TimeRAN: Unifying Time-Series Analytics for Next-Generation Radio Access Networks

Abstract

The Radio Access Network (RAN) is evolving into a programmable and disaggregated infrastructure that increasingly relies on AI-native algorithms for optimization and closed-loop control. However, current RAN intelligence is still largely built from task-specific models tailored to individual functions, resulting in model fragmentation, limited knowledge sharing across tasks, poor generalization, and increased system complexity. To address these limitations, we introduce **TimeRAN**, a unified multi-task learning framework for time-series modeling in the RAN. `TimeRAN` leverages a lightweight time-series foundation model with few task-specific heads to learn transferable representations that can be efficiently adapted across diverse tasks with limited supervision. To enable large-scale pretraining, we further curate and open-source **TimeRAN DataPile**[1], the largest time-series corpus for RAN analytics to date, comprising over **355K** time series and **0.56B** measurements across diverse telemetry sources, protocol layers, and deployment scenarios. We evaluate `TimeRAN` across a comprehensive set of RAN analytics tasks, including *anomaly detection*, *classification*, *forecasting*, and *imputation*, and show that it achieves state-of-the-art performance with minimal or no task-specific fine-tuning. Finally, we integrate `TimeRAN` into a proof-of-concept 5G testbed and demonstrate that it operates efficiently with limited resource requirements in real-world scenarios.

1 INTRODUCTION

The Radio Access Network (RAN) is undergoing a fundamental architectural transformation toward 6G, evolving from a monolithic and inflexible system into a disaggregated, multi-purpose infrastructure capable of jointly supporting diverse functionalities, including communication, computation, and sensing [26]. In this context, AI-RAN has emerged as a promising paradigm that re-architects the RAN as a programmable computing platform built on general-purpose accelerated hardware capable of supporting both cellular and AI workloads simultaneously [3]. Leveraging this architecture, a growing body of work has begun designing, integrating, and evaluating AI-native RAN algorithms to enhance network performance and efficiency. Representative examples include autoencoder-based techniques for efficient channel estimation [48], reinforcement learning–based schedulers for radio resource allocation and control [38], and AI-driven solutions for higher-layer RAN functions such as anomaly detection [34], and mobility management [33].

Despite these promising advances, several important challenges still remain largely unexplored and unresolved. More specifically, current approaches predominantly focus on designing task-specific learning-based modules for highly specialized RAN functions, leading to a proliferation of models that must be deployed, maintained, and orchestrated across the RAN, thereby increasing system complexity and operational overhead [39]. In addition, these methods lack mechanisms for effectively sharing learned representations across tasks, requiring separate models per use case and resulting in redundant learning and inefficient use of training and computational resources [38]. Along the same lines, current deployed models exhibit limited representation capacity, leading to poor generalization, as models trained in specific operational environments often fail to transfer across distributed cell sites with diverse traffic patterns and radio propagation conditions, frequently requiring full retraining. Finally, these challenges are further exacerbated by the limited computational resources available at the network edge.

A promising direction to address these issues is to leverage *foundation model–inspired paradigms*, which have shown strong capability in learning rich contextual representations and enabling generalization across diverse downstream tasks with limited supervision [53]. Despite their success across domains, their adoption in telecommunications remains relatively limited. The main reasons are threefold: (i) constrained data availability, as RAN data are often proprietary and not widely accessible [39], (ii) the inherent complexity of modeling such data, which consist of stochastic, high-dimensional multivariate time series with varying temporal granularities [16], and (iii) limited computational resources at cell sites, which further complicates the deployment of large and computationally intensive foundation models.

Inspired by these limitations, we introduce `TimeRAN`, a unified multi-task learning framework designed to eliminate model fragmentation, enable cross-task generalization, and reduce system complexity for scalable deployment of learning-based RAN intelligence. `TimeRAN` employs a lightweight transformer-based time-series architecture to learn generalizable representations from RAN telemetry, coupled with a few task-specific heads for efficient adaptation across diverse network applications. To enable representation learning at scale, we curate and *open-source* the largest time-series corpus for RAN analytics to date, *TimeRAN DataPile* [4], comprising over 355K time series and 0.56B measurements across

[1]The `TimeRAN DataPile` corpus and the pretrained `TimeRAN` model weights are available at https://anonymous.4open.science/r/TimeRAN

diverse telemetry sources, RAN protocol layers, and deployment scenarios, and use it for training. We evaluate TimeRAN across a comprehensive set of RAN analytics tasks, including *anomaly detection*, *classification*, *forecasting*, and *imputation*, achieving state-of-the-art performance with minimal supervision. Finally, we integrate TimeRAN into a proof-of-concept 5G testbed, demonstrating low-latency inference with a limited computational footprint. More specifically, this work makes the following key contributions:

- We introduce TimeRAN, a unified multi-task learning-based architecture to eliminate task-specific model fragmentation and learn generalizable representations across diverse RAN tasks and deployments.
- We curate and open-source the *TimeRAN DataPile*, the largest time-series corpus for RAN analytics to date, to enable large-scale pretraining and benchmarking.
- We evaluate TimeRAN across a comprehensive set of RAN tasks, demonstrating state-of-the-art performance with minimal fine-tuning and supervision.
- We integrate TimeRAN into a real over-the-air 5G testbed, demonstrating low-latency inference with limited compute requirements in real-world scenarios.
- We release the pretrained TimeRAN weights to enable further research and experimentation.

2 BACKGROUND

RAN Architecture. The RAN acts as the wireless interface connecting the User Equipment (UE) to the cellular core network, supporting user-plane data transmission and control plane operations. In 5G, the RAN protocol stack is disaggregated into three main units: the *Radio Unit (RU)*, the *Distributed Unit (DU)*, and the *Centralized Unit (CU)*. The RU performs radio-frequency processing and lower Physical Layer (PHY) functions at the cell site, while the DU executes latency-critical baseband processing, including upper PHY operations, along with the Medium Access Control (MAC) and Radio Link Control (RLC) layers. Higher protocol layers, such as the Packet Data Convergence Protocol (PDCP) and Radio Resource Control (RRC), are handled by the CU under more relaxed latency constraints. This decomposition corresponds to the widely adopted *functional split 7.2* [2]. Finally, these units can be virtualized and deployed on commodity infrastructure, enabling flexible placement and improved compute utilization through resource pooling [36].

Foundation Models for the RAN. Foundation models have emerged as a powerful paradigm for learning representations from large-scale data via self-supervised objectives, enabling a wide range of downstream tasks within a unified framework [53]. Initially developed for natural language processing, this paradigm has since been extended to multiple modalities, including vision, speech, and time-series data [6]. Regardless of the input modality, foundation models typically follow a common processing pipeline: raw inputs are first converted into modality-specific tokens, then projected into high-dimensional embeddings, and finally processed by stacked transformer layers [47] comprising self-attention mechanisms and feedforward networks, enabling the modeling of complex contextual dependencies. In RAN systems, where large volumes of heterogeneous, high-dimensional telemetry are continuously generated, such models provide a natural approach for learning generalizable representations.

