
Preprint:
Liu, Z., Liu, B., Valcarce, A. et al. (Submitted: 2025) LLM-based emulation of the radio resource control layer: towards AI-native RAN protocols. [Preprint - arXiv] (Submitted)

https://doi.org/10.48550/arXiv.2505.16821

LLM-Based Emulation of the Radio Resource Control Layer: Towards AI-Native RAN Protocols

Ziming Liu, *Member, IEEE*, Bryan Liu, *Member, IEEE*, Alvaro Valcarce, *Senior Member, IEEE*, and Xiaoli Chu, *Senior Member, IEEE*

***Abstract*—Integrating Large AI Models (LAMs) into 6G mobile networks promises to redefine protocol design and control-plane intelligence by enabling autonomous, cognitive network operations. While industry concepts, like ETSI's Experiential Networked Intelligence (ENI), envision LAM-driven agents for adaptive network slicing and intent-based management, practical implementations still face challenges in protocol literacy and real-world deployment. The current paper reports a novel end-to-end demonstration of a LAM that generates *standards-compliant, ASN.1-encoded* Radio Resource Control (RRC) messages as part of control-plane procedures inside a gNB. We treat RRC messaging as a domain-specific language and fine-tune a decoder-only transformer model (LLaMA-class) using parameter-efficient Low-Rank Adaptation (LoRA) on RRC messages linearized to retain their ASN.1 syntactic structure before standard Byte Pair Encoding (BPE) tokenization. This enables a combinatorial generalization over RRC protocol states while minimizing training overhead. On 30k field-test RRC request–response pairs, our 8B model achieves a median cosine similarity of 0.97 with ground-truth messages on an edge GPU — a 61% relative gain in similarity over a zero-shot LLaMA 3 8B baseline, indicating substantially improved structural and semantic RRC fidelity. Overall, our experiment results show that LAMs, when augmented with Radio Access Network (RAN)-specific reasoning, can directly orchestrate control-plane procedures—a stepping stone toward the AI-Native Air Interface paradigm. Beyond RRC emulation, this work lays the groundwork for broader LAM applications in radio protocol learning.**

***Index Terms*—6G, Radio Resource Control, protocol learning, AI-Native Air Interface, Large AI Model**

I. Introduction

Large AI Models (LAMs) are poised to transform wireless network design by enabling autonomous, cognitive capabilities that go beyond the scope of traditional optimization methods. The forthcoming sixth generation (6G) of mobile networks is expected to incorporate Artificial Intelligence (AI) methods not merely as auxiliary optimisation tools but also as integral design primitives of the AI-Native Air Interface (AI-AI). Recent proposals for an AI-AI posit that Machine Learning (ML) models could co-design Physical Layer (PHY) - and Medium Access Control (MAC) procedures [1], thereby enabling radio stacks that adapt automatically to dynamic service requirements, resource constraints, and propagation conditions [2]. The rapid evolution of LAMs [3], [4] strengthens this vision—when endowed with multi-modal perception and reasoning, such models become plausible agents for protocol-centric tasks that demand both linguistic competence and structured decision making. This raises a timely research challenge: determining the extent to which LAMs can directly engage in the generation and interpretation of control signalling in Radio Access Network (RAN). Can LAMs achieve sufficient protocol literacy for RRC emulation? What fine-tuning strategies are effective? What is the performance fidelity of such a system? These are the research questions this paper intends to answer.

One particularly compelling target for AI intervention is the Radio Resource Control (RRC) layer of 5G New Radio (NR). RRC is the Layer 3 (L3) of the cellular network. It orchestrates connection establishment, mobility and configuration between User Equipment (UE) and the Next-Generation Node B (gNB). Conceptually, RRC messages form a specialised, domain-specific language; learning this language is analogous to learning syntax and semantics in natural-language processing. Treating RRC message as such positions LAM-based sequence models as natural emulators of the RRC layer. Demonstrating accurate RRC emulation would constitute an early, concrete instantiation of AI-AI principles, with direct implications for the design of the 6G control plane.

Industry roadmaps reinforce this trend. The recent ETSI Experiential Networked Intelligence (ENI)-051 report introduced an Multi-Agent Core Architecture (AI-Core), where multiple heterogeneous agents—each supported by a Domain-Specific Large AI Model (AgentGPT) and a larger, shared knowledge model (i.e., the Telecom Knowledge Large AI Model (NetGPT))—jointly manage next-generation network slicing [5]. Within this framework, an *Actor* component translates high-level agent decisions into concrete protocol messages. Emulating RRC via LAMs directly aligns with the Actor's mission and thus provides an empirical testbed for the ENI vision, bridging theoretical architecture with practical implementation.

The scope of LAM-enabled control extends beyond RRC emulation [6]. A closely related application is Root Cause Analysis (RCA) of RRC traces and broader RAN telemetry, where a model fine-tuned on historical logs could infer latent failure modes and recommend remedial actions. Such actions are still manually done today. Additional opportunities include LAM-driven anomaly detection, adaptive L3 resource control, and intent-based radio configuration. Although these topics lie outside the scope of the present study, they illustrate the

Z. Liu and X. Chu are with the School of Electrical and Electronic Engineering, The University of Sheffield, Sheffield S10 2TN, U.K. (e-mail: {ziming.liu,x.chu}@sheffield.ac.uk).

B. Liu and Á. Valcarce are with Nokia Bell Labs, 12 Rue Jean Bart, 91300 Massy, France (e-mail: {bryan.liu,alvaro.valcarce_rial}@nokia-bell-labs.com).

breadth of potential once protocol literacy is achieved by a Large Language Model (LLM).

Can the protocol literacy of a LAM—specifically, its ability to parse and generate 5G RRC messages—be systematically improved, and through which fine-tuning strategies? This paper answers this question through a novel two-phase methodology that (i) imparts RRC knowledge to a decoder-only LAM via supervised fine-tuning enhanced with Low-Rank Adaptation (LoRA), and (ii) evaluates the resulting model's fidelity in emulating RRC procedures under realistic network scenarios. The findings offer evidence for the feasibility of LAM-based RRC and identify the conditions under which such models may contribute towards the AI-AI in forthcoming 6G systems.

Without loss of generality, this work focuses on *decoder-only* transformer architectures (e.g., LLaMA-class models [7]). Autoregressive decoding matches the turn-taking structure of RRC exchanges and permits key–value caching, which curtails inference latency and memory footprint, both critical for deployment at the gNB. Decoder-only models thereby strike a favourable balance between capacity and computational efficiency, especially when combined with parameter-efficient fine-tuning techniques such as LoRA [8]. LoRA freezes the original model weights and introduces low-rank update matrices, reducing trainable parameters by orders of magnitude while maintaining expressiveness; the approach mitigates both the cost of domain adaptation and the runtime overhead of inference, two primary barriers to operational use.

Training LAMs for specialist telecom domains demands curated datasets that capture the combinatorial breadth of protocol states, while deployment must meet stringent latency budgets to avoid impairing control-plane responsiveness. Addressing these issues entails advances in data-efficient learning, model compression, and hardware-accelerated execution—topics that are treated in subsequent sections.

II. System Model

This section delineates the architectural and operational principles underlying the proposed LAM-integrated gNB. We first describe the system architecture, detailing the functional decomposition of the disaggregated gNB and the embedding of a LAM within the RRC layer, while ensuring adherence to standardized interface definitions. Then, the model-driven control-plane workflow is examined, encompassing the reception and contextual interpretation of uplink RRC messages and the autoregressive synthesis of standards-compliant downlink responses.

