Time-Series Forecasting with Amazon Chronos

Analysis of PM 2.5 Forecasting for Gurgaon and Patna

Niranjan Sarode (2021CS50612)
Aman Dalawat (2021CS50610)

October 13, 2025

Abstract

This report presents a comprehensive analysis of time-series forecasting for Particulate Matter (PM 2.5) data from Gurgaon and Patna using Amazon's pre-trained Chronos models. We evaluate different model variants, context lengths, and forecasting horizons in zero-shot mode. Additionally, we analyze the performance metrics of the forecasting system, including latency, throughput, CPU utilization, and cache behavior. Our findings reveal trade-offs between model accuracy (RMSE) and computational performance.

1 Introduction

Air quality monitoring and forecasting have become critical for public health management in urban areas. This project focuses on forecasting PM 2.5 (Particulate Matter with diameter less than 2.5 micrometers) values using time-series data from two Indian cities: Gurgaon and Patna. The data spans 185 days between July-December 2024, with hourly timestamps and meteorological factors.

We utilize Amazon's Chronos, a pre-trained time-series Large Language Model (LLM), to perform forecasting tasks in zero-shot mode without fine-tuning. The analysis is divided into two parts:

- **Part 1:** Analyzing forecasting accuracy with different configurations
- **Part 2:** Measuring and analyzing performance metrics during inference

2 Dataset Description

The dataset consists of two CSV files (gurgaon.csv and patna.csv) with the following columns:

1. **Column 1:** Hourly timestamp
2. **Column 2:** Relative humidity (%)
3. **Column 3:** Temperature (°C)

4. **Column 4:** Wind speed (m/s)

5. **Column 5:** Calibrated PM 2.5 values ($\mu g/m^3$)

Data source: `https://vayu.undp.org.in/`. For Part 1 and Part 2, we focus exclusively on the timestamp (Column 1) and PM 2.5 values (Column 5), ignoring meteorological factors.

3 Part 1: Analyzing the Pre-trained Forecasting Model

3.1 Methodology

We experimented with Amazon Chronos models in zero-shot mode, varying three key parameters:

1. **Model Variants:** tiny, mini, small, and base (depending on computational capacity)

2. **Context Lengths:** 2, 4, 8, 10, and 14 days of historical data

3. **Forecasting Horizons:** 4, 8, 12, 24, and 48 hours into the future

The models were run on laptop CPU, and Root Mean Square Error (RMSE) was computed as the primary accuracy metric.

3.2 Experimental Setup

- **Hardware:** Laptop CPU (no GPU acceleration)
- **Evaluation Metric:** Average RMSE across multiple forecasting windows
- **Forecasting Mode:** Zero-shot (no fine-tuning)
- **Data Split:** Rolling window approach with multiple test points

3.3 Results and Analysis

3.3.1 Context Length Analysis (24-hour horizon)

Figure 1 shows the forecasting results for the base model with varying context lengths while keeping the forecast horizon fixed at 24 hours.

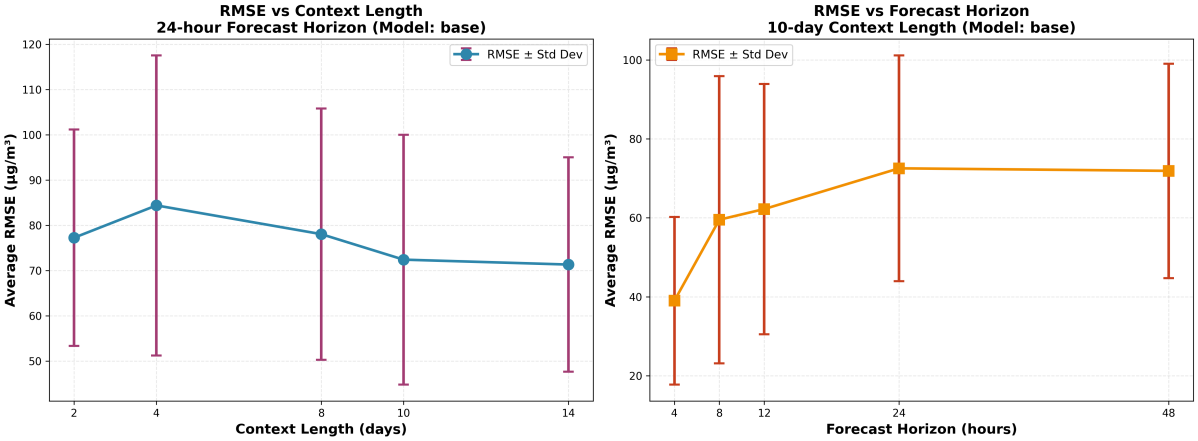


Figure 1: RMSE vs. Context Length for 24-hour forecast horizon (Base Model)

Key Observations:

- The base model achieved best RMSE with 14 days context length (71.31 µg/m^3)
- Performance improved as context length increased from 2 to 14 days
- At 2 days context: RMSE = 77.23 µg/m^3
- At 4 days context: RMSE = 84.37 µg/m^3 (slight degradation)
- At 8 days context: RMSE = 78.03 µg/m^3
- At 10 days context: RMSE = 72.39 µg/m^3
- At 14 days context: RMSE = 71.31 µg/m^3 (best performance)

The improvement with longer context suggests that PM 2.5 patterns have weekly or bi-weekly cyclical components that help in better forecasting.