3 RELATED WORK

RAN intelligence, primarily driven by time-series telemetry, has emerged as an active research area focused on extracting actionable insights from high-dimensional RAN data to enable efficient network monitoring, optimization, and control [7]. Common approaches include statistical [28], reconstruction-based [46], and sequential deep learning models [10, 15, 24, 31, 33, 35] for anomaly detection and forecasting; deep neural network architectures for classification [23, 34, 35, 38, 52]; and reconstruction methods for imputation [28, 29, 46]. Despite their competitive performance, these models are often tightly coupled to specific operational environments, requiring continuous retraining or fine-tuning under changing conditions, resulting in substantial engineering overhead and maintenance costs [38, 39]. Recent work has begun exploring unified learning paradigms, including foundation-model-based approaches, to learn transferable representations [11, 16, 49]. Preliminary results demonstrate promising performance across a relatively narrow set of tasks. However, existing works are not specifically designed for RAN settings and remain limited in scope, failing to systematically examine whether and how such models can be effectively adapted to the unique characteristics of RAN. Therefore, this gap motivates the need for unified models that support scalable and generalizable learning in the RAN.

4 TIMERAN DATAPILE

To address the lack of large-scale data for pretraining and representation learning in the RAN domain, we collate multiple telemetry datasets from widely used public repositories and augment them to form a unified and comprehensive corpus, which we refer to as the *TimeRAN DataPile*[2]. Our corpus supports a broad set of learning tasks, including *anomaly detection, classification, imputation, and forecasting*, while providing time series measurements spanning the entire RAN and UE protocol stacks. It comprises approximately *29 GB* of telemetry data, including *355K* unique time series

[2]We envision *TimeRAN DataPile* as a reference corpus for RAN downstream tasks, analogous in spirit to foundational datasets such as ImageNet [12] in computer vision and The Pile [18] in natural language processing.

and *0.56 billion* timestamps collected from both operational cellular networks and over-the-air experimental testbeds.

The *TimeRAN DataPile* exhibits diversity in both channel[3] dimensionality and temporal granularity. Channel dimensionality ranges from a few variables to high-dimensional telemetry with hundreds of variables, reflecting the heterogeneity across RAN deployments and configurations. Temporal resolution spans multiple monitoring scales, from fine-grained millisecond-level measurements to coarser second- and minute-level intervals, capturing both short-term dynamics and long-term trends. Time-series lengths also vary significantly, ranging from short traces with limited observations to long sequences comprising millions of samples. In terms of dataset composition, *TimeRAN DataPile* includes 9 anomaly detection datasets comprising 23M observations across 21K unique time series; 7 classification datasets with 5M observations across 248 time series, covering tasks such as mobility classification, service identification, congestion detection, traffic classification, and root-cause analysis; and 25 datasets for imputation and forecasting, forming the largest portion of the corpus, with 269M observations across 167K unique time series. Further analytical details on the corpus, curation, and processing pipeline are provided in [4].

5 TIMERAN DESIGN & ARCHITECTURE

5.1 Overview

Building on the large-scale *TimeRAN DataPile*, we next introduce `TimeRAN`, a unified learning-based framework that supports a broad range of RAN tasks through a shared, lightweight multi-task architecture, focusing on higher-layer RAN functions that do not require strict real-time inference.

As illustrated in Fig. 1, `TimeRAN` consists of multiple modules including: ❶ a *Multi-channel Patching* module that segments multivariate RAN telemetry into fixed-length temporal patches; ❷ a *Projection* module that maps the patches into a shared latent representation space; ❸ a *Masking* module, activated only during pre-training, that enables self-supervised learning by selectively masking portions of the patches; ❹ a *Positional Encoding* module that preserves the temporal ordering of the patches; ❺ a *Transformer Encoder* backbone that captures temporal dependencies and cross-channel correlations through stacked self-attention layers; and ❻ a set of *Task-Specific Heads* that operate on the shared latent representation to enable diverse RAN analytics tasks.

5.2 Design Requirements

We design *TimeRAN* with the following key requirements.
1) Unified multi-task learning. The system should follow a *"one encoder, many tasks"* paradigm, where a shared backbone is reused across many downstream tasks.

[3]Throughout this paper, we use the term *channel* to denote an individual variable (i.e., one variate) of a multivariate time series.

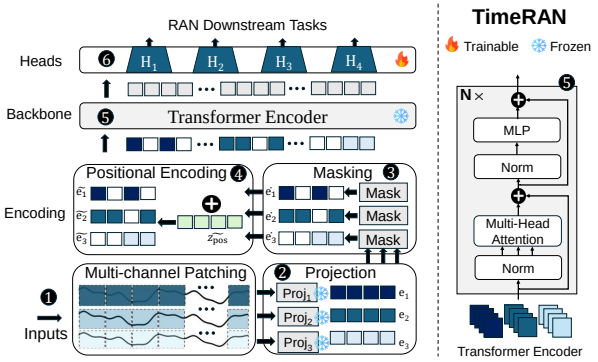


Figure 1: TimeRAN system architecture.

2) Generalization and efficient adaptation. The framework should learn robust representations that generalize across heterogeneous and unseen environments, while enabling efficient adaptation to new tasks through zero-shot or lightweight fine-tuning without retraining from scratch.
3) Computational efficiency. The system should operate efficiently in operational deployments, such as at the network edge or at cell sites with limited computational resources.

5.3 TimeRAN Architecture

`TimeRAN` adapts a modular architecture. More specifically:

5.3.1 Patching and Projection Modules.

The first components of `TimeRAN` are the *Patching and Projection* modules, which convert multivariate RAN telemetry into high-dimensional embeddings that serve as input tokens for the subsequent modules. More specifically, let $\mathbf{X} \in \mathbb{R}^{C \times T}$ denote the telemetry collected over an observation window of length T, where C represents the number of channels. First, each channel is partitioned into non-overlapping temporal windows of length P, referred to as *patches*. Accordingly, each channel $c \in \{1, \ldots, C\}$ is divided into $N = \lfloor T/P \rfloor$ patches, where the i-th patch is defined as $\mathbf{x}_c^{(i)} \in \mathbb{R}^{1 \times P}$. Next, each patch is normalized and projected into a d-dimensional latent space through a trainable linear projection layer, producing an embedding $\mathbf{e}_c^{(i)} \in \mathbb{R}^{1 \times d}$. Finally, the resulting embeddings across all channels are organized into a sequence $\mathbf{E} \in \mathbb{R}^{(C \cdot N) \times d}$, which is then forwarded to the next module.

5.3.2 Positional Encoding Module.

To preserve temporal ordering, positional embeddings are added to the patch embeddings before being processed by the *Transformer Encoder*. Specifically, this module takes as input a sequence of embeddings $\mathbf{E} \in \mathbb{R}^{(C \cdot N) \times d}$ and adds a positional embedding to each patch embedding to inject temporal order information. The resulting position-aware embeddings are then forwarded to the *Time-Series Transformer Encoder*.

5.3.3 Time-Series Transformer Encoder.

Building on the previous stages, the *Time-Series Transformer Encoder* takes as input the position-aware embeddings and processes them to model temporal dependencies and cross-channel

correlations in the multivariate input telemetry. TimeRAN employs a lightweight encoder based on the vanilla Transformer [47], adapted for time-series data. More precisely, the encoder consists of L stacked transformer layers, each comprising multi-head self-attention followed by a position-wise feed-forward network, with residual connections and normalization. Within each layer, the input representation $\mathbf{E}$ is projected into queries, keys, and values as $Q = \mathbf{E}W_Q$, $K = \mathbf{E}W_K$, and $V = \mathbf{E}W_V$. Scaled dot-product attention is then computed as $\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V$. To capture diverse temporal dependencies, the model employs B attention heads operating in parallel on different representation subspaces; the outputs of these heads are concatenated and linearly projected to form the final attention representation. Through this attention mechanism, each embedding attends to all others in the sequence, enabling the model to capture long-term temporal dependencies across the observation window. The encoder therefore produces contextualized representations for all embeddings, denoted as $\mathbf{Z} \in \mathbb{R}^{(C \cdot N) \times d}$.