A. Architecture and Functional Integration

The proposed Large RRC Model integrates a decoder-only transformer-based LAM into the RRC layer of a disaggregated gNB, as depicted in Fig. 1. Following the 3rd Generation Partnership Project (3GPP) NG-RAN standard, the gNB is partitioned into a Central Unit (CU) and a Distributed Unit (DU), with the CU further subdivided into separate control-plane Control Plane of Central Unit (CU-CP) and user-plane User Plane of Central Unit (CU-UP) entities. The CU-CP hosts the RRC layer, responsible for essential control-plane tasks such as connection establishment, radio bearer management, and mobility decisions. Replacing the traditional rule-based logic, the embedded RRC LLM serves as an intelligent decision engine within the RRC, producing control messages and associated protocol decisions via autoregressive inference. Conversely, the CU-UP encompasses user-plane layers, primarily Service Data Adaptation Protocol (SDAP) and user-plane Packet Data Convergence Protocol (PDCP), that manage the transport of user data. The DU, hosting lower-layer (e.g., Radio Link Control (RLC), MAC, and PHY layers) protocols, interfaces directly with the Remote Radio Heads (RRHs) performing Radio Frequency (RF) and low-level physical-layer tasks. Fig. 2 illustrates all these interfaces and functions. It's important to note that integrating a LLM at the RRC level is intended to preserve the external protocol behavior, leaving northbound and lateral interfaces to peer nodes unaltered, thus ensuring interoperability with existing network deployments.

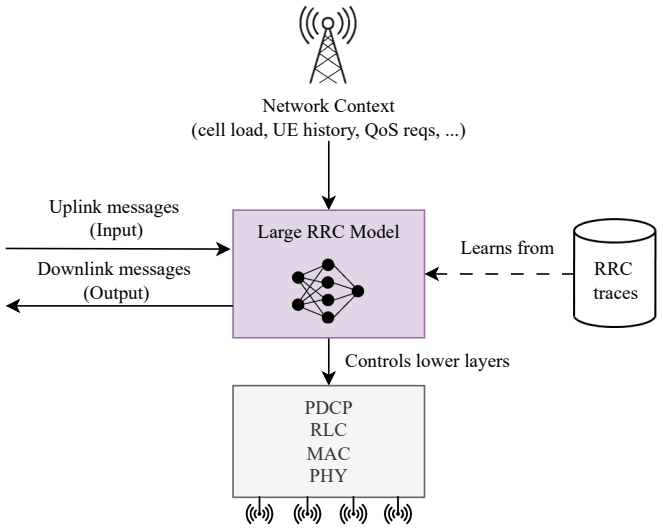

Fig. 1. High-level concept of an AI-native gNB-side RRC layer powered by a LLM, illustrating its core inputs, outputs, learning source, and interaction with the protocol stack.

As in standard 5G New Radio (NR), uplink RRC messages from UEs, such as Connection Requests or Measurement Reports, arrive at the CU-CP via Control-Plane Interface between CU-CP and DU (F1-C), where they are processed by the embedded RRC LLM, replacing the conventional RRC layer implementation. The model interprets these inputs contextually, considering ongoing connection states, historical interactions, and dynamic network conditions to generate suitable downlink RRC responses (e.g., Connection Setup or Mobility Reconfiguration). The generated messages maintain compliance with standardized 3GPP encoding, thus transparently replacing the message generation logic. Downlink responses are then transmitted back to the UE through the DU via the same interface.

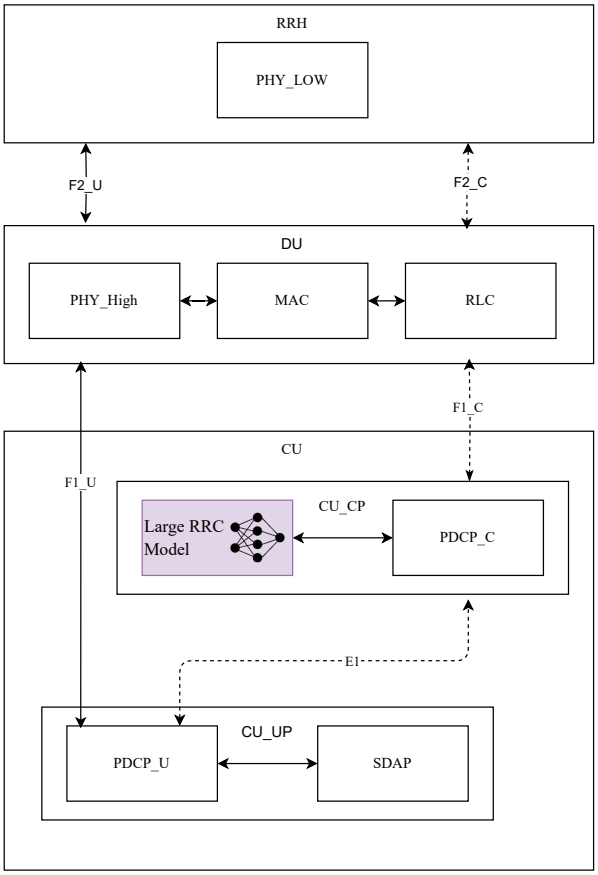

Fig. 2. Low-level architecture of an NR gNB disaggregating an LLM-based RRC layer.

B. LLM-Driven Signaling Workflow and Model Operation

Interactions between the RRC layer and the user-plane modules are similarly orchestrated. When the Large RRC Model decides to initiate or modify data radio bearers—potentially based on inferred context or changing service demands—it signals these decisions to the CU-UP over the Interface between CU-CP and CU-UP (E1). The CU-UP subsequently configures the corresponding SDAP and PDCP entities, enabling adaptive management of the user-plane traffic. Meanwhile, data-plane traffic flows remain unmodified and proceed independently over the established user-plane interfaces (User-Plane Interface between CU-UP and DU (F1-U), User-Plane Interface between RRH and DU (F2-U)).

Fig. 3 illustrates the fine-tuning and inference workflow specific to the proposed RRC LLM. The decoder-only transformer is taught RRC domain knowledge using supervised fine-tuning on a curated dataset of historical RRC message traces collected from a real-world 5G network. This specialized fine-tuning imparts explicit protocol knowledge to the model, enabling it to emulate standard RRC behaviors accurately during live inference. Operationally, the trained model functions continuously within the CU-CP, dynamically generating control-plane messages as new UE requests are received.

III. Fine-tuning of LLM for RRC Message Processing

Our approach consists of *dataset construction*, *LoRA-based supervised fine-tuning*, and *latency-aware evaluation*. First, raw 5G-NR traces are filtered and reorganised into strict one-request–one-response question–answer pairs (Section III-A1). Second, a LLaMA3-8B backbone is adapted with low-rank adapters and trained on this corpus to internalise RRC syntax and semantics (Section III-B1). Finally, the resulting RRC-LLM is benchmarked in an end-to-end stand-alone (SA) testbed, where we measure BERT-based cosine similarity against ground-truth replies and on-device response latency (Section IV).

A. Dataset Construction

1) Data Collection and Pre-processing: The data used in this work originates from field test logs of 5G New Radio (5GNR) RRC sessions. As these traces include non-RRC messages, such as those related to the Access and Mobility Management Function (AMF)—they require preprocessing to isolate the relevant RRC-specific exchanges. To enable effective use in LLM training, the data is cleaned and reorganized to exclude non-RRC entries and meta-data. Fig. 4 illustrates a portion of a processed RRC session, showing only the RRC-specific messages after the removal of unrelated protocol content.

