Machine Learning for Network Traffic Prediction and Capacity Planning

Suryansh Singh Sijwali
Dept. of Computer Science
Penn State University
University Park, PA, USA
sss6371@psu.edu

***Abstract*—Accurate prediction of network traffic is a core enabler for intelligent capacity planning in backbone networks. Overestimating demand leads to expensive over-provisioning, while underestimating it causes congestion and packet loss. This paper studies how forecasting quality translates into capacity planning decisions by comparing a traditional statistical model—Seasonal ARIMA (SARIMA)—with a deep learning model—Long Short-Term Memory (LSTM)—on a synthetic but realistic backbone network. I design a complete simulation pipeline that generates multi-day traffic traces with diurnal patterns and burstiness on a 12-node topology, trains SARIMA per link and an LSTM jointly over all links, evaluates forecasting quality using RMSE, MAE, and MAPE, and finally quantifies capacity planning outcomes such as maximum link utilization and overload fraction. The results show that LSTM reduces RMSE by approximately 35% relative to SARIMA and cuts overload events by a factor of roughly four at similar capacity margins, demonstrating that modern sequence models can directly improve network engineering decisions and not just prediction accuracy.**

***Index Terms*—Network traffic prediction, capacity planning, SARIMA, LSTM, time series forecasting, backbone networks.**

I. Introduction

Modern communication networks carry highly variable traffic with strong temporal and spatial structure. Operators must determine how much capacity to deploy on each link to accommodate peak loads while keeping cost under control. Traditionally, capacity planning relies on heuristics, long-term averages, and conservative safety margins. However, as traffic patterns become more complex and applications more latency-sensitive, *data-driven* forecasting is increasingly important for making principled planning decisions rather than just "best guesses."

Network traffic exhibits well-known features such as diurnal cycles, weekly periodicity, and heavy-tailed bursts. [1] These properties motivate both classical time series models (e.g., ARIMA/SARIMA) and modern deep learning models (e.g., LSTM, graph neural networks) for traffic prediction. [2], [3], [5] From a statistics point of view, traffic prediction is a fairly standard forecasting problem; from a networking point of view, small differences in error can translate into very different engineering decisions.

In this project I wanted to connect those two perspectives: not just "which model has lower RMSE," but "does that actually change how much capacity I would provision on a backbone link?" To explore that question in a controlled setting, I built a Python simulation pipeline that generates synthetic traffic on a small backbone topology, trains both SARIMA and LSTM forecasts, and feeds the resulting predictions into a simple capacity planning rule. The complete, reproducible codebase is publicly available.[1]

A. Problem formulation and assumptions

I consider a backbone network with L links. At discrete times $t = 1, 2, \ldots$, we observe a vector of link loads

$$\mathbf{y}_t \in \mathbb{R}^L_{\geq 0},$$

where $y_{\ell,t}$ is the aggregate load on link ℓ during interval t. Given a history of past loads, the goal is to predict future loads:

$$\hat{\mathbf{y}}_{t+1} = f_\theta(\mathbf{y}_t, \mathbf{y}_{t-1}, \ldots, \mathbf{y}_{t-W+1}),$$

where W is the input window size and f_θ is a forecasting model with parameters θ.

Throughout this paper, I make the following assumptions:

- **Regularity.** Traffic exhibits approximate daily structure and slow evolution over the 14-day horizon.
- **Measurement availability.** Per-link load measurements are available at a fixed sampling interval (5 minutes).
- **Fixed topology.** The network topology and routing do not change over the forecasting window.
- **Single-step horizon.** I focus on one-step-ahead prediction at a 5-minute resolution.

I compare:

- a per-link SARIMA model, treating each $y_{\ell,t}$ as an independent univariate series;
- a joint LSTM model, trained across links.

II. Classification of ML Approaches for Traffic Prediction

This section classifies existing work on ML for traffic prediction, setting up the categories used in the rest of the

[1] Repository: https://github.com/SuryanshSS1011/ML-for-Network-Traffic-Prediction-and-Capacity-Planning

paper and giving some intuition on where different methods fit.

A. Model-based vs. data-driven axes

Two high-level axes are useful when zooming out over the literature.