5.3.4 Task-Specific Heads. Finally, the contextualized patch embeddings produced by the *Time-Series Transformer Encoder* are consumed by lightweight task-specific heads to enable multiple downstream tasks. TimeRAN supports four time-series tasks, namely *anomaly detection*, *classification*, *forecasting*, and *imputation*, and employs specialized heads for each objective. All heads share a similar architecture, realized as a stack of linear layers M that operate on the *Transformer Encoder* outputs to produce task-specific predictions. For classification tasks, a global representation is first obtained by aggregating the contextualized patch embeddings. Since no dedicated summary token (e.g., a [CLS] token [13]) is used, the global representation is computed via mean aggregation as $z = \frac{1}{C \cdot N} \sum_{i=1}^{C \cdot N} Z_i \in \mathbb{R}^{1 \times d}$, which is then used by the classification head to learn a mapping to the target classes. In contrast, the remaining heads operate directly on the full sequence of contextualized patch embeddings.

5.3.5 Self-Supervised Pre-training via Masking. To learn generalizable representations, TimeRAN is trained using a masked time-series modeling objective enabled by the *Masking Module*, which is activated only during pre-training. First, a masking ratio ρ is applied to the sequence of patch embeddings produced by the *Projection Module*, where a subset of patches is selected uniformly at random and replaced with a learnable mask token. The partially masked sequence is then processed by the *Time-Series Transformer Encoder* to produce contextualized latent representations. Finally, TimeRAN employs a lightweight reconstruction head that acts as a decoder to reconstruct the input time series from the encoded representations Z. It learns accurate representations by minimizing the Mean Squared Error (MSE) between the ground-truth and reconstructed time-series over the masked positions, encouraging robust contextual learning.

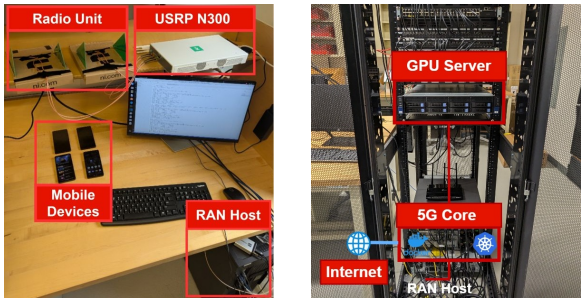
5.3.6 Fine-tuning on RAN Downstream Tasks. TimeRAN supports multiple RAN time-series analytics tasks as mentioned earlier. For anomaly detection and imputation, it leverages the reconstruction head to predict missing values and identify anomalies in the time series. For anomaly detection, the entire input time series is reconstructed, and anomalies are identified based on the reconstruction error aggregated over channels[4]. In contrast, for forecasting, TimeRAN replaces the reconstruction head with a forecasting head, which first flattens all the patch embeddings into a $C \times N \times d$ dimensional vector, and then projects it into a $C \times H$ dimensional output (values to be forecasted), where H denotes the forecasting horizon. For classification, TimeRAN uses a classification head that projects the averaged global representation $z \in \mathbb{R}^{1 \times d}$ into a vector of class logits. Finally, TimeRAN can be fine-tuned under different regimes: (i) end-to-end fine-tuning ($\text{TimeRAN}_{\text{FF}}$), (ii) linear probing ($\text{TimeRAN}_{\text{LP}}$), where only the task-specific head is trained while the encoder is frozen, and (iii) zero-shot inference (TimeRAN_0), which is only supported for anomaly detection and imputation, where the pre-trained reconstruction head is used without additional training. We also explored parameter-efficient fine-tuning using LoRA [25]; however, its performance was not competitive with other regimes and is therefore omitted.

6 SYSTEM IMPLEMENTATION

Prototype Implementation. We implemented TimeRAN in approximately ~10K lines of Python and Shell code. The system is implemented in Python using the PyTorch and HuggingFace Transformers libraries, while Shell scripting is employed to collect fine-grained runtime resource metrics.

System Integration. We integrated TimeRAN in a proof-of-concept 5G testbed, as illustrated in Fig. 2. The setup consists of a disaggregated 5G base station (Fig. 2(a)) coupled with a Software-Defined Radio (SDR) frontend (USRP N300), a cloud-native 5G core (Fig. 2(b)), and four Google Pixel 7 mobile devices enabling fully over-the-air experimentation. The RAN baseband functions run on a high-performance computing platform equipped with an AMD Ryzen Threadripper PRO 5975WX processor (32 cores) and 504 GB of DDR4 memory. GPU acceleration is provided by separate GPU servers equipped with NVIDIA RTX 3090, RTX 4090, and RTX A6000 GPUs. The core network runs on a separate server with an AMD EPYC 7352 24-core processor and 128 GB of DDR4 memory. The base station connects to the SDR via a dedicated 10 Gbps fronthaul link, with the SDR interfaced to 6 dBi directional antennas, and communicates with the core

[4]Estimating optimal thresholds for anomaly detection is beyond the scope of this study and is left for future work.



(a) 5G RAN. **(b)** 5G Core Network.

Figure 2: Proof-of-concept over-the-air 5G testbed.

through a 10 Gbps backhaul link. Both the RAN and core network are implemented using the OAI software stack [32]; the core network follows a cloud-native microservice-based architecture where all the virtual network functions, including the AMF, SMF, NRF, UPF, UDM, UDR, and AUSF, are deployed as containerized pods in an on-premises Kubernetes cluster. The system operates in the n78 TDD band with 20 MHz bandwidth centered at 3.319 GHz and follows a 5 ms radio frame structure configured with seven downlink slots and two uplink slots with 30 kHz subcarrier spacing.

7 EVALUATION SETUP

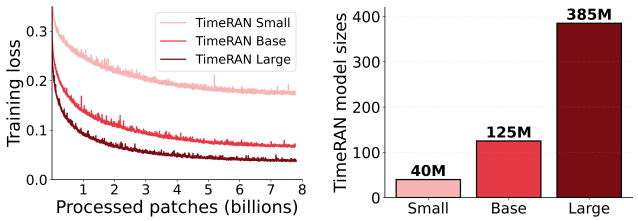
7.1 Corpus Partitioning and Benchmarking

We construct disjoint training and test splits, for each dataset in the *TimeRAN DataPile*. When official splits are provided by the dataset creators, we adopt them directly; otherwise, we apply a 70%/30% train–test split strategy. For long continuous time-series datasets, we perform a temporal split, assigning the first 70% of the sequence to training and the remaining 30% to testing. For datasets consisting of multiple independent short time-series, we assign each complete time series exclusively to either the training or the test set.

During pre-training, we restrict `TimeRAN` to the training splits of each dataset, excluding both test splits and anomaly detection datasets to prevent information leakage and ensure that only normal temporal patterns are learned. For evaluation, we also construct a comprehensive and diverse benchmark using a representative subset of the *TimeRAN DataPile* datasets. As illustrated in Table 1, five datasets are selected for each downstream task to capture diverse telemetry characteristics, temporal granularities, and operational conditions across the RAN protocol stack.