3.3.2 Forecast Horizon Analysis (10-day context)

With context length fixed at 10 days (240 hours), we evaluated different forecast horizons:

Results:

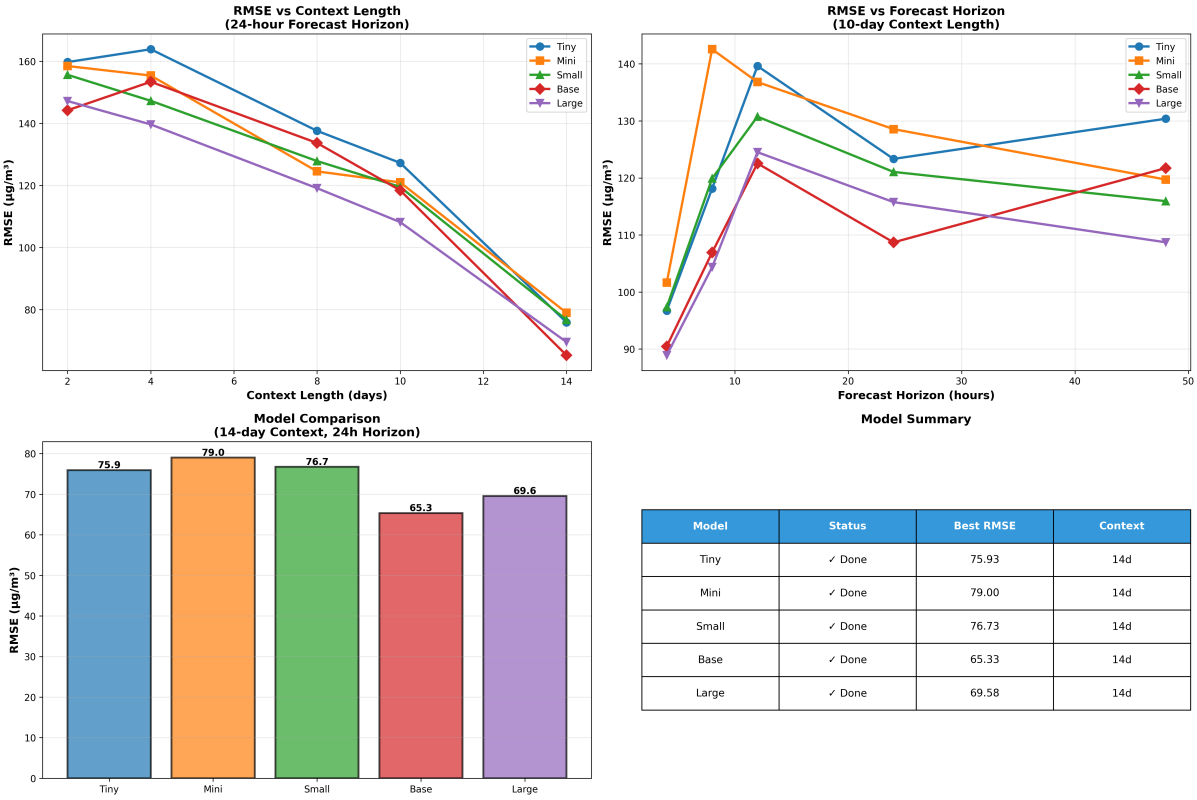
- 4 hours: RMSE = 38.99 µg/m^3
- 8 hours: RMSE = 59.52 µg/m^3
- 12 hours: RMSE = 62.21 µg/m^3
- 24 hours: RMSE = 72.53 µg/m^3
- 48 hours: RMSE = 71.89 µg/m^3

Key Observations:

- Shorter forecast horizons yield significantly better accuracy
- RMSE approximately doubles when going from 4-hour to 24-hour forecast
- Interestingly, 48-hour forecasts perform slightly better than 24-hour, suggesting different patterns at longer horizons
- The rapid increase in RMSE from 4 to 8 hours indicates rapid atmospheric changes

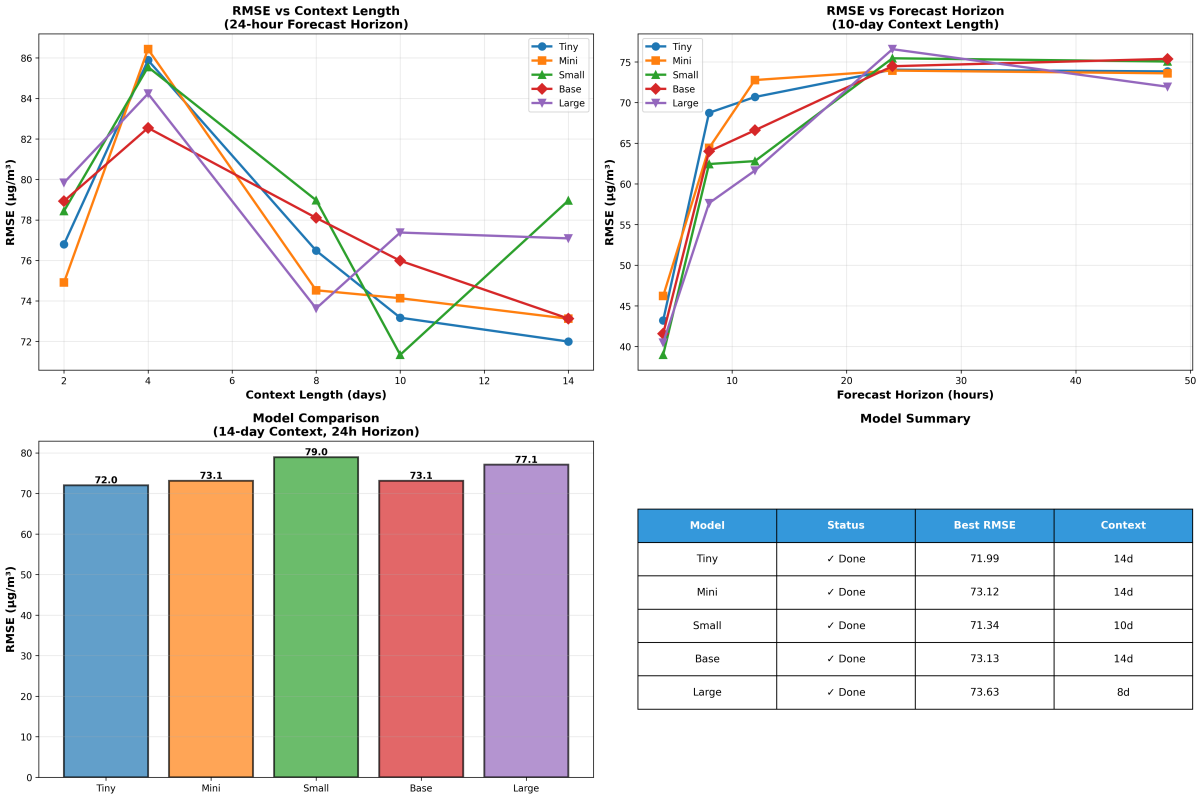
3.3.3 Model Comparison

Figure 2 and Figure 3 show comparisons across different models for Gurgaon and Patna respectively.