1) Modeling paradigm: The first axis concerns *how* traffic is modeled:

- **Classical statistical and shallow ML models** (e.g., ARIMA, SARIMA, VAR, SVR): emphasize linear structure, interpretable parameters, and simple feature sets. [2], [7]
- **Deep sequence models** (e.g., LSTM, GRU, temporal CNNs): learn complex nonlinear temporal dependencies directly from data. [3], [5]
- **Graph-based spatio-temporal models** (GNNs): explicitly incorporate network topology as a graph, combining message passing and temporal modeling. [4], [14]
- **Data-centric and hybrid approaches**: focus on data representation, missing data, and combinations such as ARIMA + deep residuals. [10], [15]

2) Network context and prediction target: The second axis concerns *what* is being predicted and in which environment:

- **Context:** backbone/WANs, datacenters, access networks, or cellular systems. [5], [6]
- **Target:** link loads, traffic matrices (TMs), or flow-level metrics such as flow completion times.

Backbone and WAN settings often emphasize TMs and link loads; cellular work often focuses on per-cell load or user demand; datacenter studies may look at flow-level or rack-level metrics.

B. Evaluation metrics and pitfalls

Across these contexts, evaluation most often uses RMSE, MAE, and MAPE. [7] A few practical issues show up repeatedly:

- **MAPE on small values.** When y_i is close to zero, MAPE can explode; many works add a small ϵ or report MAPE only above a threshold.
- **Per-link vs. global metrics.** Global averages can hide large errors on individual links; per-link distributions are important for network engineers.
- **Train/test leakage.** Using future samples in normalization, or tuning hyperparameters on test sets, is surprisingly common in older work.
- **Operational relevance.** A model with slightly lower RMSE is not necessarily better if it leads to worse decisions downstream (e.g., unstable capacity plans).

These issues motivated the design of the experiment in Section IV: look at both error metrics and how those errors propagate into a simple capacity planning decision.

III. Modeling Categories: Details, Pros, and Cons

This section presents each modeling category in more detail, along with typical use cases, advantages, and limitations.

A. Classical statistical and shallow ML

Classical models such as ARIMA and SARIMA have been widely applied to network traffic. [1], [2] After differencing, the residual time series is assumed to be approximately stationary and modeled as a linear combination of past values and past noise.

Typical use. Per-link or aggregate traffic prediction in backbones and access networks; quick forecasting for capacity reporting and trend analysis.

Extensions. Multivariate time series models like vector autoregression (VAR) and state-space models can capture interactions between series, but they scale poorly as the number of links grows. [7]

Pros:

- Interpretable coefficients and clear statistical assumptions.
- Fast to fit and diagnose; well-tested tooling.
- Strong performance on smooth, strongly seasonal traffic.

Cons:

- Limited ability to capture nonlinear dynamics or sharp bursts.
- Usually treat each link independently, missing cross-link correlations.
- Order selection and seasonal structure must be chosen by the user or tuned per series.

Best suited for: relatively stable enterprise or backbone networks with clear seasonality and limited burstiness, where interpretability and speed are more important than squeezing out the last few percent of accuracy.

Shallow ML methods (SVR, random forests, gradient boosting) have also been applied. [7] They often use hand-crafted features (lags, moving averages, time-of-day indicators) and can be seen as a middle ground between purely linear models and deep learning.

B. Deep sequence models

Deep sequence models treat traffic as a multivariate time series and use architectures such as LSTMs, GRUs, or temporal CNNs. [3], [5], [6]

Aloraifan *et al.* [9] use deep neural networks to predict traffic matrices in large IP backbones, reporting significant improvements over ARIMA-style baselines. Hu *et al.* [10] propose a deep convolutional LSTM network for TM prediction with partial information, mixing spatial and temporal extraction.

Beyond these individual examples, several recent works apply deep architectures to traffic matrix prediction and network

optimization in a variety of settings. Nie *et al.* [8] introduce deep learning for TM prediction in large-scale IP backbone networks. Troia *et al.* [11] explore deep learning-based traffic prediction as a building block for network optimization in optical networks, while Lai *et al.* [12] develop a deep learning-based traffic prediction method for digital twin networks. Zhao *et al.* [13] apply neural traffic-matrix prediction to optical network-on-chip (ONoC), showing how similar ideas carry over to on-chip interconnects.