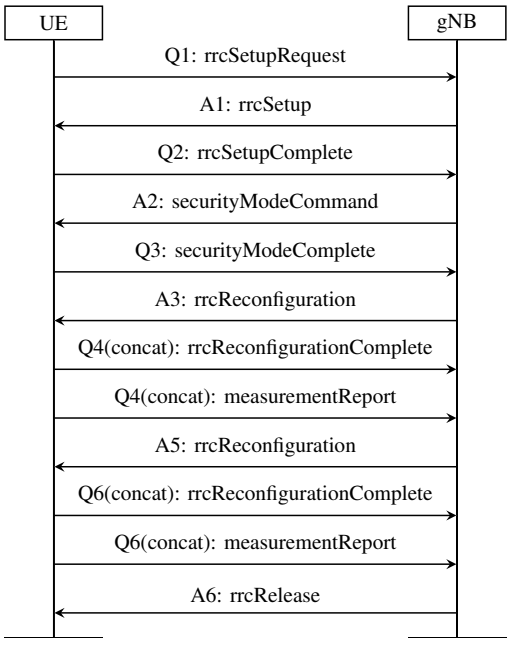

Fig. 4. Example of a pre-processed RRC trace segment. Messages are structured into Question (Q) / Answer (A) pairs for LLM training, where 'Q' denotes the input RRC message(s) (X) and 'A' the target response (Y). Note the consolidation of multiple physical messages (e.g., *rrcReconfigurationComplete* and *measurementReport*) into a single logical request, labeled here as Q4(concat) and Q6(concat).

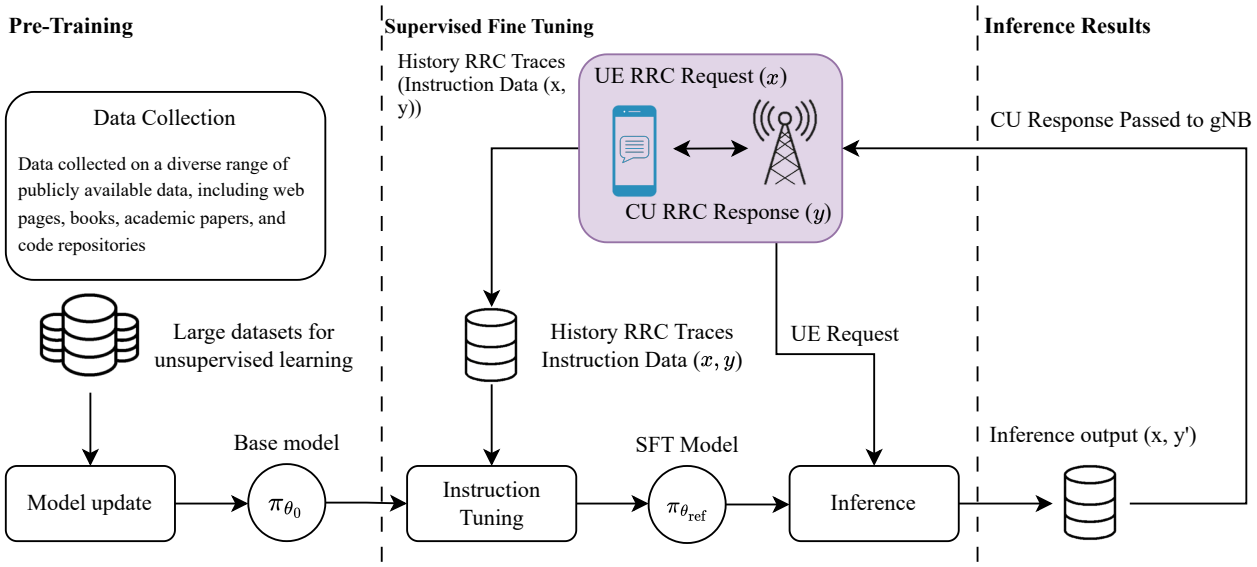

Fig. 3. Data flow for fine-tuning and inference in the LLM-based RRC system.

The RRC-LLM module in Fig. 2 handles tasks related to RRC signalling and connection management. While the current implementation primarily focuses on RRC interactions, additional messages from the core network can be appended as context to the LLM inputs. Historical RRC exchanges can also be appended to enrich the model's contextual understanding.

To facilitate instruction fine-tuning, a strictly one-question-one-answer labeled dataset is required, where each "question" corresponds to an uplink RRC message input, and each "answer" corresponds to the downlink RRC message output. In the original traces, request-reply pairs are frequently interleaved due to chronological logging, and multiple RRC procedures may overlap—a common occurrence in real network traces. To address this issue, UE-specific signalling exchanges are extracted by isolating one RRC procedure at a time. By filtering out AMF-related messages and consolidating only the RRC-relevant information, the dataset is reorganized to meet the one-question-one-answer standard. Specifically, all RRC request messages within a session are classified as a consolidated "request," and all RRC response messages are classified as a consolidated "response."

B. Fine-Tuning Procedure

A pre-trained LLM is fine-tuned using supervised learning on the processed dataset. This allows the model to learn RRC-specific message structures, improving its ability to generate compliant responses. The objective is to make the model's responses closely match the ground truth annotations in terms of structure and content, thereby enhancing its reliability in RRC message generation tasks.

1) Training and Inference Procedure: The system implements a structured, LLM-based training and inference process for handling RRC messages in a 5G network, as outlined in Fig. 3. It begins with an *Input Sentence* derived from a linearized RRC message, preserving essential syntactic and semantic relationships to streamline protocol message processing.

During the *Forward Pass*, Byte Pair Encoding (BPE) tokenizes the input into variable-length sequences of subword tokens [9]. These sequences are then padded to a standardized length before being processed by the fine-tuned model $\pi_{\theta_{\text{ref}}}$. Low-Rank Adaptation (LoRA) weights supplement base weights via low-rank matrices, enhancing model adaptability. The cross-entropy loss is computed between the predicted and ground truth token distributions, quantifying the discrepancy at each timestep and guiding the model to better capture RRC-specific semantic patterns.

In the *Backward Pass*, gradients for LoRA parameters are computed via backpropagation, prompting weight updates that minimize prediction errors [10]. The Adam optimizer is applied to these LoRA-specific parameters, iterating to align the model's output with the expected RRC responses.

Finally, the *Output Sentence* represents the generated downlink response for the given uplink RRC message.

TABLE I
TRAINING CONFIGURATION

Parameter	Value
Model	LLaMA3-8B
Max Sequence Length	4096
Training Precision	bf16
Device	CUDA
Batch Size	2
Epochs	18
Gradient Accumulation Steps	1
Optimizer	AdamW
Learning Rate	2e-5
Loss Function	CrossEntropyLoss

Table I summarizes the key hyperparameters and configura-

tions used during the supervised fine-tuning of the LLaMA3-8B model [11]. These settings ensure that the model is trained under optimal resource constraints while preserving fidelity to the original RRC message structures.

2) Tokenization and Sequence Generation from RRC Messages: At present, each complete RRC message (whether a request or a response) is composed of multiple fields containing specific parameters and information elements as defined by 5G NR standards. These fields are systematically extracted and linearized into a flat text representation while preserving their hierarchical and semantic relationships. The linearized text forms a "Sentence", defined here as a single, coherent textual representation that encapsulates the entire meaning of the RRC message.

This Sentence is tokenized using the same Byte Pair Encoding (BPE) tokenizer employed during the pre-training of the LLaMA3-8B model, ensuring consistency in token representation. In this context, a "sequence" refers to the ordered set of BPE-derived tokens. This methodology ensures that structural and contextual information within the RRC message fields is preserved, enabling the language model to learn the intricate dependencies among different fields.