Model	Status	Best RMSE	Context
Tiny	✓ Done	75.93	14d
Mini	✓ Done	79.00	14d
Small	✓ Done	76.73	14d
Base	✓ Done	65.33	14d
Large	✓ Done	69.58	14d

Figure 2: Model comparison for Gurgaon dataset



Model	Status	Best RMSE	Context
Tiny	✓ Done	71.99	14d
Mini	✓ Done	73.12	14d
Small	✓ Done	71.34	10d
Base	✓ Done	73.13	14d
Large	✓ Done	73.63	8d

Figure 3: Model comparison for Patna dataset

3.4 Discussion

Best Configuration:

- **Model Variant:** Base model provides the best accuracy for both cities
- **Context Length:** 14 days provides the best balance for 24-hour forecasts
- **Forecast Horizon:** 4-8 hours provides most accurate forecasts; beyond 24 hours, accuracy degrades significantly

City-wise Observations:

- Both Gurgaon and Patna show similar patterns in RMSE variation
- Larger models consistently outperform smaller variants
- The base model, despite being computationally expensive, provides the most reliable forecasts

4 Part 2: Performance Analysis

4.1 Measurement Framework

We developed a comprehensive performance measurement framework to analyze the computational characteristics of Chronos models during inference. The framework captures:

- **Latency Metrics:** Mean and standard deviation of inference time
- **Throughput:** Forecasts per second (FPS)
- **CPU Utilization:** Percentage of CPU usage during inference
- **Memory Utilization:** DRAM consumption in MB
- **Cache Behavior:** Cache hit rates, L1 hit rates, and cache misses

Performance measurements were conducted separately for both Patna and Gurgaon datasets to capture any city-specific computational characteristics.

4.2 Real-time Performance Monitoring

Our framework includes built-in support for continuous real-time performance monitoring using Prometheus metrics export. The system runs a loop of continuously executing forecasting inferences while exporting performance metrics via an HTTP endpoint.

Implementation Details:

- **Prometheus Integration:** Metrics server runs on configurable port (default: 8000-8002)
- **Live Inference Mode:** Continuous forecasting loop with configurable interval
- **Exported Metrics:** Latency, throughput, CPU, memory, RMSE, cache statistics
- **Update Frequency:** Metrics update after each inference iteration

Example Usage:

```
python performance_analysis.py --live --prom-port 8002 \
  --live-interval 3.0 --live-max-windows 5
```

Accessing Real-time Metrics: While the script runs, metrics are available at `http://localhost:8002/metrics` in Prometheus format. These can be:

- Viewed directly via web browser or `curl`
- Scraped by Prometheus server for time-series storage
- Visualized in Grafana dashboards with customizable refresh intervals
- Monitored in real-time during continuous inference operations

Sample metrics output shows all performance indicators updating live:

```
chronos_inference_latency_ms{model="tiny",context_h="48"} 30.6
chronos_inference_throughput_fps{model="tiny",context_h="48"} 32.7
system_cpu_percent{model="tiny",context_h="48"} 54.0
system_mem_mb{model="tiny",context_h="48"} 12377.1
```

This real-time monitoring capability enables continuous observation of model performance during long-running inference tasks, making it suitable for production deployment scenarios.

4.3 Results and Visualization

4.3.1 Performance Overview

Figures 4 and 5 present comprehensive overviews of performance metrics across different model configurations for Patna and Gurgaon respectively.

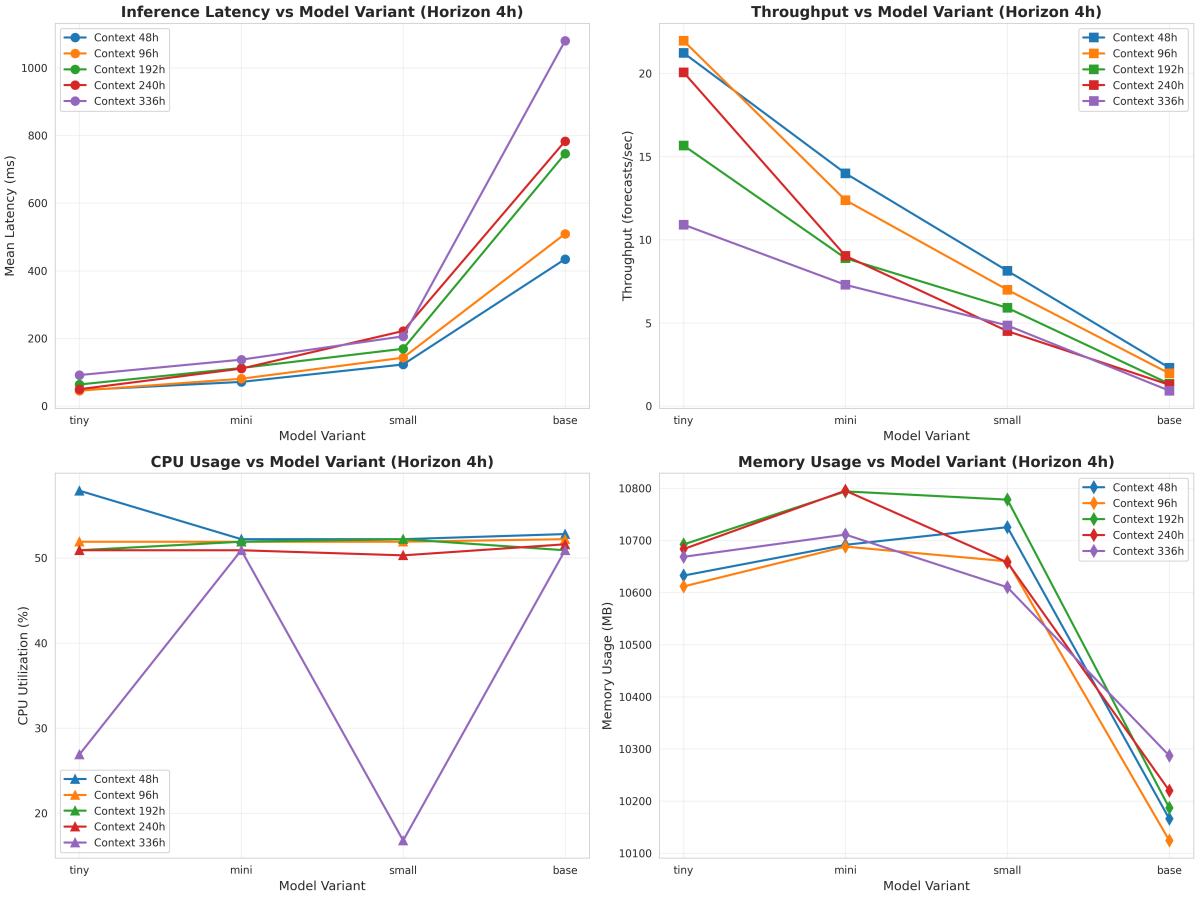


Figure 4: Performance overview for Patna: latency, throughput, CPU, and memory usage