Pros:

- Capture nonlinear temporal dependencies and complex interactions between flows and links.
- Naturally handle multi-step outputs and multiple covariates.
- Often significantly outperform classical models on complex or bursty traffic. [7]

Cons:

- Require more data and careful hyperparameter tuning (learning rate, hidden size, regularization).
- Harder to interpret; model decisions are often opaque.
- Training and inference are more computationally intensive than fitting an ARIMA per link.

Best suited for: backbone, cellular, or datacenter environments where long traffic histories are available, traffic exhibits strong nonlinearity or bursts, and the operator is willing to deploy and maintain ML models.

C. Graph-based spatio-temporal models

Graph neural networks (GNNs) explicitly incorporate network topology. [4], [14] Nodes represent routers, PoPs, or regions; edges represent links. Spatio-temporal architectures apply graph convolutions at each timestep, then feed node embeddings through temporal modules.

Sun *et al.* [14] use a ChebNet-based architecture for traffic flow prediction with non-local spatial correlations. Similar ideas have been applied to transportation, cellular networks, and IP networks.

Pros:

- Capture spatial correlations and propagation of congestion naturally.
- Share information across nearby nodes/links in a structured way.
- Can encode topology changes explicitly.

Cons:

- More complex to implement, tune, and debug.
- Require a reasonably accurate and stable graph representation.
- Benchmarking and reproducibility are harder than with standard sequence models.

Best suited for: settings where topology effects are important (e.g., rerouting, link failures, spatially correlated bursts), and where operators already have good graph abstractions of their networks.

D. Data-centric and hybrid approaches

Data-centric and hybrid approaches focus on representation, missing data, and combining different model types. [7], [10], [15]

Examples include:

- Joint TM completion and prediction using deep architectures. [10]
- Treating traffic as images (e.g., time vs. OD pair) and applying CNNs. [15]
- Using deep models to learn residuals on top of ARIMA or Holt–Winters. [7]

Pros:

- Address practical issues such as missing counters and noisy measurements.
- Allow the combination of interpretable baselines and flexible deep models.

Cons:

- Pipelines can become complex, with multiple components to maintain.
- Reproducibility is harder; small implementation details matter.

Best suited for: real-world deployments where measurement gaps, noise, and changing instrumentation are the main pain points, and where incremental improvements over existing baselines are preferred over a full model replacement.

In the rest of the paper, I zoom in on one specific comparison inside this landscape: a classical per-link SARIMA model versus a deep sequence LSTM model, and I evaluate not just forecasting error but also capacity planning outcomes.

IV. Performance Comparison via Simulation

This section describes the simulation setup, forecasting models, evaluation metrics, and results. It corresponds to the "performance comparison" part of the assignment.

A. Simulation setup and key parameters

I simulate a backbone network with:

- $N = 12$ nodes (points-of-presence),
- undirected edges representing bidirectional links.

The topology is generated with `networkx` and stored in a compact NumPy format (`data/topology.npz`). Each undirected edge is modeled as a single link time series representing aggregate load in both directions.

I simulate traffic for:

- $T_{\text{days}} = 14$ days,
- sampling interval $\Delta t = 5\,\text{min}$,

yielding:

$$T = 14 \times 24 \times \frac{60}{5} = 4032$$

time steps per link.

Traffic on each link is generated via a simple model combining:

- link-specific baseline b_ℓ and amplitude A_ℓ,
- a smooth diurnal pattern $f_{\text{daily}}(t)$,
- Gaussian noise $\epsilon_\ell(t)$,
- occasional bursts $B_\ell(t)$.

I split the 14 days into 10 days for training and 4 days for evaluation. The key configuration parameters are summarized in Table I; they are defined in a central configuration file (`config.py`) so that the entire pipeline is reproducible.

