural ideas behind Nemotron 3 Nano and their relationship to inference efficiency. In particular, it examines the role of Mamba-2, Transformer attention, and sparse expert routing, and discusses why these components are combined in a single model. The analysis is complemented by a small inference benchmark using Qwen3-30B-A3B-Instruct-2507 as a reference model. Qwen3 has a similar total and active parameter count and also uses a Mixture-of-Experts architecture, making it a more suitable reference than the much smaller Llama 3.2 3B model used in the earlier version of this work.

The main research question addressed in this work is:

How do Mamba-2, Transformer attention, and Mixture-of-Experts contribute to the architecture of Nemotron 3 Nano and its approach to efficient inference?

To address this question, the report first introduces the principles of Transformers, State Space Models, Mamba, and Mamba-2. It then discusses Mixture-of-Experts and examines how these components are combined in Nemotron 3 Nano. Finally, the practical evaluation measures generation throughput, generation latency, peak GPU memory, and model loading time on an NVIDIA A100-SXM4 80GB GPU. The benchmark is used as a

supporting case study for the architectural analysis rather than as a general comparison of model quality.

The remainder of the report is organized as follows. Section 2 introduces Transformers, State Space Models, Mamba, and Mamba-2. Section 3 discusses Mixture-of-Experts and sparse expert routing. Section 4 presents the architecture of Nemotron 3 Nano. Section 5 compares its architecture with Qwen3-30B-A3B. Section 6 describes the experimental setup, and Section 7 presents the benchmark results. Section 8 discusses the relationship between the architecture and the measured performance. The report concludes with the limitations and conclusion.

2 Background

2.1 Transformer Architecture

The Transformer architecture was introduced by Vaswani et al. and is now widely used for language models [Vas+17]. A Transformer processes a sequence of tokens using attention and feed-forward layers. The main idea of attention is to allow each token to use information from other tokens in the sequence. This helps the model learn relationships between different parts of the input.

A typical Transformer block contains a self-attention layer followed by a feed-forward network. Residual connections and normalization are also used. One of the main advantages of the Transformer is that attention allows the model to directly connect information from different positions in a sequence. This makes it possible to model long-range relationships between tokens. However, the cost of attention becomes more important as the sequence length increases.

2.2 Self-Attention

Self-attention uses three main representations called queries, keys, and values. These representations are calculated from the hidden states of the input sequence. The attention operation can be written as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (1)$$

The queries are compared with the keys to calculate attention weights. These weights are then used to combine the values. As a result, each token can use information from other tokens in the sequence. This is one of the main reasons why Transformers work well for language modeling.

The main limitation is that the number of possible token interactions grows with the sequence length. Therefore, attention can require a large amount of computation and memory for long sequences. This problem becomes especially important during inference because the model generates tokens one at a time.

2.3 Language Model Inference

During autoregressive inference, a language model generates one token at a time. The generated token is added to the sequence and is then used to predict the next token. This

process continues until the required number of tokens has been generated.

The time required for this process depends on several factors. These include the model architecture, input length, output length, hardware, numerical precision, and software implementation. Therefore, the architecture of a model alone does not completely determine its practical inference speed.

2.4 Key-Value Cache

Transformer models normally use a Key-Value cache during autoregressive generation. The cache stores the keys and values from previous tokens so that they do not need to be calculated again at every generation step. This makes generation more efficient.

However, the size of the KV cache increases as more tokens are processed. For long sequences, the cache can therefore become an important part of the GPU memory requirement. Reducing the amount of attention used by a model can reduce the amount of KV-cache memory that is required.

This is one reason why alternative sequence-processing methods such as Mamba are important when studying long-context inference.

2.5 State Space Models

State Space Models provide another way to process sequential data. Instead of directly comparing every token with other tokens using attention, an SSM maintains an internal state. This state is updated whenever a new input is processed.

A simple state-space formulation can be written as

$$x_t = Ax_{t-1} + Bu_t, \tag{2}$$

$$y_t = Cx_t + Du_t. \tag{3}$$

Here, u_t represents the input at time t , x_t represents the internal state, and y_t represents the output. The state contains information from previous inputs and is updated as new inputs arrive.

The main difference from attention is that an SSM does not need to explicitly compare the current token with every previous token. Instead, previous information is represented through the internal state. This makes SSMs an interesting alternative for sequence modeling.

2.6 Mamba

Mamba is a State Space Model designed for efficient sequence modeling [GD24]. One of its main ideas is the use of selective state spaces. Instead of using fixed state-space parameters for every input, Mamba allows important parameters to depend on the current input.

This allows the model to decide which information should be kept and which information can be ignored. This is useful for language because different tokens have different importance. For example, a name, number, or variable in a program may need to be remembered for a later part of the sequence, while other information may not be important.