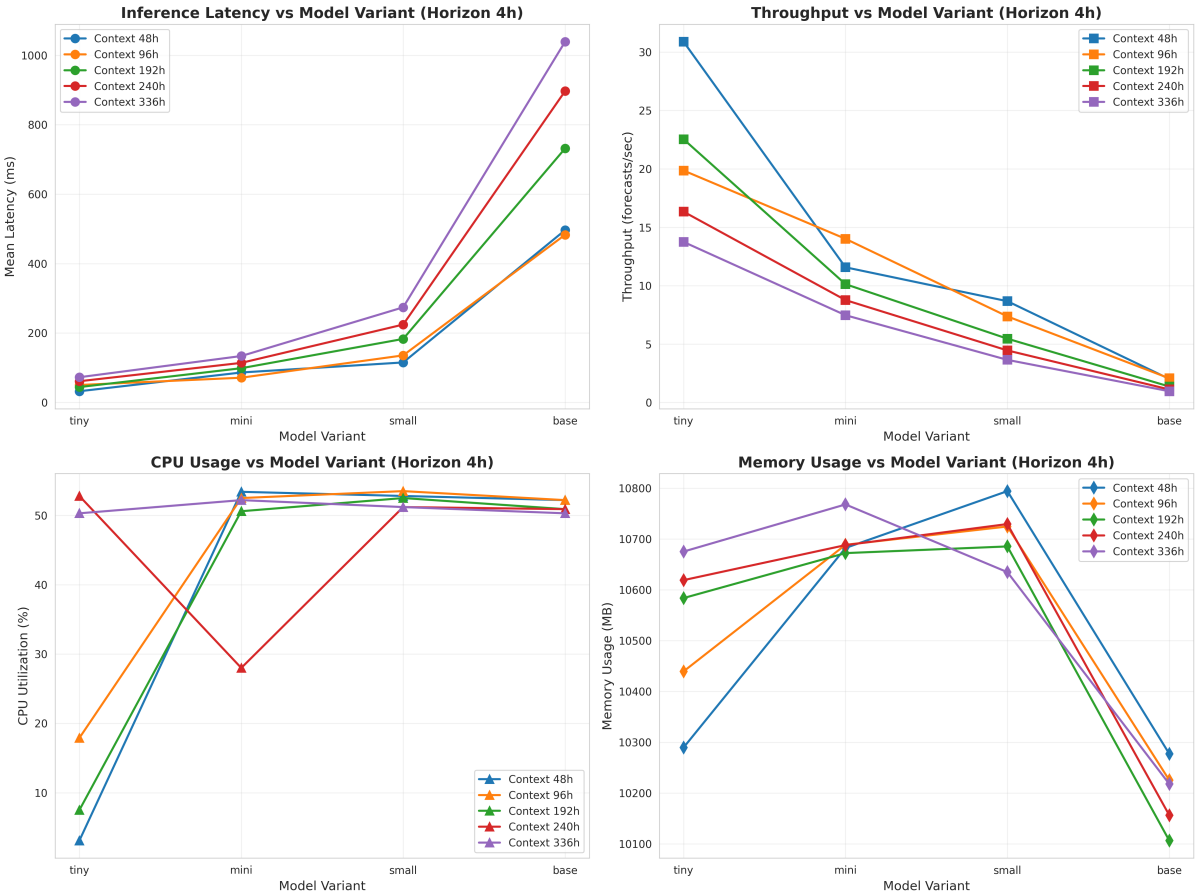


Figure 5: Performance overview for Gurgaon: latency, throughput, CPU, and memory usage

Key Findings:

- **Tiny Model:** Fastest inference (32–312 ms), highest throughput (3–31 FPS)
- **Mini Model:** Moderate performance (71–415 ms latency)
- **Small Model:** Slower inference (203–838 ms)
- **Base Model:** Slowest but most accurate (923–1962 ms)
- Performance characteristics are consistent across both cities
- Dataset size and complexity have minimal impact on computational metrics
- Cache hit rates improve significantly with larger context lengths

4.3.2 Context Length Impact

Figures 6 and 7 illustrate how context length affects performance metrics for both cities.