7.2 Configuration and Training Settings

TimeRAN Configuration. We leverage the pretrained encoder of MOMENT [22] as the backbone of our *Time-Series Transformer Encoder*, due to its state-of-the-art performance achieved through training on billions of time-series samples across diverse domains, as well as its lightweight architecture, which enables efficient learning. We then perform continued



(a) Training loss. **(b) Number of parameters.**

Figure 3: Training loss and size of TimeRAN variants.

pre-training on the *TimeRAN DataPile* to adapt the model to RAN telemetry data. We train three `TimeRAN` variants with capacities aligned to the sizes of the T5 encoder [41]. Specifically, the `Base` (`Small`, `Large`) configuration employs a transformer encoder with $L = 12$ ($\underline{6}$, $\overline{24}$) layers, embedding dimension $d = 768$ ($\underline{512}$, $\overline{1024}$), $B = 12$ ($\underline{8}$, $\overline{16}$) attention heads, and feed-forward dimensions of 3072 ($\underline{2048}$, $\overline{4096}$), resulting in approximately 125 ($\underline{40}$, $\overline{385}$) million parameters. Each variant processes multivariate time-series inputs with observation window length of $T = 512$, which are then partitioned into $N = 64$ non-overlapping patches of length $P = 8$. Prior to projection, patches are normalized using reversible instance normalization. Sinusoidal absolute positional encodings are added to the patch embeddings to preserve temporal ordering across the sequence. During continued pre-training, $\rho = 30\%$ of the patch embeddings are randomly masked following a uniform sampling strategy, enabling self-supervised learning over the input sequence. Finally, all task-specific heads employ $M = 2$ linear layers.

Training Setup. `TimeRAN` variants are trained using the Adam optimizer with decoupled weight decay, with $\lambda = 0.05$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Gradients are clipped at a maximum norm of 5.0, and training is performed with a batch size of 2048. We employ a cosine learning rate schedule with initial and final learning rates of 10^{-4} and 10^{-5}, respectively. During pre-training, we improve sample efficiency via overlapping patching with stride values $\{1, 2, 4\}$, selected based on dataset size, with smaller datasets using finer strides and larger datasets using coarser strides. Dataset heterogeneity is addressed through a size-aware sampling strategy that inversely reweights datasets, preventing dominance of large datasets during continued pre-training. Training efficiency and memory usage are further optimized through gradient

Tasks	Supervision	Datasets
Anomaly Detection	Zero-shot Linear probing	AERPAW 18/23 [5, 43], BLT [15], Hetnets [10], Jamshield [35]
Classification	Linear probing Full-finetuning	AERPAW 20 [42], Irish [40], Spotlight [46], TelecomTS [16], Tractor [23]
Forecasting	Linear probing Full-finetuning	AERPAW 18/24/25 [5, 44, 45], QoE Aware [27], Queens [14]
Imputation	Zero-shot Linear probing	Irish [40], Open Ireland [52], Hetnets [10], WINS [19], NOK [17]

Table 1: TimeRAN evaluation benchmark.

checkpointing and mixed-precision training. All model variants are trained for one full epoch in a mixed-precision setting, using `float32` for numerically unstable operations and `bfloat16` for all other computations. Fig. 3 illustrates the convergence of the training loss of the TimeRAN variants.

Fine-tuning Setup. Unless stated otherwise, we employ `TimeRAN-Base` (approximately 125M parameters, requiring less than `500 MB` in `float32`), which provides a favorable balance between performance, training efficiency, and resource footprint, making it well-suited for deployments with limited compute and memory resources. `TimeRAN-Large` achieves higher performance, as illustrated in Fig. 3, though full results are omitted due to space limitations. Finally, detailed configurations of the fine-tuning strategies and task-specific hyperparameters are provided in [4].

7.3 Baselines

We compare TimeRAN with state-of-the-art deep learning and statistical models across tasks. These include pretrained general-purpose time-series foundation models (e.g., MOMENT [22]), transformer-based time-series architectures (e.g., Autoformer [50], Informer [54], TimesNet [51]), deep learning sequential models [30] (e.g., 1D-CNN, LSTM), and classical statistical methods [8] (e.g., ARIMA, ETS). Notably, LSTM- and CNN-based approaches are widely employed for time-series tasks in the telecommunications domain, and we adopt architectures consistent with those reported in prior work [10, 23, 35]. For a fair comparison, we use model configurations with parameter sizes comparable to `TimeRAN-Base` for the transformer architectures. Detailed architectural and hyperparameter settings for all methods are provided in [4].

7.4 Evaluation Metrics

We assess TimeRAN using standard task-specific metrics commonly adopted in the time-series literature. For anomaly detection, we use the Adjusted Best F1 score [21], which selects the optimal threshold over reconstruction-based anomaly scores (e.g., MSE) to maximize the F1 score, where an anomaly segment is considered correctly detected if any timestep within the segment is identified. For classification, we report Precision (P), Recall (R), and F1-score. For imputation and forecasting, we use MSE and Mean Absolute Error (MAE) to evaluate prediction accuracy.

8 EVALUATION RESULTS

Our evaluation is guided by the following research questions:

RQ1: How well does TimeRAN perform across diverse RAN downstream tasks (§8.1)?

RQ2: To what extent does TimeRAN learn generalizable and transferable representations across heterogeneous datasets and previously unseen environments (§8.2)?

RQ3: What resource overheads does TimeRAN incur (§8.3)?

RQ4: How does TimeRAN perform in real-world scenarios across multiple downstream tasks (§9)?

8.1 Downstream Task Performance

To answer RQ1, we evaluate TimeRAN across multiple downstream tasks using the benchmark described in Section §7.1. As the selected datasets in the curated benchmark capture different target variables and scales, we refer the reader to the corresponding dataset references for further details.

8.1.1 Anomaly Detection. TimeRAN consistently outperforms all baselines in anomaly detection (Table 2). In particular, $\text{TimeRAN}_{\text{LP}}$ achieves the strongest performance, indicating that representations learned from normal RAN time series enable accurate anomaly detection without full fine-tuning. Furthermore, TimeRAN remains effective in the zero-shot setting, surpassing most methods, with consistent gains across both univariate and multivariate time series datasets.

8.1.2 Classification. For classification, TimeRAN with full fine-tuning achieves state-of-the-art performance across representative tasks (Table 3). Moreover, deep learning sequential models often outperform transformer-based time-series baselines under limited labeled data due to stronger inductive biases and more data-efficient training. Nevertheless, TimeRAN remains highly adaptable and consistently matches or exceeds their performance when fully fine-tuned.

8.1.3 Forecasting. In forecasting, TimeRAN achieves satisfactory performance across datasets and horizons (Table 4). While no single method dominates, TimeRAN remains competitive without extensive fine-tuning. Performance degrades with increasing forecasting horizon for all methods, reflecting the inherent uncertainty of long-range prediction. In this regime, $\text{TimeRAN}_{\text{LP}}$ outperforms both fully fine-tuned TimeRAN and MOMENT variants, while deep learning approaches consistently outperform statistical methods due to the high dimensionality and stochasticity of RAN telemetry.

8.1.4 Imputation. Finally, TimeRAN achieves notable imputation performance across masking ratios (Table 5). As the masking ratio increases, reconstruction becomes more challenging, leading to degradation across all methods. Nevertheless, TimeRAN remains robust under both training settings, consistently outperforming most statistical methods.

Overall, the results indicate that TimeRAN provides a unified and effective solution across diverse RAN downstream analytics tasks, achieving strong performance under both minimal adaptation and full fine-tuning strategies.