Mamba therefore provides a way to process sequences while avoiding the need to use standard attention in every layer. The original Mamba work also focuses on efficient hardware implementation and reports good scaling with sequence length [GD24].

2.7 Mamba-2 and State Space Duality

Mamba-2 extends the ideas introduced by Mamba. A central part of Mamba-2 is the State Space Duality framework, introduced by Dao and Gu [DG24]. SSD shows a connection between state-space models and certain attention-like computations. This provides a mathematical connection between two different approaches to sequence modeling.

This connection is important because it allows state-space computation to be expressed in a way that can make better use of modern hardware. Mamba-2 also improves the implementation of the main state-space layer. The authors report that the Mamba-2 layer can be significantly faster than the original Mamba layer in their experiments [DG24].

Mamba-2 is particularly important for this report because it is one of the main sequence-processing components of Nemotron 3 Nano. Instead of using attention throughout the complete model, Nemotron uses Mamba-2 for most of its sequence processing and keeps a smaller number of attention layers.

3 Mixture-of-Experts

3.1 Introduction to Mixture-of-Experts

Mixture-of-Experts (MoE) is another important part of modern language-model architectures. In a normal feed-forward layer, the same parameters are used for every token. An MoE layer instead contains several expert networks and a router that decides which experts should process each token.

The main idea is that a token does not need to use every available expert. Only a small number of experts can be selected for each token. This is known as sparse computation. It allows the complete model to contain many parameters while keeping the amount of computation for each token smaller than that of a dense model with the same total number of parameters.

3.2 Sparse Expert Routing

The router receives the representation of a token and calculates a score for each available expert. The experts with the highest scores are then selected for that token. A simplified form of the output can be written as

$$y = \sum_{i \in S(x)} g_i(x) E_i(x), \quad (4)$$

where $S(x)$ represents the selected experts, E_i represents an expert, and $g_i(x)$ represents the routing weight for that expert.

The important point is that the complete set of experts does not have to process every token. This reduces the active computation while allowing the model to have a large number of parameters.

3.3 Total and Active Parameters

Nemotron 3 Nano has approximately 31.6 billion total parameters, while approximately 3.6 billion parameters are active for each token [NVI25].

The total parameter count represents the complete model, including the different experts. The active parameter count represents the part of the model that participates in the computation for one token.

However, inactive experts still need to be stored. Therefore, an MoE model with 3.2 billion active parameters does not have the same memory requirements as a dense 3.2 billion parameter model. This distinction is important when interpreting GPU memory measurements.

3.4 Advantages and Limitations

The main advantage of MoE is that it allows a model to have a large capacity without using every parameter for every token. This can increase the number of parameters available to the model while keeping the active computation relatively small.

However, MoE also introduces additional work. The router has to select experts and the system has to send tokens to the selected experts. The experts may also be stored in different parts of GPU memory. Therefore, the final inference speed depends not only on the number of active parameters but also on the routing mechanism, memory movement, GPU kernels, and software implementation.

4 Nemotron 3 Nano Architecture

4.1 Architecture Overview

Nemotron 3 Nano uses a hybrid architecture that combines Mamba-2, Transformer attention, and Mixture-of-Experts layers [NVI25]. The model contains 52 layers, including Mamba-2 layers, MoE layers, and a smaller number of attention layers. This means that the same type of computation is not used throughout the complete network.

The main sequence-processing work is handled by Mamba-2, while MoE layers provide sparse computation. A smaller number of attention layers are kept to provide direct interactions between tokens. This combination is the main feature that makes Nemotron different from a standard Transformer-based language model.

4.2 Hybrid Mamba-2 and Attention

The combination of Mamba-2 and attention is one of the most important parts of the Nemotron architecture. Mamba-2 processes sequence information using an internal state, while attention allows the model to directly compare information from different positions in the sequence.

These two methods have different properties. Mamba-2 can process sequence information without requiring attention at every layer, while attention provides a direct way to retrieve information from other positions. Nemotron combines both approaches instead of depending completely on one of them.

This design allows the model to use Mamba-2 for most of the sequence processing while keeping attention in selected layers. The idea is to reduce the amount of computation and

memory associated with using attention throughout the whole model without removing attention completely.

4.3 Mamba-2 Layers

Mamba-2 is used in a large part of the Nemotron layer stack. During generation, these layers maintain their internal state rather than using a normal attention KV cache for every layer.

This can be useful for long sequences because the sequence information in these layers is represented through the state. The model therefore does not need to maintain an attention cache for all of its layers.

However, this does not mean that Nemotron has no memory cost for long sequences. The model still contains its weights, MoE experts, attention layers, runtime buffers, and other memory requirements. The main point is that the Mamba-2 layers do not have the same KV-cache behavior as normal Transformer attention layers.