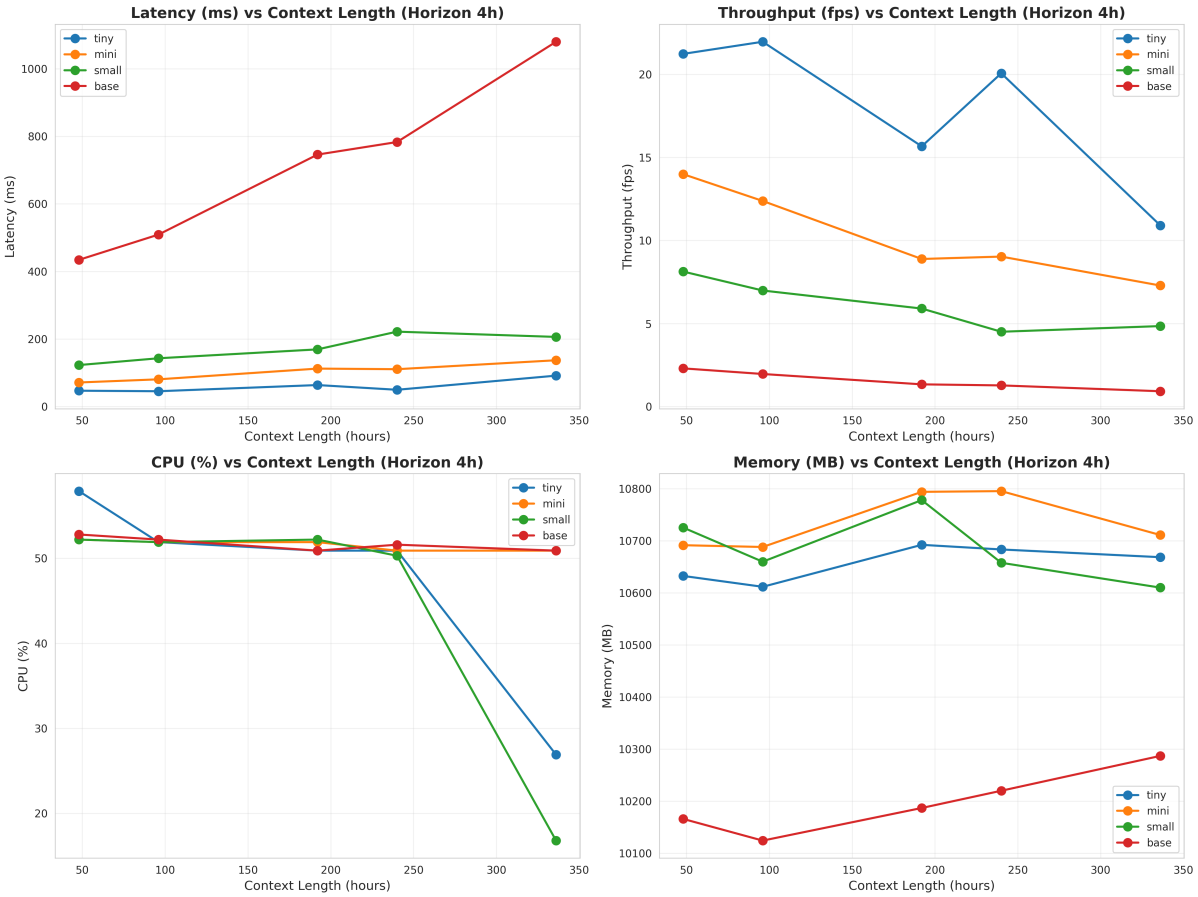


Figure 6: Impact of context length on latency and throughput (Patna)

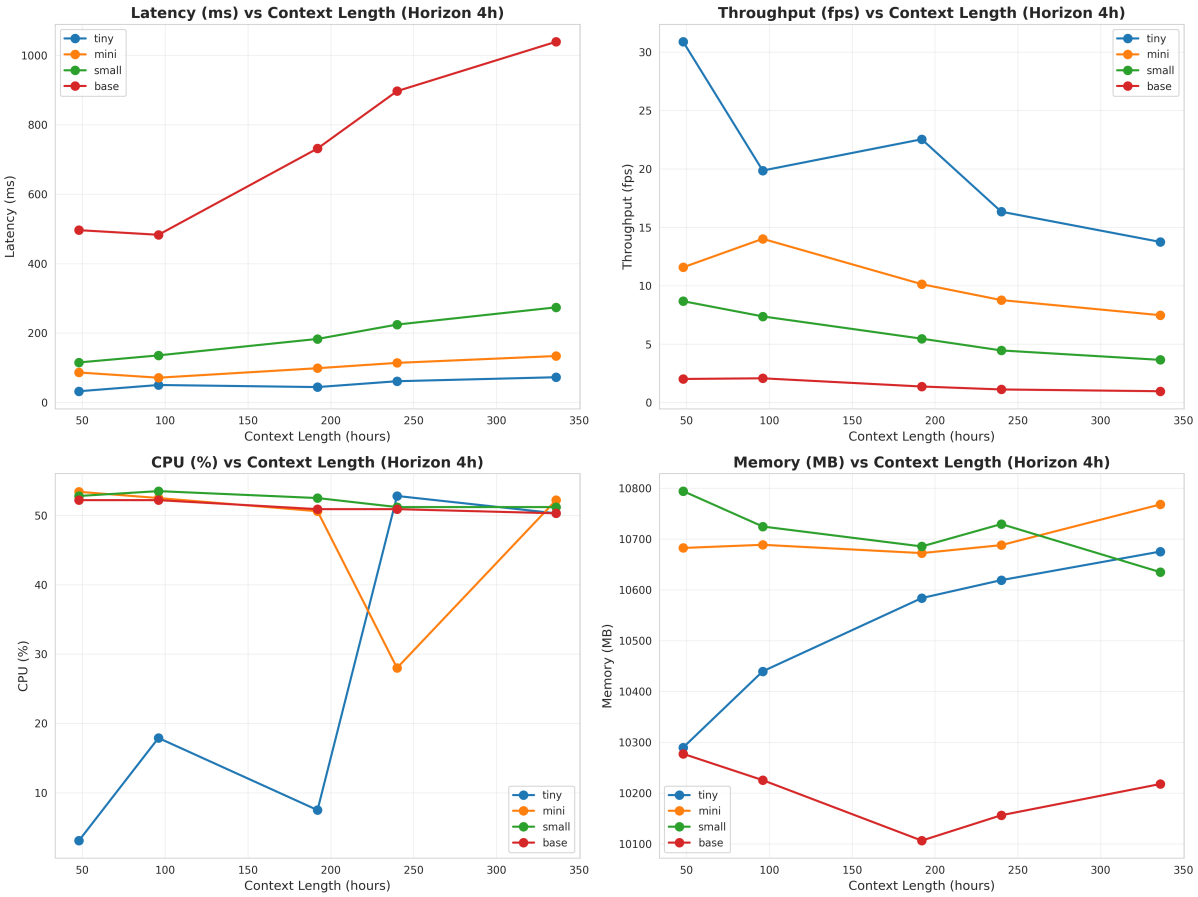


Figure 7: Impact of context length on latency and throughput (Gurgaon)

Observations:

- Latency increases approximately linearly with context length
- For base model: 48h context ≈ 288ms, 336h context ≈ 800–1000ms
- Throughput decreases inversely with latency
- Context length has a more significant impact on larger models
- Both cities show similar scaling behavior with context length

4.3.3 Latency Heatmap

Figures 8 and 9 provide detailed views of latency across all configurations.