Key Definitions:

- **Input** (X): The input Sentence represented as a sequence of tokens
$$X = \{x_1, x_2, \ldots, x_S\}, \tag{1}$$
obtained using the BPE tokenizer.
- **Label** (Y): The target Sentence corresponding to X, represented as
$$Y = \{y_1, y_2, \ldots, y_T\}, \tag{2}$$
also tokenized by BPE.
- **Inference Output** (Y'): The Sentence generated by the model during inference,
$$Y' = \{y'_1, y'_2, \ldots, y'_{T'}\}, \tag{3}$$
obtained by decoding the output tokens.

Here, S, T, and T' denote the number of tokens in the input sequence X, in the target sequence Y, and in the model-generated output Y', respectively.

Example of BPE Tokenization: Consider an excerpt from an RRC message, expressed in human-readable ASN.1 encoding. The hierarchical structure of the message is first converted into a linear text Sentence. Each word or parameter is then split into subword units via BPE, merging frequently co-occurring characters to form more compact tokens.

As shown in Table II, each token corresponds to a meaningful subword (or word) from the original RRC message fields. These tokens maintain critical semantic relationships and are suitable for processing by the LLM. By defining the sequence in this manner, both the syntactic and semantic content of RRC messages are captured in a machine-learning-friendly format.

Integration with LLaMA: To efficiently process text, we employ LLaMA's built-in BPE tokenizer. This approach:

- Converts RRC Sentences into sequences of token IDs via BPE.
- Maps tokens $\{x_i\}$ to unique integer IDs $\{\mathrm{id}(x_i)\}$.

TABLE II
EXAMPLE OF TOKEN AND TOKEN ID LOOKUP TABLE

Token	Token ID	Token	Token ID
`me`	1 047	`Serving`	110 058
`as`	288	`Cell`	5 346
`Results`	12 928	`serv`	2 096
`meas`	9 802	`0`	15
`Id`	906	`rs`	15 181
`␣`	220	`rp`	49 866
`1`	16	`50`	1 434
`Result`	2 769	`rq`	116 596
`10`	702	`-`	533

- Feeds these token IDs as inputs to the LLaMA model for training or inference.

It is worth noting that, consistent with standard BPE practice, this tokenizer represents numerical fields as sequences of digits or sub-word tokens, rather than treating them as atomic numerical entities. This direct usage of LLaMA's tokenizer ensures compatibility with existing model infrastructure and preserves crucial domain-specific details within the RRC messages. Consequently, the LLM can learn from, and eventually generate, properly tokenized protocol messages, thereby aiding enhanced RRC in telecommunication systems.

3) QA-Modeling of RRC Procedures: This models RRC request-response exchanges as question-answering (QA) tasks. To illustrate this, consider the following conditional probability distribution between an RRC request (X) and its corresponding response (Y):

$$P(Y \mid X; \theta) \;=\; \prod_{t=1}^{T} P\big(y_t \;\big|\; y_{<t}, X; \theta\big), \tag{4}$$

where θ is the set of all trainable model parameters, y_t denotes the token generated at time step t, and $y_{<t} = \{y_1, \ldots, y_{t-1}\}$ denotes the tokens generated before time step t. By explicitly modeling $P(Y \mid X; \theta)$, the model incorporates the input context X throughout the generation process, leading to outputs that remain closely aligned with the given RRC request.

In our supervised fine-tuning (SFT) framework, the SFT model $\pi_{\theta_{\text{ref}}}$ is trained to estimate this conditional probability distribution, capturing the relationships in the labeled dataset. Specifically, during training, the model is presented with input sequences X (i.e., uplink RRC requests) alongside their corresponding target sequences Y (i.e., downlink responses). By minimizing the negative log-likelihood of $\{(X, Y)\}$ pairs, the SFT model $\pi_{\theta_{\text{ref}}}$ learns how to generate responses Y' conditional on inputs X, effectively mastering the QA-style mappings inherent in RRC request-response protocols.

C. Model and Training Formulation

a) Dataset definition: Let the raw trace corpus be $\mathcal{T}_{\text{raw}} = \{\tau^{(j)}\}_{j=1}^{M}$, where each $\tau^{(j)}$ is a complete base-station log and $M = |\mathcal{T}_{\text{raw}}|$ is therefore the total number of such logs. A deterministic preprocessing function $\mathcal{F} : \mathcal{T}_{\text{raw}} \to \{(X, Y)\}$—performing *filtering*, *de-duplication*, and *chronological re-ordering*—yields the supervised training set $\mathcal{D} = \{(X^{(i)}, Y^{(i)})\}_{i=1}^{N}$, where $N = |\mathcal{D}|$ denotes the number of

aligned uplink request–downlink response pairs obtained after preprocessing and $X^{(i)}, Y^{(i)}$ are defined in Sec. III-B2.

b) Supervised fine-tuning: A copy of the foundation model is initialised as $\pi_{\theta_{\text{ref}}} = \text{LoRA}(W_0; A, B)$, with the resulting low-rank update given by $W_\theta = W_0 + AB$, where:

- $W_0 \in \mathbb{R}^{d\times k}$ is the *frozen* weight matrix from the base model (e.g., a query, key, or value projection);
- $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$ are the *trainable* low-rank factors;
- d is the input (row) dimension, typically equal to the model's hidden size;
- k is the output (column) dimension of the original weight;
- $r \ll \min\{d, k\}$ is the chosen rank of the LoRA update.

The objective of SFT is to maximize the conditional likelihood:

$$\mathcal{L}_{\text{SFT}}(\theta_{\text{ref}}) = -\frac{1}{N}\sum_{i=1}^{N} \log P\big(Y^{(i)} \mid X^{(i)}; \theta_{\text{ref}}\big) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T^{(i)}} \text{CE}\big(y_t^{(i)}, \pi_{\theta_{\text{ref}}}(y_t \mid y_{<t}, X^{(i)})\big), \quad (5)$$

where $\text{CE}\big(y_t^{(i)}, \pi_{\theta_{\text{ref}}}(y_t \mid y_{<t}, X^{(i)})\big)$ is the cross-entropy loss,and $T^{(i)}$ is the length (in BPE tokens, including the explicit end-of-sequence marker) of the i-th target sequence $Y^{(i)}$.

c) Inference.: Let X be an unseen uplink RRC message and let π_θ denote the fine-tuned model whose parameters θ were obtained after supervised fine-tuning. The model generates a downlink response by greedy decoding,

$$\hat{Y} = \arg\max_{Y} \pi_\theta\big(Y \mid X\big), \quad (6)$$

and transmits $\hat{Y}$ as the downlink RRC message.