4.4 Attention and Grouped-Query Attention

Nemotron does not completely remove attention. A small number of layers still use attention because direct interactions between tokens can be useful for language modeling.

The attention layers use Grouped-Query Attention (GQA). GQA uses fewer key and value heads than query heads. This reduces the amount of information that needs to be stored in the KV cache.

Therefore, Nemotron reduces the number of attention layers and also uses GQA inside those layers. These two choices help reduce the amount of attention-related memory compared with a model that uses standard attention throughout the complete network.

4.5 MoE Layers

Nemotron also uses Mixture-of-Experts layers. The model contains a large number of experts, but only a subset of these experts is selected for each token.

This allows the model to have a large total parameter count while keeping the active computation smaller. The Mamba-2 and MoE components therefore address different parts of the inference problem. Mamba-2 changes how sequence information is processed, while MoE changes how many parameters are used for each token.

4.6 Long-Context Processing

Nemotron 3 Nano is designed for long-context workloads, and NVIDIA reports support for contexts of up to 1 million tokens [NVI25]. This is important when considering why the model uses a hybrid architecture.

In a standard Transformer, every attention layer contributes to the KV cache. In Nemotron, only a small part of the network uses attention, while most of the layers use Mamba-2. Therefore, the amount of attention-related cache can be reduced.

The benefit of this design becomes more interesting as the context becomes longer. However, the exact memory and speed benefits depend on the hardware and inference implementation used to run the model.

4.7 Why the Architecture Can Help Inference

The intended efficiency of Nemotron comes from several parts of its architecture. Mamba-2 provides an alternative to using attention in every layer. MoE reduces the number of expert parameters that are active for each token. GQA reduces the amount of KV-cache data in the attention layers.

These ideas work together, but they do not guarantee higher measured throughput in every environment. The actual performance also depends on the GPU, software, kernels, memory movement, and the type of workload being used.

5 Comparison with Qwen3-30B-A3B

5.1 Reason for the Comparison

The original version of this report compared Nemotron 3 Nano with Llama 3.2 3B. However, these models have very different sizes. Llama 3.2 3B has about 3 billion parameters, while Nemotron has about 30 billion total parameters. Because of this difference, it is difficult to use the old comparison to study the effect of architecture.

For the new benchmark, Qwen3-30B-A3B-Instruct-2507 is used instead. Qwen3 has a similar total parameter count and active parameter count to Nemotron. It also uses a Mixture-of-Experts architecture. This makes the comparison more useful for studying the difference between the sequence-processing approaches used by the two models.

5.2 Architectural Differences

Nemotron and Qwen3 both use MoE, but their sequence-processing architectures are different. Nemotron uses a combination of Mamba-2 and attention, while Qwen3 mainly uses a Transformer-based architecture. The two models also have different layer configurations and expert settings.

Therefore, the benchmark cannot isolate Mamba-2 as the only difference between the models. They also differ in training, layer structure, dimensions, expert configuration, and implementation. However, their similar parameter counts make the comparison more balanced than the previous comparison with Llama 3.2 3B.

Table 1: Main architectural characteristics of the two models.

Property	Nemotron 3 Nano	Qwen3-30B-A3B
Total parameters	about 31.6B	about 30.5B
Active parameters	about 3.6B	about 3.3B
MoE	Yes	Yes
Mamba-2	Yes	No
Attention	Hybrid	Transformer-based

The similar parameter counts make this comparison more useful for studying architectural differences. At the same time, the comparison should still be treated as a comparison between two complete released models rather than as a controlled experiment on one individual architectural component.

6 Experimental Methodology

6.1 Hardware

The benchmark was performed on an NVIDIA A100-SXM4 GPU with 80 GB of GPU memory. The model files were stored locally because the benchmark was performed in an offline HPC environment.

6.2 Software

The benchmark used Python, PyTorch, CUDA, and the Hugging Face Transformers library. Both models were evaluated using BF16 precision and a batch size of one.

6.3 Models

The following two models were tested:

- NVIDIA Nemotron 3 Nano 30B-A3B BF16
- Qwen3-30B-A3B-Instruct-2507

Both models were loaded from local checkpoints. Using local checkpoints avoided network-related differences during the benchmark and ensured that both models were evaluated using the same local execution environment.

6.4 Benchmark Prompts

Five short prompts were used to evaluate the inference performance of both models. The prompts had input lengths between approximately 11 and 17 tokens. For each model, all five prompts were executed once using BF16 precision, batch size one, and a maximum output length of 128 tokens.

The benchmark was designed to measure inference performance rather than language quality. Therefore, the prompts were kept simple and the same general generation configuration was used for both models.

6.5 Evaluation Metrics

Four main measurements were recorded during the benchmark.