8.2 Generalization Analysis

We next evaluate the ability of TimeRAN to learn representations that generalize across unseen environments. In particular, we examine both (i) the separability of learned representations without task-specific fine-tuning and (ii) the

Methods Dataset	TimeRAN$_{LP}$ Adj. F1	TimeRAN$_{0}$ Adj. F1	MOMENT$_{LP}$ Adj. F1	MOMENT$_{0}$ Adj. F1	Autoformer Adj. F1	Informer Adj. F1	TimesNet Adj. F1
AERPAW 18 [5]	**0.93**	0.86	0.83	0.90	0.91	0.90	0.89
AERPAW 23 [43]	**0.99**	0.98	0.98	0.98	0.93	0.94	0.94
BLT [15]	**0.98**	0.95	0.93	0.91	0.86	0.89	0.83
Hetnets [10]	**0.99**	0.99	0.98	0.98	0.96	0.95	0.96
Jamshield [35]	**0.97**	0.93	0.91	0.90	0.92	0.92	0.93

Table 2: Anomaly detection performance on the curated evaluation benchmark.

Methods		TimeRAN$_{FF}$			TimeRAN$_{LP}$			MOMENT$_{FF}$			Informer			TimesNet			1D-CNN			LSTM		
Dataset	**Task**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**
AERPAW 20 [42]	Location	**1.00**	**0.99**	**0.99**	0.62	0.80	0.70	1.00	0.98	0.99	0.65	0.76	0.71	0.61	0.69	0.68	0.95	0.90	0.93	0.68	0.72	0.71
Irish[40]	Mobility	**1.00**	**0.99**	**0.99**	0.96	0.89	0.92	0.99	0.99	0.99	0.95	0.91	0.92	0.96	0.92	0.94	0.98	0.99	0.98	0.96	0.93	0.95
Spotlight [46]	Anomaly	0.93	0.94	0.93	0.84	0.83	0.84	**0.93**	**0.95**	**0.94**	0.83	0.89	0.86	0.81	0.89	0.84	0.78	0.92	0.81	0.90	0.83	0.87
	Root-Cause	**0.99**	**0.99**	**0.99**	0.55	0.45	0.40	0.96	0.95	0.95	0.57	0.58	0.58	0.61	0.59	0.57	0.63	0.66	0.61	0.64	0.65	0.64
TelecomTS [16]	Congestion	**0.98**	**1.00**	**0.99**	0.58	0.95	0.72	0.97	0.98	0.98	0.74	0.99	0.85	0.72	0.99	0.84	0.87	0.93	0.90	0.75	0.99	0.86
	Coverage	0.88	0.89	0.88	0.54	0.55	0.55	0.81	0.81	0.80	0.80	0.81	0.79	0.82	0.81	0.82	0.81	0.80	0.81	**0.90**	**0.89**	**0.89**
Tractor [23]	Services	0.97	0.97	0.97	0.75	0.72	0.71	**0.98**	**0.98**	**0.98**	0.70	0.68	0.68	0.70	0.62	0.61	0.90	0.88	0.87	0.81	0.79	0.79
	Slice Type	**0.99**	**0.99**	**0.99**	0.83	0.81	0.80	0.98	0.98	0.98	0.92	0.91	0.91	0.89	0.90	0.90	0.96	0.97	0.96	0.94	0.94	0.94

Table 3: Classification performance on the curated evaluation benchmark.

Methods		TimeRAN$_{FF}$		TimeRAN$_{LP}$		MOMENT$_{FF}$		Autoformer		Informer		TimesNet		1D-CNN		LSTM		ETS		ARIMA	
Dataset	**Horizon (H)**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
AERPAW 18 [5]	32	25.55	3.86	23.55	3.78	31.92	4.33	21.70	3.55	31.30	4.16	22.78	3.67	28.63	4.08	**20.43**	**3.40**	34.32	4.61	31.78	4.33
	64	26.46	3.99	26.29	3.97	31.61	4.32	**22.79**	**3.66**	23.47	3.77	24.46	3.82	27.41	4.05	22.92	3.70	38.81	4.78	34.12	4.60
	128	28.17	4.07	25.15	3.86	27.45	4.05	23.36	3.76	24.63	3.84	24.11	3.81	25.66	3.93	**23.26**	**3.73**	39.21	4.81	37.21	4.71
	208	28.69	4.10	25.71	3.93	27.32	4.04	25.25	3.87	25.81	3.96	**24.40**	**3.81**	24.69	3.85	24.84	3.87	41.44	5.11	40.11	5.01
AERPAW 24 [45]	32	17.70	3.53	14.75	3.14	19.00	3.56	16.86	3.50	20.26	3.79	**14.27**	**3.10**	27.33	4.39	15.45	3.18	29.80	4.65	26.48	4.32
	64	17.72	3.53	14.67	3.13	20.17	3.77	23.77	3.97	16.48	3.23	**14.19**	**3.09**	27.36	4.40	16.65	3.43	44.94	5.13	44.11	5.10
	128	19.67	3.74	17.02	3.51	**14.11**	**3.07**	23.37	3.95	16.87	3.50	19.13	3.71	25.58	4.11	19.01	3.69	68.26	6.35	69.90	6.41
	208	36.79	5.07	27.08	4.37	**13.99**	**2.96**	25.74	4.27	19.76	3.74	22.72	3.85	29.60	4.61	30.90	4.84	110.22	8.21	109.92	8.13
AERPAW 25 [44]	32	6.35	2.06	5.79	1.93	6.25	2.05	5.27	1.86	5.25	1.85	5.32	1.86	5.92	1.95	**5.18**	**1.83**	5.96	2.01	6.00	2.01
	64	6.56	2.07	5.85	1.94	6.43	2.06	5.57	1.90	5.27	1.86	5.31	1.86	5.81	1.93	**5.23**	**1.85**	6.01	2.02	6.08	2.02
	128	6.83	2.08	5.89	1.95	6.56	2.07	5.70	1.92	5.29	1.86	5.34	1.87	5.83	1.94	**5.28**	**1.86**	6.08	2.03	6.17	2.04
	208	6.85	2.08	5.90	1.95	6.53	2.07	5.52	1.90	5.29	1.86	5.31	1.86	5.90	1.95	**5.29**	**1.86**	6.12	2.03	6.23	2.05
QoE Aware [27]	32	48.81	4.98	**46.93**	**4.91**	51.03	5.15	52.15	5.21	51.05	5.15	50.88	5.14	61.06	5.64	47.13	4.93	71.63	6.01	54.68	5.31
	64	56.23	5.42	**50.05**	**5.10**	56.98	5.41	53.21	5.26	52.48	5.22	52.42	5.22	62.36	5.70	50.68	5.11	84.51	6.41	62.96	5.72
	128	60.17	5.60	**52.63**	**5.24**	60.50	5.59	54.46	5.31	54.18	5.30	54.22	5.31	63.70	5.76	53.32	5.26	95.13	6.90	68.68	5.88
	208	62.27	5.68	**54.18**	**5.31**	62.85	5.71	55.16	5.35	55.12	5.34	55.24	5.35	64.60	5.80	54.82	5.33	101.24	7.16	73.02	6.07
Queens [14]	32	17.29	3.03	17.88	3.14	**16.62**	**2.88**	46.75	5.05	27.64	3.91	27.04	3.86	46.77	5.09	22.20	3.40	17.94	3.18	16.98	2.93
	64	24.76	3.64	23.89	3.63	**23.63**	**3.42**	50.32	5.21	35.39	4.38	36.18	4.43	51.77	5.36	29.34	3.94	26.50	3.76	26.03	3.66
	128	35.91	4.42	**32.29**	**4.24**	33.45	4.30	55.46	5.55	44.97	4.99	45.83	5.03	59.50	5.75	40.39	4.47	42.53	4.90	42.32	4.49
	208	48.32	5.17	**42.47**	**4.86**	44.39	4.92	60.57	5.79	53.65	5.49	53.46	5.46	63.78	5.95	52.10	5.40	64.48	5.99	64.20	5.98

Table 4: Forecasting performance on the curated evaluation benchmark.