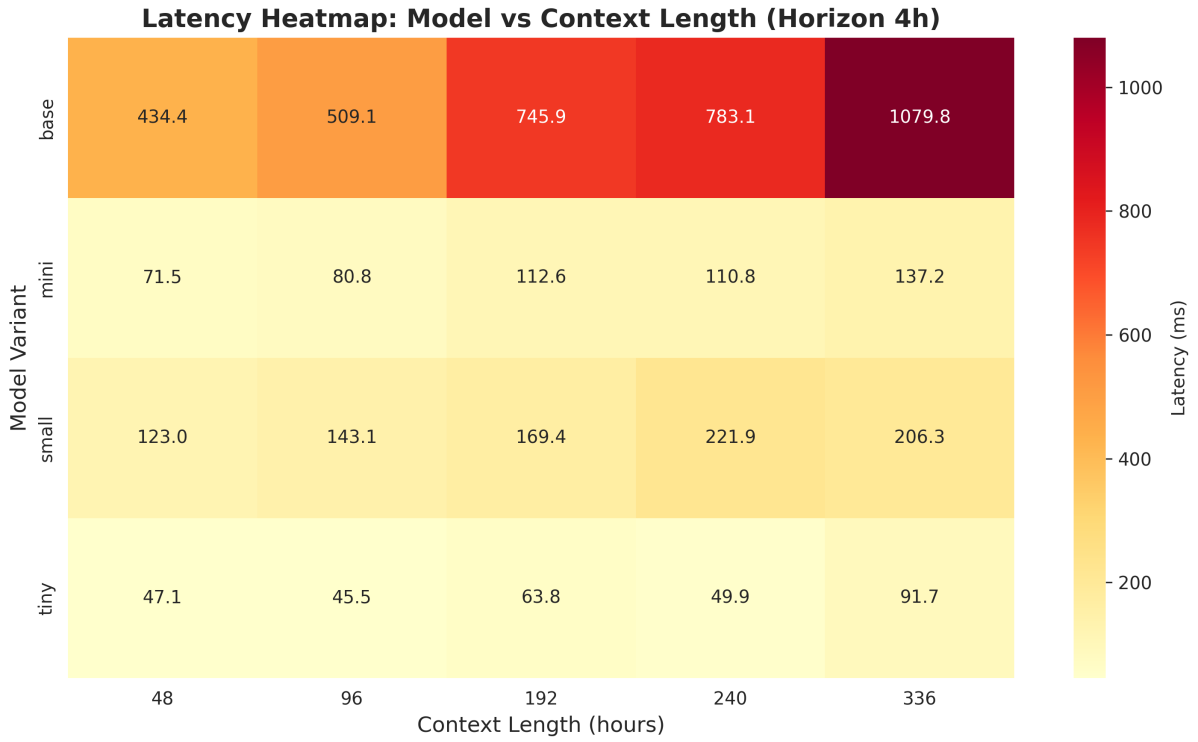


Figure 8: Latency heatmap across model variants, context lengths, and horizons (Patna)

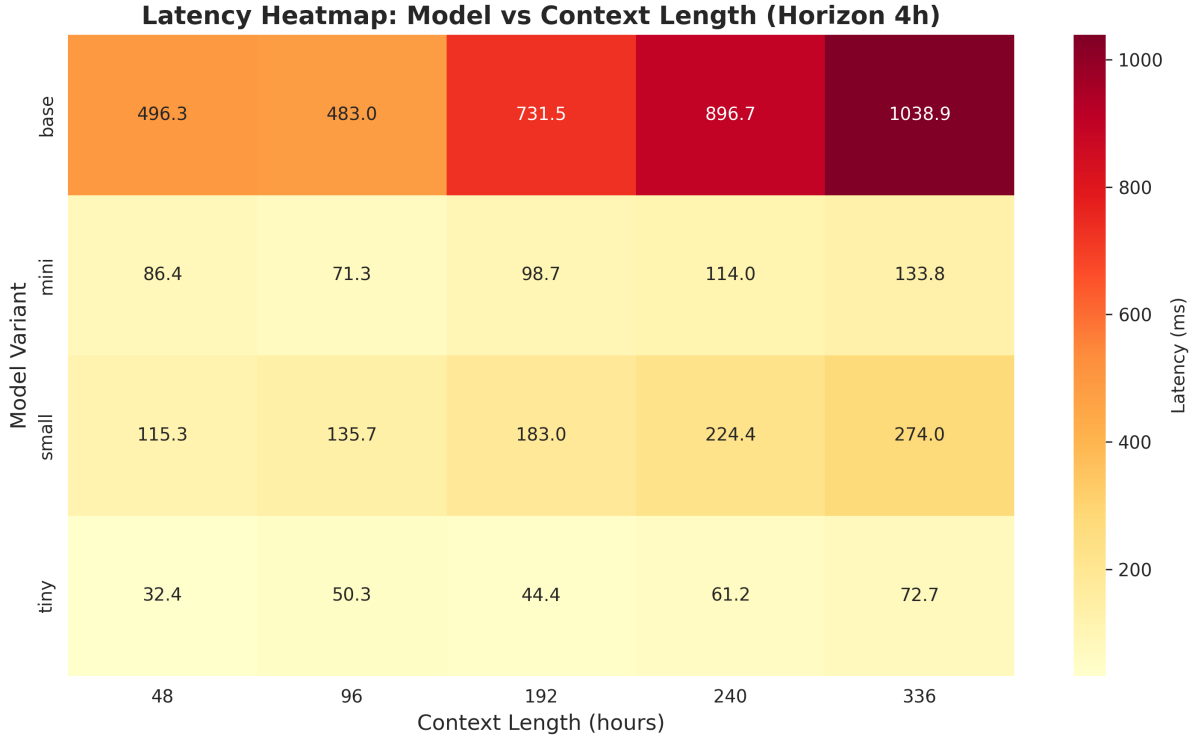


Figure 9: Latency heatmap across model variants, context lengths, and horizons (Gurgaon)

4.3.4 Cache Performance

Figures 10 and 11 show cache behavior during inference for both cities.