Generation throughput measures the number of generated tokens per second. It is calculated as

$$\text{Throughput} = \frac{\text{Generated Tokens}}{\text{Generation Time}}.$$

Generation latency measures the total time required to generate the output. Peak GPU memory measures the highest amount of GPU memory used during inference. Model loading time measures the time required to load the model from its local checkpoint before generation starts.

Loading time is reported separately because it does not directly measure token generation speed.

6.6 Benchmark Procedure

Each model was first loaded from its local checkpoint. After loading, the model was evaluated on the same five prompts under identical generation settings. Each prompt was executed once, with BF16 precision, batch size one, and a maximum output length of 128 tokens.

For every prompt, the generation time, number of generated tokens, and peak GPU memory usage were recorded. The throughput was calculated from the generated token count and generation time. The reported benchmark values for each model were then calculated as arithmetic means across the five prompts.

Model loading time was measured separately from generation and was reported as the time required to load the local model checkpoint.

The experiment is intentionally small. Its purpose is to connect the architectural discussion with practical inference behavior rather than to provide a comprehensive benchmark across different hardware, workloads, or inference configurations.

7 Experimental Results

7.1 Overall Results

The average results from the five prompts are shown in Table 2.

Table 2: Average results from the five-prompt benchmark.

Model	Throughput tokens/s	Latency seconds	Peak Memory GB	Load Time seconds
Qwen3-30B-A3B	6.50	19.69	56.89	470.31
Nemotron 3 Nano	1.54	83.15	59.04	666.57

In this benchmark, Qwen3 generated tokens faster than Nemotron. Qwen3 also had lower generation latency. The peak memory values were closer, with Qwen3 using about 56.89 GB and Nemotron using about 59.04 GB.

These results are specific to the benchmark setup and should not be treated as a general comparison of the two models. The experiment used one GPU, a small number of prompts, short inputs, and a short generation length. The difference in generation performance can be interpreted in the context of the complete inference pipeline. Although Nemotron uses Mamba-2 and sparse Mixture-of-Experts layers to reduce reliance on conventional attention, these architectural choices do not automatically result in higher throughput on every hardware and workload configuration. The measured performance also depends on kernel implementations, memory movement, expert routing, model initialization, and the software stack. Therefore, the benchmark demonstrates the behavior of the two complete model implementations on the A100 GPU rather than isolating the contribution of any single architectural component.

7.2 Generation Throughput

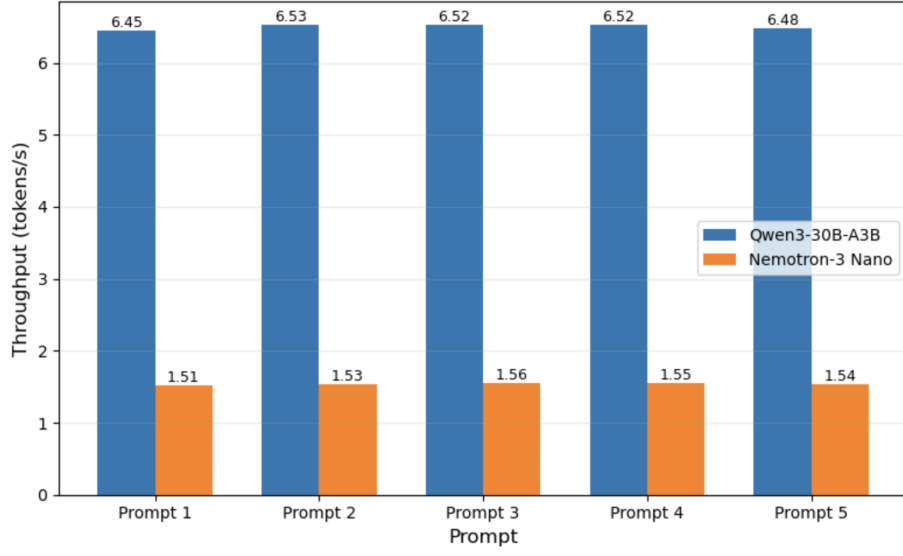


Figure 1: Generation throughput for the five benchmark prompts.

Qwen3 achieved between approximately 6.45 and 6.53 tokens per second. Nemotron achieved between approximately 1.51 and 1.56 tokens per second. The values were relatively similar between the different prompts for each model.

The result shows that Qwen3 had higher measured throughput in this particular environment. However, the result does not show that the Transformer architecture is always faster than Mamba-2. It only shows the performance of these two complete model implementations under the selected benchmark conditions.