Algorithm 1 RRC-LLM Training and Inference Pipeline

Require: Raw RRC traces $\mathcal{T}_{\text{raw}}$, pre-trained model $\pi_{\theta_{\text{init}}}$
Ensure: Fine-tuned model π_{θ^*}

Phase 1: Dataset Construction
$\mathcal{D} \leftarrow$ `Preprocess`$(\mathcal{T}_{\text{raw}})$ {Filter, align, and extract (X, Y) pairs}
for all $(X^{(i)}, Y^{(i)}) \in \mathcal{D}$ **do**
  $X^{(i)} \leftarrow$ `BPE_Tokenize`$(X^{(i)})$
  $Y^{(i)} \leftarrow$ `BPE_Tokenize`$(Y^{(i)})$
end for
Phase 2: Supervised Fine-Tuning (SFT)
$\pi_{\theta_{\text{ref}}} \leftarrow$ `LoRA_Init`$(\pi_{\theta_{\text{init}}})$
for $e = 1$ to E_{SFT} **do**
  for all $(X^{(i)}, Y^{(i)}) \in \mathcal{D}$ **do**
    $\mathcal{L}_{\text{SFT}} \leftarrow -\log \pi_{\theta_{\text{ref}}}(Y^{(i)} \mid X^{(i)})$
    Update θ_{ref} using Adam on $\mathcal{L}_{\text{SFT}}$
  end for
end for
Phase 3: Inference
Given new uplink message X, compute
  $\hat{Y} \leftarrow \arg\max_Y \pi_\theta(Y \mid X)$
return $\hat{Y}$

D. Training Configuration and Convergence Analysis

During the supervised fine-tuning of the LLaMA3-8B model, we observed that the training loss rapidly decreased within the first few hundred steps, reaching a relatively stable range by step 200.

Fig. 5 shows the log–log training-loss curve. Within the first 1000 steps, the EMA-smoothed loss plunges from $\sim$1.0 to the 10^{-2}–10^{-3} range, while the raw trace—though noisy—exhibits diminishing spikes, indicating increasing model stability. After step 8 400, improvements level off; therefore, the checkpoint marked by the vertical red dashed line is chosen as a compromise between rapid convergence and the risk of overfitting.

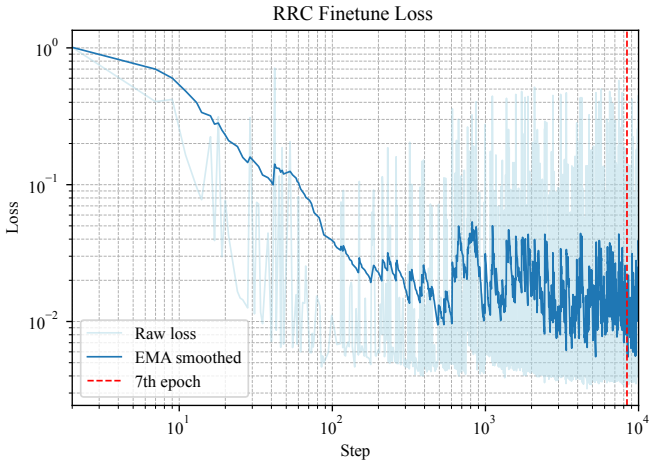

Fig. 5. Log–log plot of the per-step training loss. The pale blue trace is the raw loss at each optimizer step, and the dark blue curve is an exponential moving average (EMA) that smooths out high-frequency fluctuations. The vertical red dashed line marks the epoch that is selected for downstream evaluation.

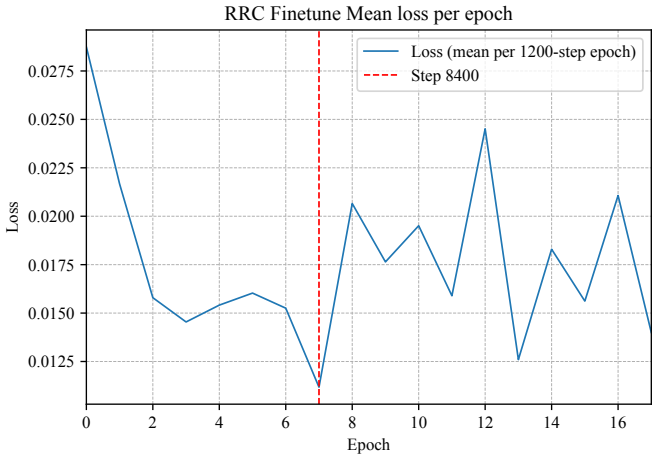

Fig. 6. Mean training loss for each 1200-step epoch. Loss falls rapidly in the first few epochs, reaches its minimum just before the chosen epoch (red dashed line), and thereafter fluctuates around a low, stable plateau as the model nears convergence.

Given the scale of the LLaMA3-8B backbone and the high memory footprint of long-sequence RRC samples, we adopt a

pragmatic checkpointing approach that balances training efficiency with generalisation. Instead of running full validation passes in parallel—an option that would roughly double GPU hours and exhaust the testbed's memory budget—we monitor the exponential-moving average of the training loss and freeze the model once that curve stabilises (Fig. 5). Generalisation is then evaluated offline on a disjoint trace suite using the Sentence-similarity framework of Section IV. This two-stage procedure keeps computational overheads manageable while still providing a reliable, out-of-sample measure of model quality.

Fig. 6 further supports this choice: the average loss per epoch decreases from approximately 0.035 in epoch 1 to around 0.012 just before the selected epoch. Beyond this point, additional fine-tuning yields only marginal improvements, with the loss fluctuating between 0.015 and 0.022. This plateau indicates that the model's parameters have largely stabilized, making the selected epoch an effective trade-off between training time and performance on the RRC task.

The "loss–step" and "loss–epoch" views together indicate three distinct training phases. Phase I (steps 0–1 k, $\approx$ 0.1 epoch) shows a two-order-of-magnitude drop—from ≈ 1 to 10^{-2}–10^{-3}% demonstrating how quickly a LoRA-adapted LLaMA-3 internalises the RRC domain once exposed to only a handful of mini-batches. Phase II (steps 1 k–8.4 k, epochs 1–7) is characterised by a gentler slope plus gradually shrinking step-to-step variance; spikes visible in the raw trace stem from the tiny effective batch size (2×4096 tokens, no gradient accumulation) but are damped in the EMA curve. Phase III ($\geq$ 8.4 k steps, epochs $\geq$ 7) plateaus around a mean cross-entropy of 0.012; further optimisation oscillates within 0.015–0.022 without consistent improvement.

Selecting the red-line checkpoint therefore (i) captures 95 %+ of the attainable loss reduction, (ii) prevents wasteful computation beyond the point of diminishing returns, and (iii) limits the risk of over-fitting that can arise when fine-tuning with low-rank adapters on a narrow protocol corpus. The remainder of this paper reports results using this 7-epoch model.

IV. Performance and Evaluation

A. Protocol-fidelity Metrics

We compare sentence similarity using BERT embeddings and cosine similarity to assess model performance [12]. The LLM-generated responses are compared with actual base station replies, ensuring protocol response accuracy.

a) BERT-based Similarity Computation: To evaluate model performance, we apply sentence similarity analysis based on BERT embeddings and cosine similarity. The responses generated by the LLM are compared against actual base station replies to ensure compliance with protocol specifications and response accuracy. A high-level overview of this sentence similarity approach is presented in Figure 7.

Formally, for each sample i in the dataset:

- **Target Sentence** $Y^{(i)}$: The reference sentence provided in the dataset.

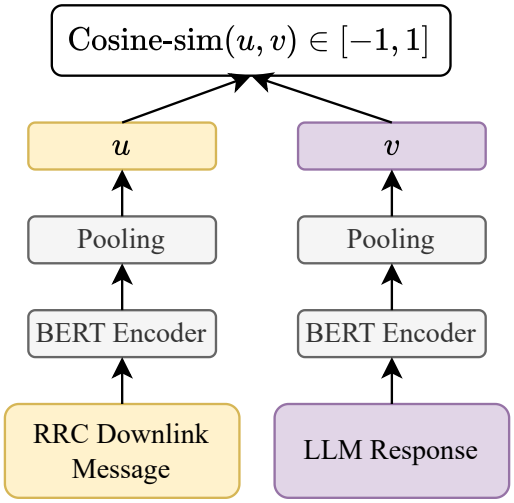

Fig. 7. Illustration of using BERT to encode ground truth RRC downlink messages and LLM-generated responses, followed by cosine similarity calculation for performance evaluation.