TABLE I: Simulation and Model Configuration

Parameter	Value / Description
`num_nodes`	12 backbone nodes
`days`	14 total days
`time_step_minutes`	5-minute sampling
`window_size`	24 (2-hour LSTM input window)
`arima_order`	$(2, 1, 2)$
`seasonal_order`	$(1, 0, 1, 72)$ (6-hour seasonality)
`capacity_margin`	1.10 (10% safety margin)
LSTM hidden units	64
Optimizer	Adam
Loss	MSE (forecast vs. ground truth)
Train/eval split	10 days train, 4 days eval

B. Forecasting models

1) Per-link SARIMA: For each link ℓ, I fit a Seasonal ARIMA (SARIMA) model:

$$\text{SARIMA}(p, d, q) \times (P, D, Q)_s,$$

with:

- $(p, d, q) = (2, 1, 2)$,
- $(P, D, Q, s) = (1, 0, 1, 72)$, corresponding to a 6 h season at 5-minute resolution.

Model fitting is implemented using `statsmodels`. Each link is modeled independently; forecasts are produced for the 4-day evaluation period. I intentionally use a fixed order rather than `auto_arima` to keep the baseline simple, reproducible, and fast.

2) Joint LSTM forecaster: The LSTM forecaster is trained jointly on all links. For each link and time, I construct input windows of length $W = 24$ (2 hours) and predict the next load:

$$\hat{y}_\ell(t+1) = f_\theta(y_\ell(t-W+1), \ldots, y_\ell(t)).$$

The architecture is:

- input shape $(\text{batch}, W, 1)$,
- one LSTM layer with 64 hidden units,
- a fully-connected layer mapping the last hidden state to a scalar.

I standardize each link using its training mean and standard deviation, train with MSE loss and the Adam optimizer, and select the best checkpoint based on validation error. The model is implemented in PyTorch and saved as `models/lstm_forecaster.pt`.

C. Evaluation metrics

1) Forecasting metrics: I compute three standard error metrics across all forecasted points and links:

Root Mean Squared Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{pts}}} \sum_{i=1}^{N_{\text{pts}}} (y_i - \hat{y}_i)^2}.$$

Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{N_{\text{pts}}} \sum_{i=1}^{N_{\text{pts}}} |y_i - \hat{y}_i|.$$

Mean Absolute Percentage Error (MAPE):

$$\text{MAPE} = \frac{100}{N_{\text{pts}}} \sum_{i=1}^{N_{\text{pts}}} \left| \frac{y_i - \hat{y}_i}{y_i + \epsilon} \right|,$$

where ϵ is a small constant to avoid division by zero.

2) Capacity planning metrics: To connect forecasting quality to capacity planning, I assume that each operator sets a fixed capacity C_ℓ over the evaluation period based on predicted peak load:

$$\hat{y}_\ell^{\max} = \max_{t \in \text{eval}} \hat{y}_\ell(t), \quad C_\ell = \alpha \cdot \hat{y}_\ell^{\max},$$

with safety margin $\alpha = 1.10$.

Given true loads $y_\ell(t)$ and capacities C_ℓ, I define:

$$u_\ell(t) = \frac{y_\ell(t)}{C_\ell},$$

and measure:

- maximum utilization per link:

$$U_\ell^{\max} = \max_t u_\ell(t),$$

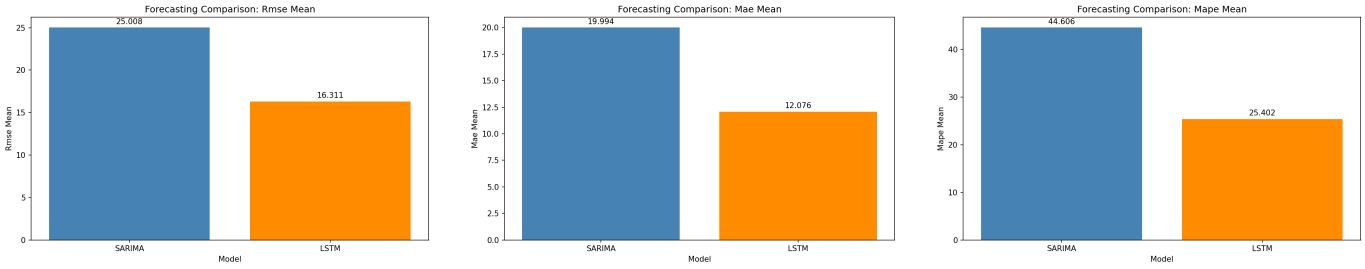
- overload fraction per link:

$$f_\ell^{\text{over}} = \frac{1}{T_{\text{eval}}} \sum_t \mathbf{1}\{u_\ell(t) > 1\}.$$

I then average these over links to obtain "Mean U_max" and "Mean Overload." An oracle baseline sets

$$C_\ell^{\text{oracle}} = \alpha \cdot \max_{t \in \text{eval}} y_\ell(t),$$

representing perfect forecasting ability.