7.3 Generation Latency

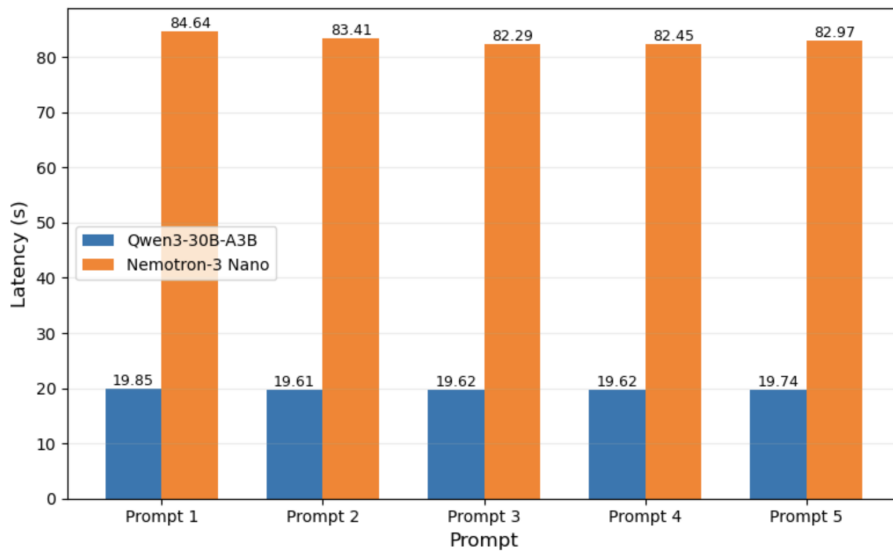


Figure 2: Generation latency for the five benchmark prompts.

Qwen3 required approximately 19.6 to 19.8 seconds to generate 128 tokens. Nemotron required approximately 82.3 to 84.6 seconds.

The latency results are consistent with the throughput results. Since both models generated the same maximum number of tokens, the model with lower throughput also required more time to finish the generation.

7.4 GPU Memory

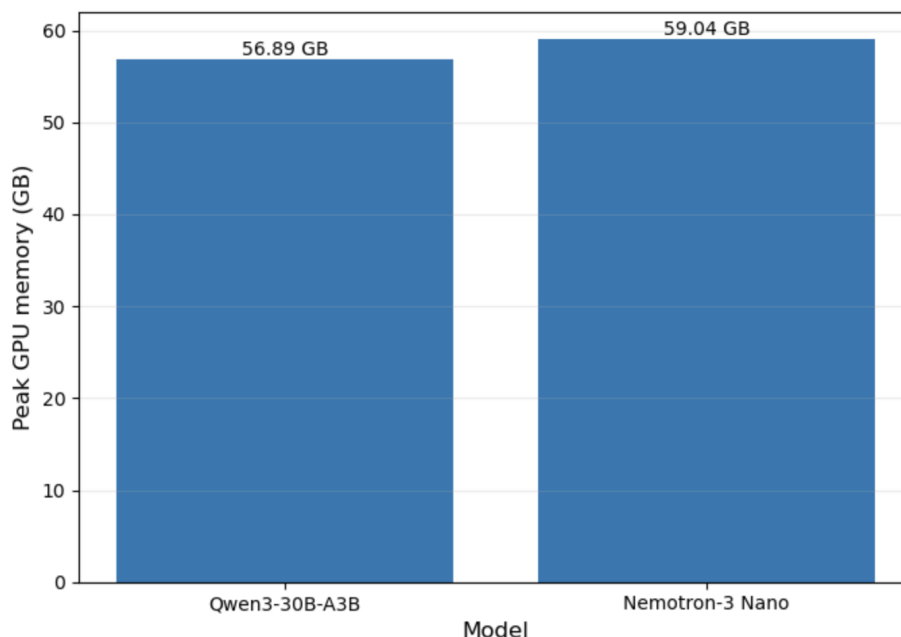


Figure 3: Peak GPU memory used during inference.

Qwen3 used approximately 56.89 GB of peak GPU memory, while Nemotron used approximately 59.04 GB. The difference was about 2.15 GB.

The memory values are relatively close considering that the models have around 30 billion total parameters. This also shows why total and active parameter counts should not be treated as the same thing. MoE reduces the number of parameters that are active for each token, but the complete model still needs to be stored.

7.5 Model Loading Time

Qwen3 required approximately 470.31 seconds to load, while Nemotron required approximately 666.57 seconds.

Loading time can depend on the checkpoint files, storage system, model initialization, and software. It can therefore vary between systems and should not be treated as the main measure of inference efficiency.

For this reason, the main focus of the benchmark is generation throughput, generation latency, and GPU memory.

7.6 Long-Context Evaluation

To investigate the behavior of Nemotron 3 Nano with increasing input context, an additional long-context experiment was performed at four context lengths: approximately 2K, 4K, 8K, and 16K tokens. The experiment used the same Nemotron 3 Nano BF16 checkpoint and the same NVIDIA A100-SXM4 80GB GPU used for the main benchmark. A controlled document-retrieval workload was constructed for each context length. The input contained a large synthetic technical document with a specific information item embedded at a controlled position. The model was then asked to retrieve the embedded information.