- **Generated Sentence** $Y'^{(i)}$: The output produced by the fine-tuned model $\pi_{\theta_{\text{ref}}}$ given input $X^{(i)}$.

We use a pre-trained BERT model BERT_ϕ with fixed parameters ϕ to obtain contextual embeddings for each token in the sentences. Let

$$\begin{aligned} E^{(i)} &= \text{BERT}_\phi\big(Y^{(i)}\big) = \big[e_1^{(i)}, e_2^{(i)}, \ldots, e_{T^{(i)}}^{(i)}\big], \\ E'^{(i)} &= \text{BERT}_\phi\big(Y'^{(i)}\big) = \big[e_1'^{(i)}, e_2'^{(i)}, \ldots, e_{T'^{(i)}}'^{(i)}\big]. \end{aligned} \tag{7}$$

To obtain fixed-size sentence embeddings, we apply a simple pooling operation:

$$\begin{aligned} u^{(i)} &= \frac{1}{T^{(i)}} \sum_{t=1}^{T^{(i)}} e_t^{(i)}, \\ v^{(i)} &= \frac{1}{T'^{(i)}} \sum_{t=1}^{T'^{(i)}} e_t'^{(i)}. \end{aligned} \tag{8}$$

The cosine similarity between the pooled sentence embeddings $u^{(i)}$ and $v^{(i)}$ is then calculated as:

$$Z^{(i)} = \cos(\theta^{(i)}) = \frac{u^{(i)} \cdot v^{(i)}}{\|u^{(i)}\| \, \|v^{(i)}\|}, \tag{9}$$

similarity score $Z^{(i)} \in [-1, 1]$

B. Datasets

We evaluate RRC-LLM on two complementary corpora that balance real-world complexity with fully open access. We captured a total of **2 524** Radio Resource Control (RRC) sessions, amounting to **30 247** *dialog pairs*, on a 5GNR network. Measurements were collected with off-the-shelf smartphones from multiple major UE vendors and gNB deployed by a single infrastructure supplier. Only metadata essential for protocol analysis (timestamp, direction, and cell identifier) was retained, and all vendor and location details were anonymised.

Owing to confidentiality reasons, this dataset is not publicly released.

To complement the proprietary log with fully reproducible data, we also generated a synthetic corpus using our open-source `nrRRC_Simulator` [13]. The resulting `RRC` dataset [14] contains paired uplink and downlink messages produced by a standards-compliant LTE core and eNodeB stack built on SRSRAN. This simulator primarily captures RRC messages related to connection setup, and transitions between IDLE and ACTIVE states under constant radio conditions. Each sample comprises: (i) an ISO-8601 timestamp, (ii) message direction (*UL/DL*), (iii) the ASN.1-encoded RRC frame, and (iv) simulator ground-truth labels (e.g. layer-3 procedure name). The dataset is released under the Apache 2.0 licence and can be obtained from the Hugging Face Hub [14].

C. Inference Results

Empirical analysis demonstrates that our fine-tuned RRC-LLM model generates responses with significantly improved fidelity to real-world RRC messages compared to a baseline Llama model. Our approach achieves higher BERT-based similarity scores, reduces inference latency, and aligns better with real-world RRC message patterns.

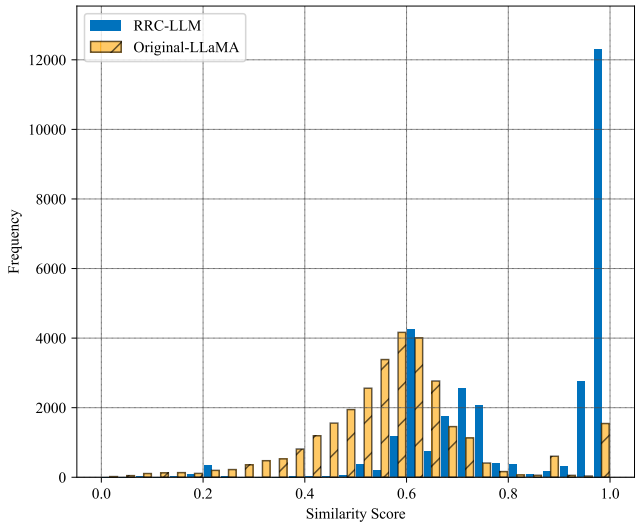

Fig. 8. Histogram of Similarity Scores

a) Comparative Analysis of Similarity Distributions: Figure 8 illustrates the distribution of similarity scores. Yellow segments correspond to the original Llama model, while blue segments correspond to the fine-tuned RRC-LLM. Notable observations include:

- **Two-Peak Distribution (RRC-LLM).** Fine-tuning creates an additional peak at high similarity (0.8–1.0), reflecting improved alignment with the intended RRC protocol format.
- **Higher Overall Scores.** The fine-tuned RRC-LLM clusters more frequently around higher similarity scores, indicating enhanced structural fidelity and reduced generation of non-protocol text.
- **Baseline Llama Limitations.** The original Llama tends to generate mid-range similarity outputs (around 0.5–0.6), often including extraneous natural language text instead of well-structured RRC fields.

The baseline Llama exhibits a median similarity of **0.60**, whereas the fine-tuned RRC-LLM reaches a median of **0.97**. This constitutes an absolute increase of **0.37**—a relative gain of approximately **61%**—highlighting the substantial improvement in structural fidelity delivered by the fine-tuning process.

b) Baseline comparison with public LLMs: To contextualize the performance of our fine-tuned RRC-LLM, we conducted a zero-shot evaluation using four publicly available LLMs[1]: Gemini 1.5-Flash, GPT-4o, Claude 3.5 Sonnet v2, and o3-mini. Table III presents the median cosine similarity scores for these public LLMs on subsets of both the synthetic and field-collected datasets, alongside the available scores for our RRC-LLM.

RRC-LLM was trained and frozen before the synthetic-simulator benchmark was finalised; consequently, that row in Table III is left blank. Because the simulator covers only a narrow slice of RRC procedures, we focused our evaluation budget on the far larger and more diverse field-collected corpus (30,247 request-response pairs), which offers a stronger indicator of real-world performance. All subsequent analysis—including the similarity-score distribution in Figure 8—therefore reflects this production-grade dataset, where RRC-LLM achieves a median cosine similarity of 0.97, comfortably outperforming every public baseline. A full rerun on the synthetic set is planned for a future release.

Evaluation Process: The zero-shot assessment for the public LLMs (Gemini 1.5-Flash, GPT-4o, Claude 3.5 Sonnet v2, and o3-mini) was performed for efficiency on a focused test set. This set consisted of 10 unique uplink-downlink (UL-DL) Radio Resource Control (RRC) message pairs from our synthetic simulator dataset and an additional 10 unique UL-DL pairs from our field-collected dataset. Each model was initially prompted with the instruction: *"You are playing the role of a base station and attempting to respond to an uplink RRC message with an appropriate downlink RRC message."* Subsequently, individual uplink (UL) RRC messages from this 20-pair combined test set were provided one at a time, and the model's generated downlink (DL) RRC message was collected for analysis.

Qualitative Observations: Beyond the quantitative similarity scores, several qualitative behaviors were noted during these evaluations.