(a) Mean RMSE (b) Mean MAE (c) Mean MAPE

Fig. 1: Forecasting error metrics for SARIMA and LSTM, averaged over all links and evaluation timesteps.

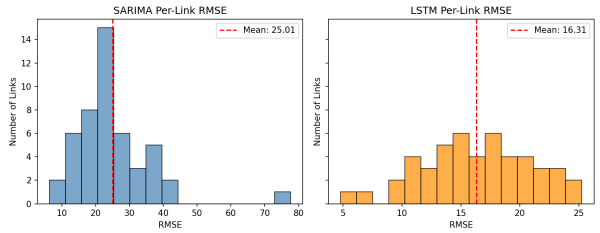


Fig. 2: Histogram of per-link RMSE for SARIMA and LSTM. The LSTM shifts most links toward lower error.

D. Results: forecasting accuracy

All steps are executed via:

```
python main.py
```

which orchestrates data generation, model training, evaluation, and plotting using the configuration in Table I.

Table II summarizes global forecasting performance.

TABLE II: Global Forecasting Metrics (Lower is Better)

Model	RMSE	MAE	MAPE
SARIMA	25.01	19.99	44.6%
LSTM	**16.31**	**12.08**	**25.4%**

The LSTM achieves a substantial reduction in error:

- approximately 35% lower RMSE than SARIMA,
- approximately 40% lower MAPE.

To examine variability across links, Fig. 2 shows the distribution of per-link RMSE values. The LSTM curve is clearly shifted toward lower error bins: visually, a large majority of links see a reduction in RMSE under LSTM, with only a small minority where SARIMA is slightly better or roughly tied.

From a "stats" perspective, the takeaway is that using the same data and similar modeling cost per time step, a relatively small LSTM can substantially reduce error across multiple metrics and across most individual links, not just on average.

E. Results: capacity planning outcomes

The capacity planning metrics are given in Table III. Fig. 3 shows all three capacity-related metrics as a single three-panel figure for easier comparison.

TABLE III: Capacity Planning Metrics (Derived from Forecasts)

Model	Mean U_max	Mean Overload
SARIMA	2.79	40.3%
LSTM	1.94	10.9%
Oracle	1.59	2.4%

Interpretation:

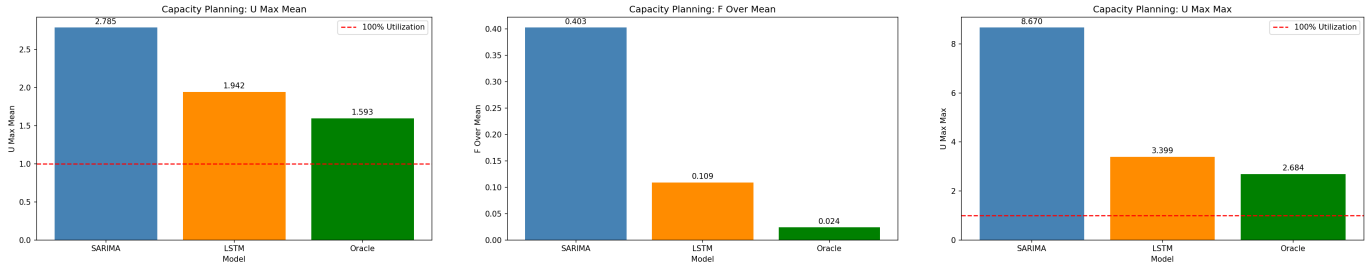
- **SARIMA**: To avoid overloads, capacity must be set high relative to true peaks, resulting in average maximum utilizations close to 2.8 and frequent overload events (about 40% of the evaluation period on average). The worst-case link experiences even higher peak utilization, as shown in Fig. 3c.
- **LSTM**: Better forecasts allow capacity to be closer to true demand. Maximum utilization drops to about 1.94 on average, with overloads reduced to around 11%. Both the mean overload fraction and the worst-case utilization move closer to the oracle baseline.
- **Oracle**: With perfect knowledge of future peaks, both over-provisioning and overloads can be minimized; this provides a lower bound and shows remaining room for model improvement.