The experiment measured the actual number of input tokens, generation latency, generation throughput, and peak GPU memory. The results are shown in Table 3.

Table 3: Nemotron 3 Nano long-context inference results.

Target Context (tokens)	Actual Input (tokens)	Latency (s)	Throughput (tokens/s)	Peak Memory (GB)
2,048	1,935	28.51	1.12	60.32
4,096	3,983	23.49	1.36	61.83
8,192	8,079	29.84	1.07	64.85
16,384	16,271	44.38	0.72	70.89

The measurements show that GPU memory increased consistently as the context length increased. Peak memory increased from 60.32 GB at approximately 2K input tokens to 70.89 GB at approximately 16K input tokens. This corresponds to an increase of approximately 10.57 GB across the tested context range.

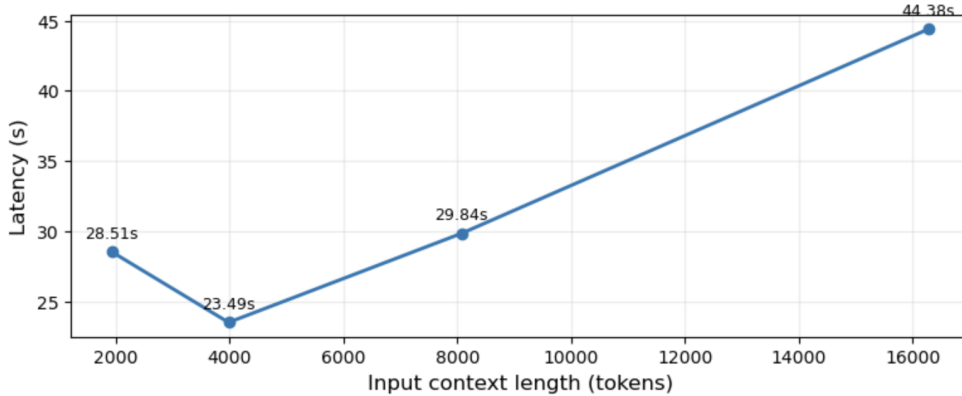


Figure 4: Generation latency of Nemotron 3 Nano across increasing input context lengths.

Generation latency was 28.51 s at approximately 2K tokens, 23.49 s at 4K tokens, 29.84 s at 8K tokens, and 44.38 s at 16K tokens. The 16K configuration therefore required substantially more time than the shorter configurations. The latency measurements do not follow a strictly monotonic trend across all four context lengths.

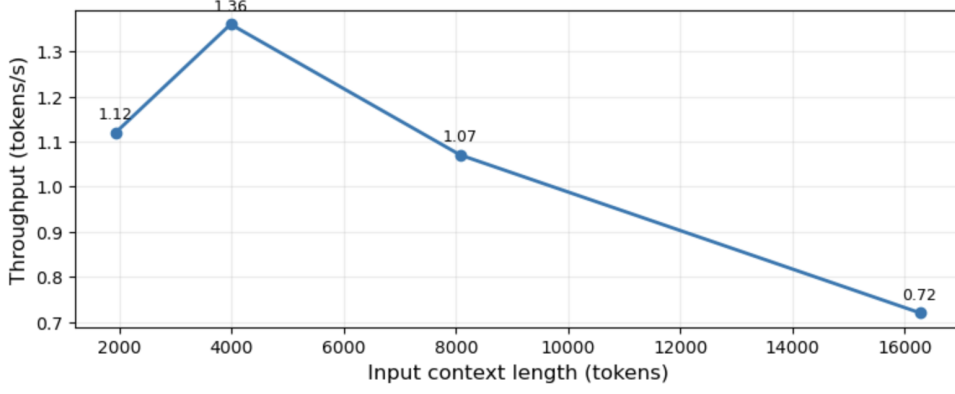


Figure 5: Generation throughput of Nemotron 3 Nano across increasing input context lengths.

Generation throughput was 1.12 tokens/s at approximately 2K tokens and increased to 1.36 tokens/s at approximately 4K tokens. It then decreased to 1.07 tokens/s at approximately 8K tokens and 0.72 tokens/s at approximately 16K tokens. Thus, the lowest measured throughput occurred at the longest tested context length.