Methods		TimeRAN$_{LP}$		TimeRAN$_{0}$		MOMENT$_{FF}$		Forward Fill		Mean		N. Neighbors		Linear		Rolling Mean	
Dataset	**Mask Ratio**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**	**MSE**	**MAE**
Irish [40]	10%	**4.95**	**1.44**	5.16	1.48	23.04	3.02	13.02	2.34	20.82	2.92	10.80	1.94	8.45	1.86	12.18	2.26
	30%	**5.96**	**1.56**	6.21	1.60	22.55	2.98	16.72	2.53	24.59	3.07	14.50	2.40	11.86	2.00	14.68	2.42
	50%	**7.38**	**1.75**	8.47	1.87	24.40	3.08	16.77	2.57	23.67	3.05	15.56	2.49	12.40	2.28	14.90	2.45
Open Ireland [52]	10%	21.02	2.54	**20.80**	**2.53**	23.64	2.78	62.46	4.38	34.52	3.78	62.33	4.37	52.35	4.17	34.46	3.61
	30%	22.75	2.69	**22.67**	**2.68**	24.60	2.87	62.80	4.40	34.90	3.80	62.55	4.38	52.16	4.16	35.03	3.80
	50%	**24.99**	**2.88**	25.04	2.90	26.53	3.03	63.79	4.45	35.04	3.81	63.21	4.41	52.90	4.18	35.82	3.84
Hetnets [10]	10%	**0.58**	**0.56**	0.67	0.59	3.42	1.17	7.21	1.65	8.84	2.29	5.64	1.38	4.40	1.31	6.01	1.47
	30%	0.78	0.61	**0.76**	**0.60**	4.13	1.29	7.70	1.67	9.38	2.32	6.22	1.50	5.06	1.36	6.27	1.53
	50%	1.03	0.65	**1.01**	**0.64**	4.55	1.34	8.29	1.71	9.12	2.31	7.07	1.62	5.70	1.40	6.52	1.58
WINS [19]	10%	246.03	5.88	**144.70**	**5.22**	280.53	6.37	1669.83	16.87	8094.32	48.35	701.80	10.68	442.69	7.62	2712.92	24.01
	30%	534.79	8.78	**523.31**	**8.75**	597.65	9.21	2268.78	20.24	7883.61	47.87	1175.44	13.83	864.97	11.24	3172.77	26.68
	50%	**1104.23**	**12.98**	1122.85	13.19	1182.96	13.86	3365.68	27.11	8004.40	48.02	2072.07	18.72	1615.33	16.27	3985.17	30.01
NOK [17]	10%	6.10	0.69	5.78	0.66	8.72	0.86	9.15	0.93	76.56	5.38	5.75	0.64	**5.43**	**0.58**	20.31	2.31
	30%	9.87	1.12	**7.07**	**0.79**	10.32	1.23	12.64	1.40	61.93	5.14	11.51	1.28	7.60	0.82	17.47	1.93
	50%	11.67	1.31	**9.13**	**0.90**	13.70	1.80	19.33	1.87	60.71	5.13	16.30	1.87	11.40	1.26	23.45	2.84

Table 5: Imputation performance on the curated evaluation benchmark.

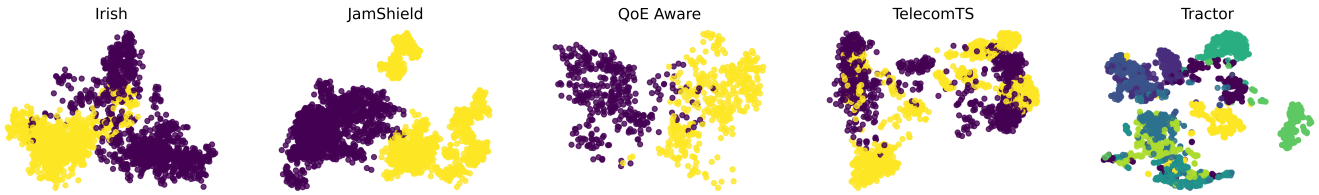
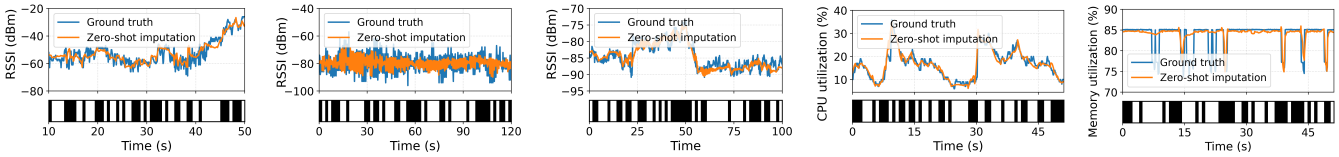
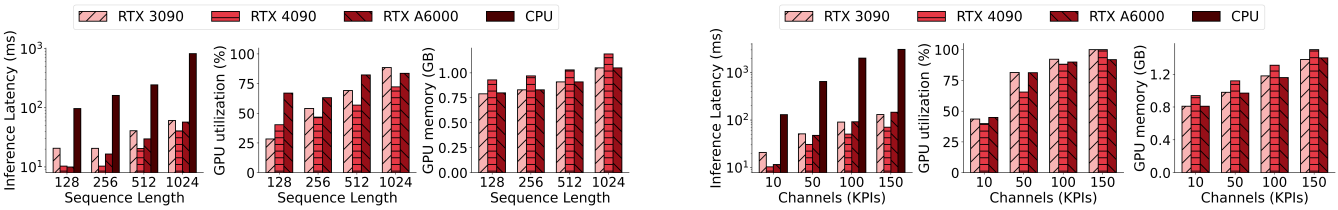


Figure 4: TimeRAN learns separable representations in a zero-shot setting without dataset-specific fine-tuning.



(a) WiFi. **(b) Bluetooth.** **(c) LoRa.** **(d) CPU utilization.** **(e) Memory utilization.**

Figure 5: TimeRAN generalizes to unseen environments in a zero-shot setting.



(a) Impact of input window size. **(b) Impact of channel dimensionality.**

Figure 6: System-level performance of TimeRAN under varying input characteristics during inference.

effectiveness of zero-shot downstream task inference in domains that differ from those observed during training.

8.2.1 Representation Separability. Fig. 4 visualizes the latent representations learned by TimeRAN across several datasets using dimensionality reduction techniques, specifically PCA [37], to project the learned high-dimensional embeddings into a lower-dimensional space. Despite operating in a zero-shot setting, the learned embeddings exhibit clear clustering structures corresponding to different classes. For example, mobility states, anomalies, and service types form well-separated clusters across the Irish [40], JamShield [35], QoE Aware [27], TelecomTS [16], and Tractor [23] datasets. These results indicate that TimeRAN learns transferable temporal representations that enable simple linear classifiers to separate distinct data patterns in the latent space.