- A common characteristic across all evaluated public LLMs was the inclusion of explanatory "wraparound" text accompanying their generated RRC replies. As this text is extraneous to the protocol message itself and can be trivially removed, we processed the outputs to extract the pure ASN.1 RRC message for assessment.
- GPT-4o exhibited a notable inconsistency in maintaining its assigned role as a base station. While it correctly responded with a DL RRC message to the first UL message, it subsequently appeared to 'forget' its role,

[1]Model snapshots: GPT-4o (OpenAI, Snapshot 2024-11-20) [15], o3-mini (OpenAI, Snapshot 2025-01-31) [16], Gemini 1.5-Flash (Google DeepMind, Snapshot 2024-09-24) [17], and Claude 3.5 Sonnet (Anthropic, Snapshot 2024-10-22) [18].

defaulting to commenting on or describing the content of later UL messages rather than generating appropriate DL RRC replies. This behavior likely impacted its overall similarity score (as seen in Table III).

- In contrast, Claude 3.5 Sonnet demonstrated a surprisingly nuanced understanding of RRC procedures. For example, it was the only public model among those tested that correctly identified scenarios where no downlink reply is necessary, such as after receiving an *RRCReconfigurationComplete* message. Furthermore, Claude 3.5 Sonnet displayed an advanced level of contextual understanding by, for instance, generating a *UECapabilityEnquiry* message when it detected discrepancies in RRC transaction identifiers between past and current UL RRC messages. This level of protocol awareness contributes to its strong performance reflected in Table III).

These qualitative insights highlight the varying degrees of protocol literacy and contextual reasoning exhibited by general-purpose LLMs when applied to the specialized domain of RRC message generation without specific fine-tuning. While some models show promise, specialized models like our RRC-LLM, extensively trained on 30,247 field data pairs and evaluated (as shown in Figure 8), demonstrate superior and more consistent protocol literacy.

TABLE III
MEDIAN COSINE SIMILARITY ON THE SIMULATOR DATASET AND FIELD-COLLECTED DATASET MEASUREMENTS.

Model	Synthetic data	Field data
Gemini 1.5-Flash	0.690	0.585
GPT-4o	0.710	0.496
Claude 3.5 Sonnet v2	0.728	0.768
GPT-o3-mini	0.678	0.705
Original Llama 3-8B	0.637	0.600
RRC-LLM (Ours)	N.A.	**0.970**

c) Similarity Responses: Table IV summarises three representative cases. In the *high-similarity* row (*sim. = 0.9991*), the fine-tuned RRC-LLM reproduces the reference message almost verbatim, diverging only in metadata such as timestamps; these negligible differences ($\leq$0.1 %) leave protocol fidelity intact. The *medium-similarity* example (*sim. = 0.92*) reveals a different behaviour: while the generated message retains the overall hierarchy, the model inserts an unsolicited `securityConfig{...}` fragment. The addition is syntactically legal and does not break ASN.1 encoding, yet it was not present in the ground-truth label—hence the reduced similarity. Finally, the *low-similarity* row (*sim. = 0.68*) shows the baseline result show emitting free-form commentary interleaved with partial RRC fields and even reversing the message direction, all of which markedly depress the score.

Overall, the fine-tuned RRC-LLM tends to preserve structure, including the sequential field order consistent with the ASN.1 definitions as learned from the linearized training data; discrepancies at the medium-similarity level typically stem from extra but valid sub-blocks such as `securityConfig`. In contrast, the untuned LLaMA often drifts into human-language prose, explaining its consistently lower similarity distribution.

d) Timestamp Handling: Our system uses timestamps from uplink RRC messages as additional input features, which helps the model maintain ordering and context during training. While the model also generates timestamps in its output for engineering convenience, these do not materially affect the similarity assessment.

Overall, fine-tuning the RRC-LLM has introduced a high-similarity peak, improved protocol adherence, and produced a more diverse similarity distribution, underscoring the model's superior performance on structured protocol tasks.

D. Training and Inference Cost

a) Hardware Topology and Parallelism.: All experiments were conducted on nodes equipped with two NVIDIA A100 GPUs (80GB PCIe), each interconnected via **NVLink** and connected to the host through a **PCIe switch**. We employed **tensor parallelism** to split model weights across the two devices, enabling pipelined execution without full model replication. This topology sustained up to 300 GB/s intra-node bandwidth and preserved throughput across long RRC sequences.

In tensor parallelism, large matrix multiplications are divided at the tensor level across multiple devices. During each forward and backward pass, intermediate activations and gradients are exchanged synchronously between GPUs. While PCIe supported host-GPU data flow, NVLink was the primary communication path between GPUs for high-bandwidth peer-to-peer transfers during collective operations (e.g., all-reduce). This setup allowed for efficient compute scaling but introduced tight coupling—any delay on one GPU could block progress on the other, making training particularly vulnerable to communication faults and imbalances.

b) Latency and Inference Throughput: Table V details the system specifications used throughout training and inference. Training consumed up to 110 GB of total GPU memory across the two A100 cards, while inference in GGUF format required approximately 29 GB per thread. Each GPU operated under CUDA 12, with tensor parallelism executed entirely on-device to minimize CPU-GPU communication latency.

To reduce computational resource usage during inference, we adopted llama.cpp, a lightweight C++-based runtime optimized for quantized LLMs. Compared to PyTorch-based inference, this approach significantly reduced VRAM consumption, requiring only 29 GB per thread in GGUF format—and lowered GPU utilization. This design choice aligns with the practical constraints of real-world deployment scenarios, where minimizing hardware requirements is crucial. In our experiments, it enabled us to assess whether the current setup could realistically meet the latency constraints imposed by mobile network standards. During fine-tuning, we used PyTorch Distributed (torch.distributed) to partition the model across two A100 GPUs, maximizing memory efficiency and parallelism.

The median inference time per RRC message is 6.9 seconds, with an average of 10.4 seconds. This falls short of real-time operation but remains practical for protocol emulation, offline testing, and system debugging. For reference, typical values for RRC timers (e.g. T300, T301) are in the order of milliseconds.

TABLE IV
EXAMPLES OF RRC-LLM RESPONSES ACROSS SIMILARITY LEVELS

High-Similarity (Similarity = 0.9991)

True Label: RRC Message

```
Time: 2024-04-15 11:34:38.469892
Message name: rrcSetup
Interface: F1_E1
Protocol: nrRrc
Time diff: 0000:00:00.001898
Message body: DL-CCCH-Message: {
 message c1 : rrcSetup : {
    rrc-TransactionIdentifier 0,
    criticalExtensions rrcSetup: {
      radioBearerConfig {
       srb-ToAddModList {
        {
         srb-Identity 1
        }
       }
...
```

RRC-LLM (Similarity = 0.9991)

```
Time: 2024-04-15 11:34:38.469982
Message name: rrcSetup
Interface: F1_E1
Protocol: nrRrc
Time diff: 0000:00:00.001988
Message body: DL-CCCH-Message: {
 message c1 : rrcSetup : {
    rrc-TransactionIdentifier 0,
    criticalExtensions rrcSetup: {
      radioBearerConfig {
       srb-ToAddModList {
        {
         srb-Identity 1
        }
       }
...
```

Medium-Similarity (Similarity = 0.92)

True Label: RRC Message

```
...
           headerCompression notUsed : NULL,
           statusReportRequired true
          },
          t-Reordering ms100
         }
        }
       }
      },
      measConfig {
       measObjectToAddModList {
        {
         measObjectId 1,
         measObject measObjectNR : {
           ssbFrequency 634080,
           ssbSubcarrierSpacing kHz30,
           smtc1 {
            periodicityAndOffset sf20 : 0,
            duration sf5
           },
...
```