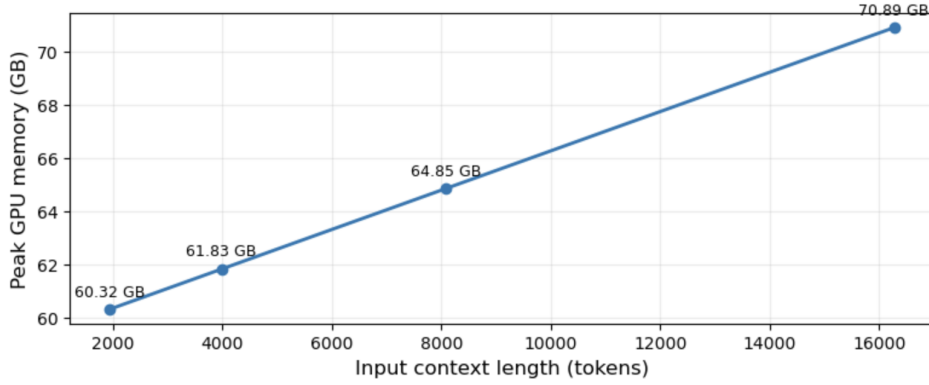


Figure 6: Peak GPU memory usage of Nemotron 3 Nano as a function of input context length.

The memory measurements provide the clearest trend across the tested context lengths. Peak GPU memory increased from 60.32 GB at 1,935 input tokens to 70.89 GB at 16,271 input tokens. This demonstrates that longer contexts increase the runtime memory requirement even though Nemotron uses a hybrid Mamba–Transformer architecture.

The increase in memory should not be attributed to a single architectural component. The measured value reflects the complete Nemotron inference system, including its Mamba-2 layers, attention layers, MoE computation, model weights, and runtime buffers.

8 Architecture-Performance Discussion

The benchmark showed lower generation throughput for Nemotron 3 Nano than for Qwen3-30B-A3B in the tested setup. This result should be considered together with

the architecture and the workload used in the experiment. Nemotron combines Mamba-2, attention, and MoE layers, but the benefit of these components depends on the type of workload and on how efficiently the operations are implemented.

8.1 Effect of the Workload

The short-prompt benchmark provides a measurement of normal autoregressive generation, but it does not strongly stress the long-context properties that motivate the hybrid Mamba-Transformer architecture. Therefore, an additional experiment was performed using approximately 2K, 4K, 8K, and 16K input contexts.

The long-context measurements show a clear increase in peak GPU memory as the input becomes longer. Memory increased from 60.32 GB at approximately 2K tokens to 70.89 GB at approximately 16K tokens. Generation throughput also decreased to 0.72 tokens/s at the 16K configuration.

These results indicate that the practical cost of long-context inference depends on more than the number of active parameters. The model still requires memory for its weights, Mamba states, attention-related cache, MoE computation, and runtime buffers. Therefore, the hybrid architecture reduces the reliance on conventional attention but does not eliminate memory growth for long-context inference.

8.2 Architecture and Implementation

The theoretical design of a model does not directly determine its practical inference speed. Nemotron combines Mamba-2 operations, attention, MoE routing, expert computation, and other model operations. The efficiency of the complete system depends on how these operations are implemented and optimized for the target hardware.

MoE can reduce the amount of expert computation because only a subset of experts is activated for each token. At the same time, routing tokens to different experts introduces additional operations. Similarly, Mamba-2 avoids using attention in every layer, but its practical benefit depends on the implementation and workload.

This means that the active parameter count or the use of an efficient architectural component alone cannot predict the final tokens-per-second result.

8.3 Comparison with Published Results

NVIDIA reports higher throughput for Nemotron 3 Nano than Qwen3-30B-A3B under its published benchmark conditions [NVI25]. However, those measurements were performed using different hardware and a much longer workload than the experiment in this report.

The benchmark in this work used an NVIDIA A100 80GB with short input prompts and a maximum output length of 128 tokens. The different hardware, workload, and software environment make a direct numerical comparison with the NVIDIA results inappropriate.

The difference between the two experiments shows that inference performance depends not only on model architecture but also on the hardware, workload, and implementation.

8.4 Role of Mamba-2, MoE, and Attention

Mamba-2 is used in Nemotron to process sequence information without relying on attention in every layer. This can reduce the amount of attention computation and the

associated KV-cache requirements. Its advantages are expected to become more relevant for longer sequences.

MoE provides another form of efficiency. Nemotron contains a large number of total parameters, while only a subset of experts is activated for each token. This allows the model to maintain high parameter capacity without computing all expert parameters for every token. However, expert routing and dispatch also introduce additional operations.

Attention is still used in selected layers. This provides direct interactions between tokens while avoiding the need to use attention throughout the entire model. The combination of these components is therefore an important part of Nemotron’s design.

8.5 Discussion

The main observation from the experiment is that architectural efficiency and measured inference performance are not necessarily the same. Nemotron contains several design choices intended to improve efficiency, but it did not achieve higher generation throughput than Qwen3 in the tested A100 workload.

This result does not provide evidence that Mamba-2 is less efficient than attention. Instead, it shows that the performance of a complete hybrid model depends on the interaction between its architecture, implementation, hardware, and workload.

The long-context experiment provides additional evidence that workload length has an important effect on inference behavior. Increasing the context from approximately 2K to 16K tokens increased peak GPU memory by approximately 10.57 GB and reduced the measured throughput at the longest tested context. These observations are consistent with the motivation for reducing the amount of conventional attention in hybrid architectures.