8.2.2 Zero-Shot Generalization to Unseen Environments. We further assess the zero-shot generalization capability of TimeRAN on downstream tasks using data not encountered during either pre-training or fine-tuning, focusing on imputation as a representative case. Fig. 5 presents results across diverse datasets spanning multiple wireless environments, including WiFi [9], Bluetooth [1], and LoRa [20], as well as system-level telemetry collected from the base station (Distributed Unit) of our testbed. These datasets are selected to cover domains with temporal characteristics similar to RAN telemetry as well as fundamentally different sources, as described in [1, 9, 20]. In each case, we randomly mask 50% of the input time series and reconstruct the missing segments (white regions). Across all scenarios, TimeRAN accurately recovers the masked portions, closely matching the original input time series.

Overall, these results demonstrate that TimeRAN learns transferable representations that can generalize across heterogeneous environments and extend beyond the RAN domain.

8.3 System Overhead During Inference

In addition, we evaluate the resource overhead introduced by TimeRAN by examining how the input window size T and channel dimensionality C impact inference latency and overall resource utilization across different computing platforms.

8.3.1 Impact of Input Window Size. Fig. 6(a) depicts system performance as the input window size T increases from 128 to 1024, with the number of input channels fixed at $C = 30$. As expected, inference latency increases with window size as the self-attention layers must process longer temporal contexts. For example, latency on the RTX A6000 increases from approximately 30 ms at window size 128 to about 70 ms at 1024, while CPU inference exceeds 800 ms. GPU utilization rises from roughly 30% to over 90%, reflecting the increased compute workload. GPU memory consumption also increases gradually with window size but remains relatively modest, reaching approximately 1.1 GB. CPU memory remains largely unchanged, with 1 core used for data transfer, as computation is offloaded to the GPU.

8.3.2 Impact of Input Channel Dimensionality. Fig. 6(b) illustrates the effect of increasing the number of input channels $C \in \{10, 50, 100, 150\}$ while keeping the window size fixed ($T = 512$). As channel dimensionality increases, inference latency and GPU utilization rise due to the larger input representation processed by TimeRAN. For instance, latency on the RTX A6000 increases from roughly 40 ms at 10 channels to around 120 ms at 150 channels, while CPU latency grows to over 2000 ms. GPU utilization approaches 100% at higher dimensionalities, and GPU memory usage increases to approximately 1.3,GB, remaining relatively modest overall.

9 EVALUATION IN REAL 5G TESTBED

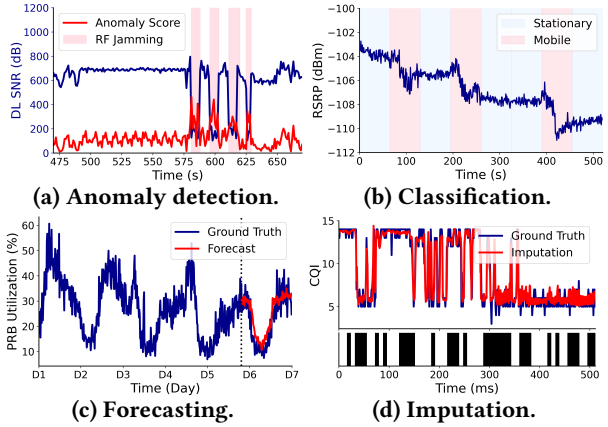
Finally, we integrate TimeRAN into our proof-of-concept 5G testbed and evaluate it across all supported RAN downstream tasks, selecting a representative real-world use case scenario for each task. All subsequent experiments are conducted using live telemetry collected directly from our private 5G network deployment, where over 20 cross-layer RAN Key Performance Indicators (KPIs) are continuously monitored at a fine-grained temporal granularity of 10 ms. Across all these tasks, TimeRAN processes the telemetry using a sliding window of length $T = 512$ (corresponding to 5.12 s).

9.1.1 Anomaly Detection. For anomaly detection, we induce a realistic RF jamming attack using a USRP X310 SDR positioned approximately 5 m from the mobile devices, transmitting over-the-air interference on the data channels at maximum gain. TimeRAN produces a normalized anomaly score in the range $[0, 1]$, where higher values correspond to abnormal network behavior. As shown in Fig. 7(a), the anomaly score sharply increases during the jamming attack, accurately identifying it in real time.

9.1.2 Classification. For classification, we consider a mobility detection task in which a single UE moves within the laboratory (approximately 80 m^2), with the goal of classifying the user state as either stationary or mobile. By jointly analyzing RSRP measurements and additional PHY-layer telemetry indicators, TimeRAN reliably distinguishes between stationary and mobile user states, as illustrated in Fig. 7(b).

9.1.3 Forecasting. For forecasting, we consider a scenario in which all mobile devices are attached to an enhanced Mobile Broadband (eMBB) slice while generating downlink traffic through YouTube video streaming. The objective is to predict the future network load at the base station, measured in terms of Physical Resource Block (PRB) utilization. Using historical telemetry observations, TimeRAN accurately forecasts future PRB utilization and reliably captures temporal fluctuations in network load, as illustrated in Fig. 7(c).

9.1.4 Imputation. For imputation, we consider a scenario in which a user with a mobile device moves within the laboratory while Channel Quality Indicator (CQI) measurements



(a) Anomaly detection. (b) Classification.

(c) Forecasting. (d) Imputation.

Figure 7: TimeRAN performance across representative RAN analytics tasks on our over-the-air 5G testbed.

are partially masked from the telemetry stream with a masking ratio of 50% (white regions). TimeRAN reconstructs the missing values from the partially observed time series. As shown in Fig. 7(d), the model accurately recovers the masked segments while preserving the temporal CQI dynamics.

These results demonstrate that TimeRAN reliably supports diverse RAN tasks directly from live telemetry, enabling real-time inference within operational 5G deployments.

10 CONCLUSION

In this work, we introduced TimeRAN, a unified multi-task learning framework for the convergence of RAN downstream tasks. Leveraging large-scale pretraining on the *TimeRAN DataPile*, TimeRAN learns transferable representations that generalize across diverse tasks and environments with minimal supervision. Extensive evaluations and real-world testbed results demonstrate state-of-the-art performance alongside low-latency inference with minimal resource overhead.

REFERENCES

[1] 2019. BLE RSSI dataset for Indoor localization. UCI Machine Learning Repository. doi:10.24432/C5B62T
[2] 2023. *NG-RAN; Architecture Description.* Technical Report TS 38.401. 3rd Generation Partnership Project (3GPP).
[3] AI-RAN Alliance. 2024. Shaping Future AI-Native Networks. White Paper. https://ai-ran.org/
[4] Anonymous Authors. 2026. TimeRAN DataPile and Model Weights. https://anonymous.4open.science/r/TimeRAN.
[5] Ram Asokan et al. 2024. Aerial RF and Throughput Measurements on a Non-Standalone 5G Wireless Network. In *2024 IEEE International Symposium on Dynamic Spectrum Access Networks.* 120–123.
[6] Muhammad Awais et al. 2025. Foundation Models Defining a New Era in Vision: A Survey and Outlook. *IEEE Trans. Pattern Anal. Mach. Intell.* 47, 4 (April 2025), 2245–2264.
[7] Gianluca Bontempi et al. 2013. Machine Learning Strategies for Time Series Forecasting. Springer, 62–77.
[8] George E. P. Box et al. 1990. *Time Series Analysis, Forecasting and Control.* Holden-Day, Inc.
[9] Sergio Caccamo et al. 2015. Extending a UGV teleoperation FLC interface with wireless network connectivity information. In *2015*

IEEE/RSJ International Conference on Intelligent Robots and Systems.