Looking at Fig. 3 overall, the visual story matches the numbers: for every capacity-related metric, the LSTM sits noticeably closer to the oracle than SARIMA does.

F. Representative time-series examples

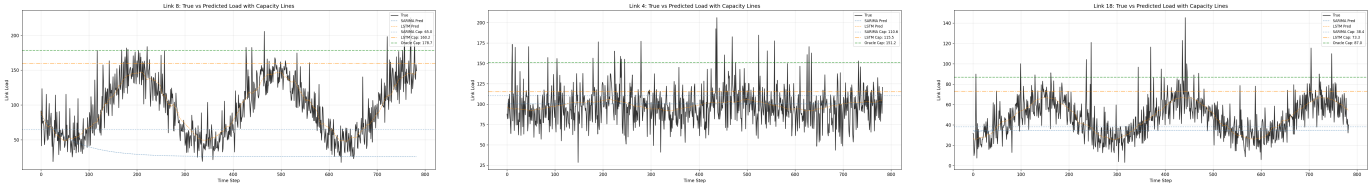
Finally, I inspect individual links to see when each model performs well. Fig. 4 shows three representative cases.

The left panel shows a link with pronounced peaks and bursts where LSTM tracks the true load much more closely than SARIMA. The middle panel shows a smoother link where SARIMA's linear structure is adequate and sometimes slightly



(a) Mean maximum utilization (b) Mean overload fraction (c) Worst-case max utilization

Fig. 3: Capacity planning metrics under SARIMA, LSTM, and Oracle capacities.



(a) Link with strong LSTM advantage (b) Link where SARIMA slightly wins (c) Link with median difficulty

Fig. 4: Representative links illustrating model behavior across different traffic patterns.

better. The right panel is a "typical" link showing moderate but consistent LSTM improvements.

These examples support the idea that in a real deployment, operators might selectively apply deep models to more challenging links while leaving simple models on well-behaved ones.

G. Discussion and link to the survey

The experiment highlights several design trade-offs:

- **Model complexity vs. robustness.** SARIMA is simple and interpretable but misses complex nonlinear effects; LSTM adapts better to bursts but needs more careful tuning.
- **Per-link vs. joint training.** The per-link SARIMA fits each series in isolation, while the LSTM jointly sees many different patterns. This joint training likely helps the LSTM generalize.
- **Error metrics vs. decisions.** A 30–40% reduction in RMSE/MAPE can look abstract; seeing overload drop from 40% to 11% makes the engineering impact concrete.

Conceptually, the survey in Sections 2 and 3 motivated the choice of SARIMA and LSTM as representative classical and deep sequence models. The simulation in this section then concretely shows how their qualitative pros and cons play out in a capacity planning scenario, turning the literature discussion into a specific, measurable comparison.

V. Conclusion and Emerging Trends

This work builds an end-to-end simulation framework to study how forecasting models shape capacity planning in backbone networks. On a synthetic 12-node topology with 14 days of generated traffic, I compare SARIMA and LSTM forecasters and show that:

- LSTM substantially reduces prediction errors (RMSE, MAE, MAPE) relative to SARIMA.
- These improvements translate into lower over-provisioning and significantly fewer overload events at the same safety margin.
- An oracle baseline shows further headroom, motivating more advanced models and topology-aware architectures.

From a broader perspective, the project reinforced a simple but important point: in network engineering, the value of a "better" model is only as real as the operational decisions it changes. By explicitly connecting forecasts to a capacity rule and then measuring overloads, it becomes much clearer why a 10–20% difference in error can matter a lot in practice.

A. Threats to validity and limitations

There are several limitations and potential threats to validity:

- **Synthetic traffic.** The experiments use synthetic traffic with hand-crafted diurnal patterns and bursts; real traces may exhibit different multi-scale dynamics, abrupt regime changes, or anomalies.
- **Single-step forecasting.** I only evaluate one-step-ahead predictions at 5-minute resolution. Multi-step forecasting (e.g., 1 hour ahead) may amplify error differences and change the relative ranking of models.
- **Single model configurations.** The study uses one fixed SARIMA order and a single LSTM architecture without extensive hyperparameter sweeps. Tuning either model

could change absolute numbers and possibly narrow or widen the gap.