However, the experiment does not isolate the contribution of Mamba-2 from the other components of Nemotron. The measured values represent the complete model and inference implementation. A controlled ablation study using models with identical parameters and training conditions would be required to attribute a specific performance improvement directly to Mamba-2.

9 Limitations and Future Work

The benchmark was conducted on a single NVIDIA A100-SXM4 80GB GPU. The short-context comparison used five prompts, while the additional long-context experiment evaluated four context lengths from approximately 2K to 16K tokens. Therefore, the results should not be interpreted as a general performance comparison across all hardware and workloads.

The long-context workload used a controlled synthetic document-retrieval setup rather than a complete real-world Retrieval-Augmented Generation (RAG) pipeline. The experiment therefore measures the effect of increasing input context on inference performance, but it does not represent the full cost of document retrieval, embedding generation, chunking, or external vector databases.

Future work could evaluate larger contexts, multiple batch sizes, and additional GPU architectures. A comparison using the same optimized inference engine and precision for both models would also provide a more controlled measurement of architectural differences. Further experiments could additionally isolate the effects of Mamba-2, attention, and MoE through controlled ablation studies.

10 Conclusion

This report studied the architecture of NVIDIA Nemotron 3 Nano, with a focus on Mamba-2, Transformer attention, and Mixture-of-Experts. These components are combined in a hybrid architecture that aims to maintain large model capacity while reducing the dependence on conventional attention throughout the complete network.

The short-context benchmark with Qwen3-30B-A3B showed that inference performance depends on both architecture and implementation. In the tested setup, Qwen3 achieved higher generation throughput, while the two models had relatively similar peak GPU memory usage.

The additional long-context experiment provided measurements at approximately 2K, 4K, 8K, and 16K input tokens. Peak GPU memory increased from 60.32 GB at approximately 2K tokens to 70.89 GB at approximately 16K tokens, while throughput decreased to 0.72 tokens/s at the longest tested context. These results demonstrate that context length has a substantial effect on the practical inference cost of the complete Nemotron system.

Overall, the experiments illustrate that architectural efficiency cannot be separated from workload, hardware, and implementation. Nemotron 3 Nano provides an example of how Mamba-2, selective attention, and MoE can be combined in a single language model, while the measured results show the practical memory and latency trade-offs that arise during inference.

References

- [DG24] Tri Dao and Albert Gu. “Transformers are SSMS: Generalized Models and Efficient Algorithms Through Structured State Space Duality”. In: *International Conference on Machine Learning*. 2024. arXiv: 2405.21060.
- [GD24] Albert Gu and Tri Dao. “Mamba: Linear-Time Sequence Modeling with Selective State Spaces”. In: *International Conference on Learning Representations*. 2024. arXiv: 2312.00752.
- [NVI25] NVIDIA. *Nemotron 3 Nano: Open, Efficient Mixture-of-Experts Hybrid Mamba-Transformer Model for Agentic Reasoning*. Tech. rep. NVIDIA, 2025.
- [Vas+17] Ashish Vaswani et al. “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems*. 2017.

A Transparency Report

This section documents the sources, tools, and assistance used during the preparation of the report.

A.1 Literature and External Sources

The report uses the original Transformer paper, the Mamba paper, the Mamba-2 paper, the NVIDIA Nemotron 3 Nano technical documentation, and the Qwen3 model documentation. These sources are cited in the main text where their information is used.

A.2 Software and Tools

The practical experiment was performed using Python, PyTorch, CUDA, and the Hugging Face Transformers library. The model checkpoints were available locally in the HPC environment.

A.3 Benchmark Data

The benchmark data was collected from five prompts for each model. The recorded values include input length, output length, generation time, generation throughput, peak GPU memory, and model loading time.

The reported results are based on the author's benchmark runs on the NVIDIA A100-SXM4 80GB GPU.

A.4 AI Assistance

Generative AI was used during the preparation of this report for language editing, improving the structure of the report, explaining technical concepts, and helping to organize the report according to the supervisor's feedback.

The benchmark itself was performed by the author. The reported measurements were obtained from the author's experimental environment. The author reviewed the generated material and is responsible for the final content, interpretation, and references.

A.5 Use of Supervisor Feedback

The report was revised based on the feedback provided by the supervisor. The main changes were to move the focus from the practical benchmark to the theoretical discussion, include Mamba-2 and State Space Duality, use a more similarly sized comparison model, improve citations, and treat the benchmark as a supporting case study.

A.6 Limitations of AI Assistance

AI assistance was not used as a source for experimental measurements. The benchmark results, model execution, and observed system behavior come from the author's own experiment.

All important technical claims should be checked against the original papers and official model documentation before submission.