[10] Ilias Chatzistefanidis et al. 2023. ML-based Traffic Steering for Heterogeneous Ultra-dense beyond-5G Networks. In *2023 IEEE Wireless Communications and Networking Conference*. 1–6.

[11] Thusitha Dayaratne et al. 2025. Robust Anomaly Detection in O-RAN: Leveraging LLMs against Data Manipulation Attacks. In *Proceedings of the free5GC World Forum*.

[12] Jia Deng et al. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 248–255.

[13] Jacob Devlin et al. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805* (2018).

[14] Habiba Elsherbiny et al. 2020. 4G LTE User Equipment Measurements along Kingston Transit 502 Bus Route. doi:10.5683/SP2/EQWKO1

[15] Luca-Andrei Fechete et al. 2025. Goal-Oriented Time-Series Forecasting: Foundation Framework Design. *arXiv:2504.17493* (2025).

[16] Austin Feng et al. 2025. TelecomTS: A Multi-Modal Observability Dataset for Time Series and Language Analysis. arXiv:2510.06063

[17] Pablo Fondo-Ferreiro et al. 2023. *Network operator KPIs time series dataset*. https://doi.org/10.5281/zenodo.8147768

[18] Leo Gao et al. 2021. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. (2021). arXiv:2101.00027

[19] Moinak Ghoshal et al. 2025. A First Large-Scale Study of Operational 5G Standalone Networks. *Proc. ACM Netw.* 3, CoNEXT4 (2025).

[20] Emanuele Goldoni et al. 2018. Experimental data set analysis of RSSI-based indoor and outdoor localization in LoRa networks. *Internet Technology Letters* 2, 1 (2018).

[21] Mononito Goswami et al. 2023. AQua: A Benchmarking Tool for Label Quality Assessment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[22] Mononito Goswami et al. 2024. MOMENT: a family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*.

[23] Joshua Groen et al. 2024. TRACTOR: Traffic Analysis and Classification Tool for Open RAN. In *IEEE International Conference on Communications*. 4894–4899.

[24] Craig Gutterman et al. 2019. RAN Resource Usage Prediction for a 5G Slice Broker. In *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*. 231–240.

[25] Edward J. Hu et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[26] Yuhong Huang et al. 2024. Communication and Computing Integrated RAN: A New Paradigm Shift for Mobile Network. *IEEE Network* (2024).

[27] Muhammad Kabeer et al. 2025. An Urban Multi-Operator QoE-Aware Dataset for Cellular Networks in Dense Environments. Mendeley Data, V1. doi:10.17632/dx5xyyfz2y.1

[28] Paulo Valente Klaine et al. 2017. A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2392–2431.

[29] Linchao Li, Jian Zhang, Yonggang Wang, and Bin Ran. 2019. Missing Value Imputation for Traffic-Related Time Series Data Based on a Multi-View Learning Method. *IEEE Transactions on Intelligent Transportation Systems* 20, 8 (2019), 2933–2943.

[30] Wenxiang Li et al. 2024. Deep Learning Models for Time Series Forecasting: A Review. *IEEE Access* 12 (2024), 92306–92327.

[31] Akrit Mudvari et al. 2021. ML-driven scaling of 5G Cloud-Native RANs. In *2021 IEEE Global Communications Conference*. 1–6.

[32] Navid Nikaein et al. 2014. OpenAirInterface: A Flexible Platform for 5G Research. *SIGCOMM Comput. Commun. Rev.* 44, 5 (Oct. 2014), 33–38.

[33] Ioannis Panitsas et al. 2024. Predictive Handover Strategy in 6G and Beyond: A Deep and Transfer Learning Approach. (2024). arXiv:2404.08113

[34] Ioannis Panitsas et al. 2025. FedJam: Multimodal Federated Learning Framework for Jamming Detection. (2025). arXiv:2508.09369

[35] Ioannis Panitsas et al. 2025. JamShield: A Machine Learning Detection System for Over-the-Air Jamming Attacks. In *IEEE International Conference on Communications*. 1067–1072.

[36] Ioannis Panitsas et al. 2025. SlicePilot: Demystifying Network Slice Placement in Heterogeneous Cloud Infrastructures. (2025). arXiv:2509.18545

[37] Karl Pearson. 1901. On lines and planes of closest fit to systems of points in space. *Philos. Mag.* 2, 11 (1901), 559–572.

[38] Michele Polese et al. 2023. ColO-RAN: Developing Machine Learning-Based xApps for Open RAN Closed-Loop Control on Programmable Experimental Platforms. *IEEE Transactions on Mobile Computing* 22, 10 (2023), 5787–5800.

[39] Michele Polese et al. 2023. Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1376–1411.

[40] Darijo Raca et al. 2020. Beyond throughput, the next generation: a 5G dataset with channel and context metrics. In *Proceedings of the 11th ACM Multimedia Systems Conference*.

[41] Colin Raffel et al. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[42] Gautham Reddy et al. 2025. WIOC: Wireless Indoor-Outdoor Classification Using WiFi and Cellular Signals. In *2025 IEEE International Conference on Communications Workshops*. 1954–1959.

[43] Simran Singh. 2024. *Android-based 4G LTE, 5G NR, and Throughput Measurements for Two Sweeps of a UAV near a Private AERPAW Base Station*. https://aerpaw.org/dataset/dataset-23-android-based-4g-lte-5g-nr-and-throughput-measurements-for-two-sweeps-of-a-uav-near-a-private-aerpaw-base-station/

[44] Simran Singh. 2024. *Android-based Public 4G LTE Measurements from a Tethered Helikite*. https://aerpaw.org/dataset/dataset-25-android-based-public-4g-lte-measurements-from-a-tethered-helikite/

[45] Simran Singh. 2024. *Detailed Cellular and Throughput Measurements for Horizontal Sweeps of a UAV across a Private AERPAW Base Station, using Keysight Nemo and PawPrints*. https://aerpaw.org/dataset/dataset-24-android-based-4g-lte-measurements-for-semi-circular-uav-trajectory-around-a-private-aerpaw-base-station/

[46] Chuanhao Sun et al. 2024. SpotLight: Accurate, Explainable and Efficient Anomaly Detection for Open RAN. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*.

[47] Ashish Vaswani et al. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30.

[48] Chao-Kai Wen et al. 2018. Deep Learning for Massive MIMO CSI Feedback. *IEEE Wireless Communications Letters* 7, 5 (2018), 748–751.

[49] Duo Wu et al. 2024. NetLLM: Adapting Large Language Models for Networking. In *Proceedings of the ACM SIGCOMM 2024 Conference*.

[50] Haixu Wu et al. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*.

[51] Haixu Wu et al. 2023. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *International Conference on Learning Representations*.

[52] Bruno Missi Xavier et al. 2024. Performance measurement dataset for open RAN with user mobility and security threats. *Computer Networks* 253 (2024).

[53] Ce Zhou et al. 2025. A Comprehensive Survey on Pretrained Foundation Models: A History from BERT to ChatGPT. *International Journal of Machine Learning and Cybernetics* 16, 12 (2025).

[54] Haoyi Zhou et al. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the AAAI*.