- **Single safety margin.** Capacity is derived from forecasts using a single safety margin $\alpha = 1.10$. In reality, operators might vary safety margins across links or use more complex cost/risk trade-offs.

These choices were deliberate to keep the project tractable and transparent, but they should be kept in mind when interpreting the results.

B. Emerging trends and future directions

Recent surveys and research point to several trends that go beyond the simple SARIMA vs. LSTM comparison here:

- **Graph-based and topology-aware models.** GNNs and spatio-temporal graph models are increasingly used to exploit network structure directly. [4], [14]
- **Data-centric design.** Handling missing counters, noisy measurements, and changes in measurement methodology is as important as the model architecture. [7], [10]
- **End-to-end integration.** There is growing interest in models that are tightly integrated with routing, admission control, and other control-plane decisions.

Natural next steps for this project include using real backbone traces instead of synthetic traffic, incorporating graph neural networks to exploit topology directly, forecasting multiple steps ahead, or experimenting with different safety margins and cost models. The current pipeline is deliberately modular so that those ideas can be plugged in without rewriting everything from scratch.

REFERENCES

[1] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, "Long-term forecasting of Internet backbone traffic," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1110–1124, Sep. 2005.

[2] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, NJ, USA: Wiley, 2015.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[5] M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep learning for network traffic monitoring and analysis (NTMA): A survey," *Computer Communications*, vol. 170, pp. 19–41, Mar. 2021.

[6] A. Chen, J. Law, and A. L. Anpalagan, "A survey on traffic prediction techniques using artificial intelligence for communication networks," *Telecom*, vol. 2, no. 4, pp. 518–535, Dec. 2021.

[7] G. O. Ferreira, C. Ravazzi, F. Dabbene, G. C. Calafiore, and M. Fiore, "Forecasting network traffic: A survey and tutorial with open-source comparative evaluation," *IEEE Access*, vol. 11, pp. 6018–6044, 2023.

[8] L. Nie, D. Jiang, L. Guo, and S. Yu, "Traffic matrix prediction and estimation based on deep learning in large-scale IP backbone networks," *Journal of Network and Computer Applications*, vol. 76, pp. 16–26, Oct. 2016.

[9] D. Aloraifan, I. Ahmad, and E. Alrashed, "Deep learning based network traffic matrix prediction," *International Journal of Intelligent Networks*, vol. 2, pp. 46–56, 2021.

[10] V. A. Le, P.-L. Nguyen, and Y. Ji, "Deep convolutional LSTM network-based traffic matrix prediction with partial information," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Arlington, VA, USA, Apr. 2019, pp. 261–269.

[11] S. Troia, R. Alvizu, Y. Zhou, G. Maier, and A. Pattavina, "Deep learning-based traffic prediction for network optimization," in *Proceedings of the 20th International Conference on Transparent Optical Networks (ICTON)*, Bucharest, Romania, Jul. 2018, pp. 1–4.

[12] J. Lai, Z. Chen, J. Zhu, W. Ma, L. Gan, S. Xie, and G. Li, "Deep learning based traffic prediction method for digital twin network," *Cognitive Computation*, vol. 15, pp. 1748–1766, 2023.

[13] J. Zhao, H. Li, and F. Liu, "Neural network-based traffic matrix prediction incorporating inter-flow correlations for optical network-on-chip (ONoC)," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Gold Coast, Australia, Jun. 2023, pp. 1–8.

[14] Q. Sun, R. Tian, X. Jia, Q. Li, and L. Sun, "ChebNet traffic flow prediction model based on non-local spatio-temporal correlation matrix," *Networks and Spatial Economics*, vol. 25, pp. 935–956, 2025.

[15] Z. Zhang, D. Shi, Y. Zhang, and Y. Sun, "Network traffic prediction by learning time series as images," *Engineering Science and Technology, an International Journal*, vol. 55, p. 101754, 2024.