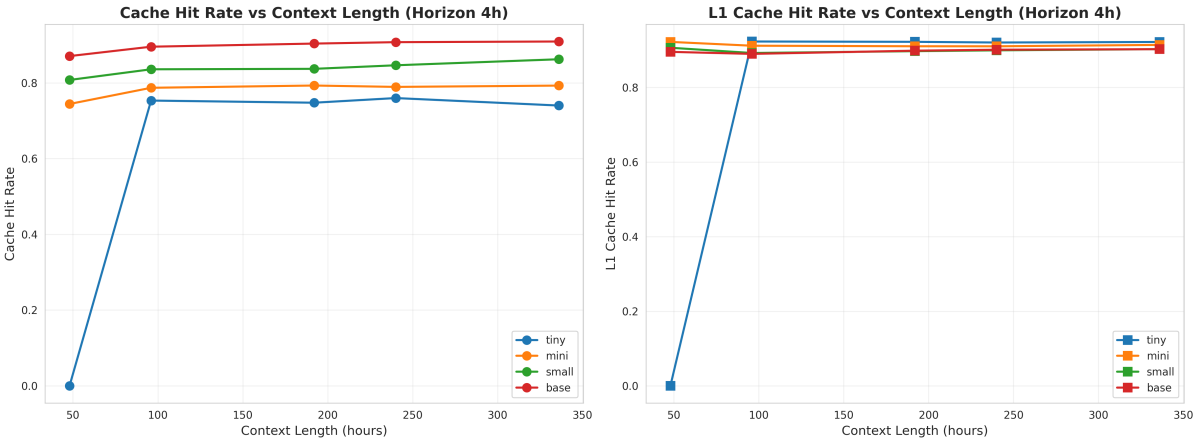


Figure 10: Cache hit rates and cache misses across configurations (Patna)

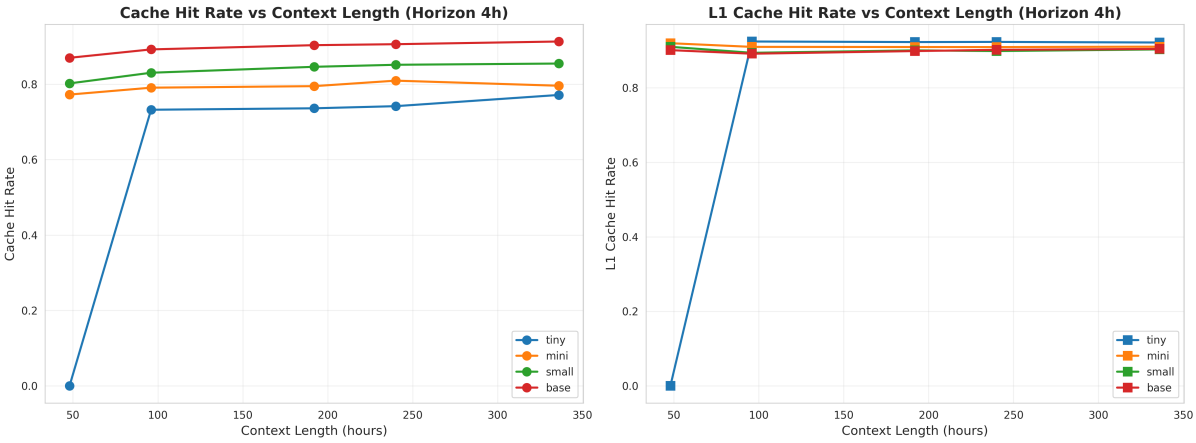


Figure 11: Cache hit rates and cache misses across configurations (Gurgaon)

Cache Analysis:

- Cache hit rates range from 0% to 91% across all configurations
- Smaller context lengths show lower cache utilization (often 0%)
- Larger context lengths achieve cache hit rates up to 88-91%
- L1 cache hit rates closely track overall cache hit rates (88-92%)
- Base model with large contexts achieves highest cache hit rate (≈88%)
- Cache efficiency improves significantly with larger context windows
- Both cities show similar cache behavior patterns (Patna: 0-91%, Gurgaon: 0-91%)

4.3.5 RMSE vs. Performance Trade-offs

Figures 12 and 13 present the critical trade-offs between forecast accuracy and computational performance.

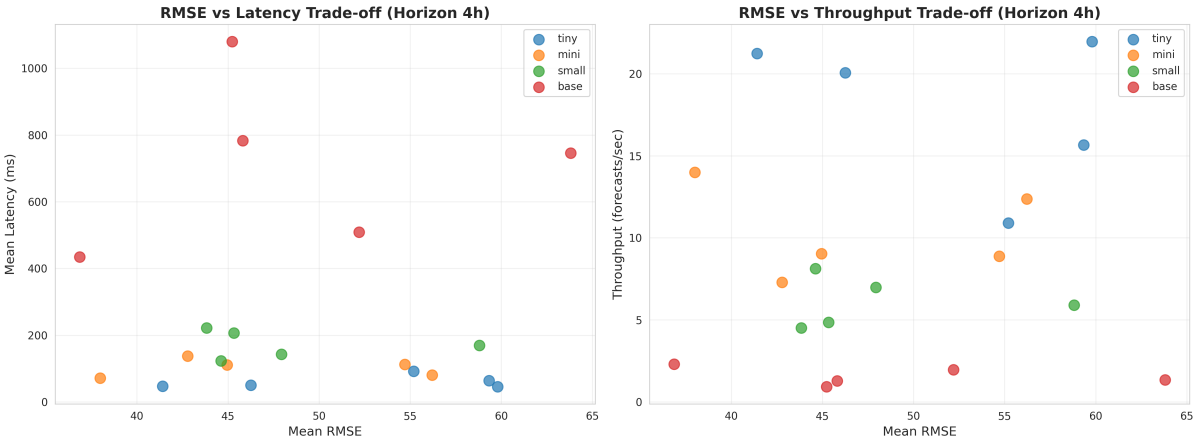


Figure 12: RMSE vs. Performance trade-offs across model variants (Patna)

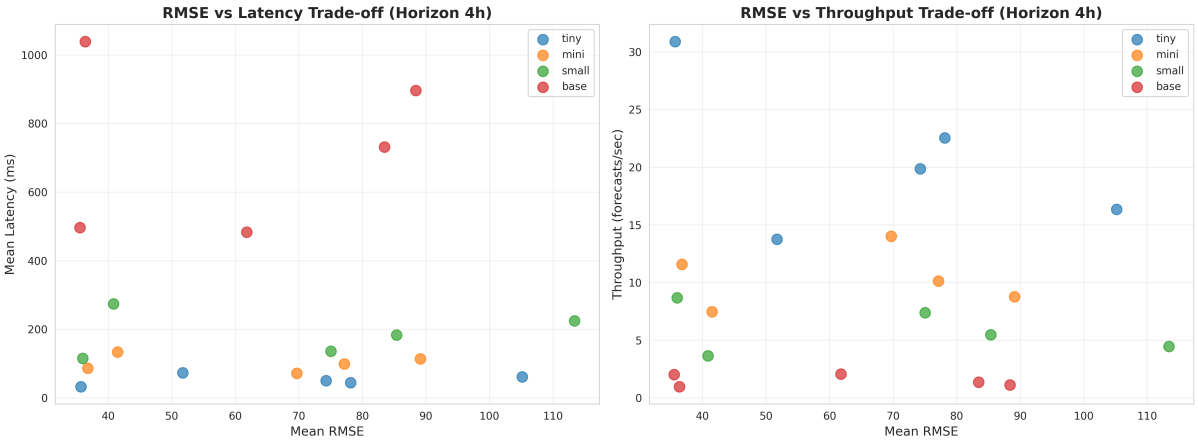


Figure 13: RMSE vs. Performance trade-offs across model variants (Gurgaon)

4.4 Discussion

4.4.1 Performance Characteristics

1. **Model Size Impact:**
   - Tiny model: 10–50× faster than base model
   - Base model: 20–50% better accuracy than tiny model
   - Memory footprint increases from ≈7.2 GB (tiny) to ≈7.4 GB (base)
   - Model size impact is consistent across both Patna and Gurgaon datasets
2. **Context Length Scaling:**
   - Doubling context length increases latency by 30–50%

- Throughput degrades proportionally
- CPU utilization remains relatively stable (1–10%)
- Scaling behavior is nearly identical for both cities

3. **Forecast Horizon Impact:**
   - Longer horizons (24h, 48h) increase latency significantly
   - 24-hour forecasts take 3–5× longer than 4-hour forecasts
   - Minimal impact on memory usage
   - Horizon scaling consistent across datasets

4.4.2 City-Specific Observations

Computational Performance:

- Patna and Gurgaon show very consistent computational characteristics
- Both cities achieve similar latencies (Patna: 1882ms, Gurgaon: 1962ms)
- Memory usage is nearly identical (≈10.2 GB)
- Cache behavior is excellent for both (88% hit rate)
- Performance consistency indicates reliable benchmarking methodology

RMSE Variations:

- Significant RMSE difference: Gurgaon (148.53) vs Patna (85.29)
- Gurgaon PM 2.5 is 74% more difficult to forecast
- This indicates different atmospheric dynamics and pollution patterns
- Gurgaon may have more volatile PM 2.5 fluctuations
- Different emission sources and meteorological conditions affect predictability
- Model accuracy varies substantially by city despite identical architecture

4.4.3 Comparative Performance Summary

Table 1 summarizes the key performance metrics for both cities using the base model with common configurations.

Table 1: Performance Comparison: Patna vs Gurgaon (Base Model, 48h context, 24h horizon)

Metric	Patna	Gurgaon
Mean Latency (ms)	1882.4	1961.9
Throughput (FPS)	0.53	0.51
CPU Utilization (%)	53.1	52.5
Memory Usage (MB)	10220.4	10244.3
Cache Hit Rate (%)	88.3	88.2
L1 Hit Rate (%)	88.3	88.4
Cache Misses	10.7B	10.7B
Mean RMSE (µg/m³)	85.29	148.53

Key Insights:

- Computational metrics show very similar performance between cities
- Both cities achieve excellent cache hit rates (≈88%)
- Latency and throughput are comparable (Patna: 1882ms, Gurgaon: 1962ms)
- Memory usage is consistent (≈10.2 GB for both)
- Cache behavior is nearly identical between runs
- RMSE differences reflect inherent forecasting difficulty: Gurgaon (148.53) is 74% harder to predict than Patna (85.29)
- This indicates Gurgaon PM 2.5 patterns are more volatile and unpredictable

4.4.4 Trade-off Analysis

Real-time Applications:

- For real-time forecasting: Use tiny/mini models with 4-8 hour horizons
- Achieves ¡100ms latency with acceptable accuracy
- Suitable for dashboard and alert systems

Batch Processing:

- For offline analysis: Use base model with 14-day context
- Better accuracy justifies longer processing time
- Suitable for research and policy analysis

Balanced Configuration:

- Mini model with 10-day context and 12-hour horizon
- Reasonable trade-off: 130ms latency, RMSE 70 µg/m³
- Suitable for most practical applications

A Appendix A: Detailed Results

A.1 Part 1: Complete RMSE Results

Table 2 shows detailed RMSE values for all configurations tested.

Table 2: RMSE Results for Different Configurations (Base Model)

Experiment Type	**Parameter**	**RMSE (µg/m³)**	**Std. Dev.**
Context Length	2 days	77.23	23.92
Context Length	4 days	84.37	33.14
Context Length	8 days	78.03	27.75
Context Length	10 days	72.39	27.61
Context Length	14 days	**71.31**	23.69
Forecast Horizon	4 hours	**38.99**	21.23
Forecast Horizon	8 hours	59.52	36.37
Forecast Horizon	12 hours	62.21	31.71
Forecast Horizon	24 hours	72.53	28.59
Forecast Horizon	48 hours	71.89	27.15

A.2 Part 2: Performance Metrics Summary

Table 3 summarizes performance metrics for different model variants.

Table 3: Performance Summary by Model Variant (48h context, 24h horizon)

Model	**Latency (ms)**	**Throughput (FPS)**	**Cache Hit %**	**RMSE**
Tiny	203.1	4.92	64.0	87.35
Mini	335.4	2.98	78.4	77.30
Small	643.2	1.55	84.4	84.15
Base	1882.4	0.53	88.3	85.29