RRC-LLM (Similarity = 0.92)

```
...
           headerCompression notUsed : NULL
          },
          t-Reordering ms100
         }
        }
       },
       securityConfig {
        securityAlgorithmConfig {
         cipheringAlgorithm nea2,
         integrityProtAlgorithm nia2
        }
       }
      },
      measConfig {
       measObjectToAddModList {
        {
         measObjectId 1,
         measObject measObjectNR : {
           ssbFrequency 634080,
           ssbSubcarrierSpacing kHz30,
           smtc1 {
            periodicityAndOffset sf20 : 0,
            duration sf5
           },
...
```

Low-Similarity (Similarity = 0.68)

True Label: RRC Message

```
Time: 2024-04-15 11:33:10.690388
Message name: rrcRelease
Interface: F1_E1
Protocol: nrRrc
Time diff: 0000:00:10.633832
Message body: DL-DCCH-Message: {
 message c1 : rrcRelease : {
    rrc-TransactionIdentifier 0,
    criticalExtensions rrcRelease: {
     }
   }
}
...
```

RRC-LLM (Similarity = 0.68)

```
It looks like there's another RRC reconfiguration
Complete message being sent!
Message name: rrcReconfigurationComplete
Time: 2024-04-15 11:33:00.243997
Interface: F1_E1
Protocol: nrRrc
Time diff: 0000:00:00.187441
Message body: UL-DCCH-Message: {
 message c1 : rrcReconfigurationComplete : {
    rrc-TransactionIdentifier 3,
    criticalExtensions rrcReconfigurationComplete: {
     }
   }
}
...
```

TABLE V
SERVER SPECIFICATIONS FOR MODEL TRAINING AND INFERENCE

Component	Specification
CPU	52-core hyper-threaded
RAM	512 GB
GPU	2 × NVIDIA A10080GB PCIe NVLink
GPU VRAM (Inference)	∼29 GB per thread (GGUF format)
GPU VRAM (Training)	∼110 GB across two cards
CUDA Version	v12

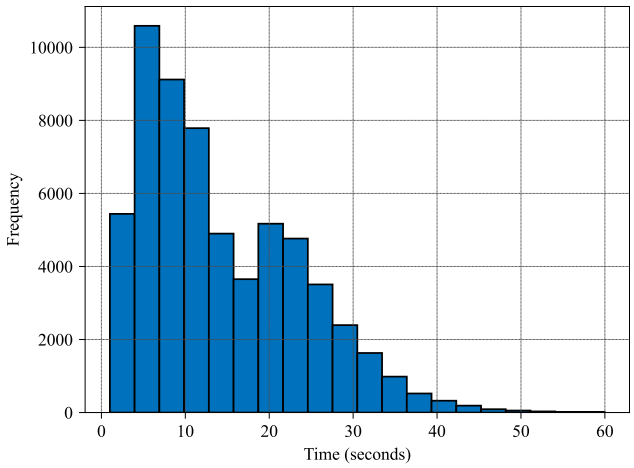

Fig. 9. Distribution of Sentence Generation Times

Future work will explore smaller and more efficient models to speed this up. Figure 9 shows the distribution of sentence generation times, where longer examples correlate with multi-field expansions and nested structures.

E. Discussion

Fine-tuning LLAMA-8B on the RRC corpus surfaced a critical—albeit rare—edge case: a small subset of control-plane exchanges, most notably the elaborate `rrcSetup` procedures, exceed the model's 8,192-token context window. While the base model supports sequences up to 8,192 tokens during inference, we fine-tuned it using a 4,096-token limit (Table I) to ensure training stability and efficient GPU memory usage. This discrepancy reflects a deliberate trade-off: shorter training windows reduce memory fluctuation and mitigate out-of-memory risks, while inference utilizes the full window to accommodate longer messages.

Rather than discarding these over-length examples, we retained them by splitting each message at a logical boundary and inserting a dedicated *segmentation marker*. This marker signals to the model that reasoning continues in the next segment, ensuring that every RRC transaction is fully represented during both training and inference. Only a small fraction of messages required such segmentation.

During inference, we apply an identical routine. The model first consumes the leading segment terminated by the marker; the subsequent segment (and, on rare occasions, a third) is then supplied with the marker prepended, and the partial generations are concatenated. This pragmatic workflow avoids architectural modifications and guarantees that no generation call breaches the positional-embedding ceiling. The trade-off is that each segment is decoded without direct visibility of its predecessors, so perfect discourse continuity cannot be ensured.

Empirically, the stitched outputs remain syntactically valid but occasionally exhibit duplicated fields or minor inconsistencies—a manifestation of the lost global context that a full-window transformer would capture. While such deviations are unacceptable in a live control plane, their impact on our offline evaluation proved negligible given the scarcity of over-length messages. Nevertheless, the study exposes an intrinsic limitation of fixed-window architectures when they confront protocol stacks with unbounded message length.

V. CONCLUSION AND FUTURE WORK

This paper has investigated the question of whether a decoder-only Large AI Model (LAM) can acquire 5G NR RRC *protocol literacy*. By fine-tuning an 8-billion-parameter LLaMA model with LoRA adapters, our RRC-LLM achieved approximately a **61% relative gain** in median cosine similarity with ground-truth RRC messages compared to a zero-shot LLM baseline. The model now reproduces hierarchical field dependencies in RRC messages, resolves partial inputs, and generates standards-compliant responses, providing empirical evidence that LAM-based RRC emulation is a viable building block for AI-native air interfaces and industry visions such as ETSI ENI 051's AI-Core.

Operational challenges. Despite the performance gains, three issues still impede the real-time execution of this model and deployment at scale: (i) computation and energy overhead (ii) autoregressive latency that strains sub-millisecond control budgets; and (iii) finite context windows that truncate multi-step procedures requiring long RRC message sequences.

Future work. The following is a non-exhaustive list of the most promising research lines on the path towards AI-native RRC layers:

1) *Extended-context modelling:* To support full-length RRC exchanges without truncation, extended-context mechanisms will be needed, including memory-augmented transformers with cached activations, hierarchical segment encoders, and efficient attention variants such as Longformer, BigBird, RoPE-extended LLaMA, and ALiBi [19]–[24]. Through this combined approach, coherence and field-level fidelity are expected to be preserved across long control-plane sequences.
2) *Retrieval-Augmented Generation (RAG):* A non-parametric memory will be attached to the RRC-LLM, indexing clause-level 3GPP specifications (Rel. 15–19), curated traces, and streaming telemetry. During inference, relevant passages will be retrieved via a dual-encoder ranker, re-scored by a cross-encoder, and injected into the prompt under a fixed template [25]. To capture long-range dependencies such as cascading timers, a Graph-RAG pipeline can be used to enable, enabling structured retrieval of interconnected subgraphs [26].

3) *Model compression and hierarchical control:* Inference efficiency can be further enhanced through knowledge distillation, quantization [27], and the adoption of protocol-aware lightweight architectures. To achieve sub-second inference latency, hierarchical model designs could be explored, wherein compact front-end agents respond rapidly to RRC events, while larger back-end models perform higher-level reasoning. This design remains compatible with orchestration frameworks such as ETSI ENI-051 [5], though the emphasis is placed on reducing latency and improving resource utilization efficiency.

The proposed efficiency measures—model compression, hierarchical agent architectures, and long-context techniques [28]—are expected to reconcile the RRC-LLM's message-level fidelity with the tight latency and energy envelopes of 5G RAN. In practical terms, they align with emerging multi-agent AI designs (e.g., those sketched in ETSI ENI-051) by translating high-level cognitive decisions into standards-compliant RRC messages without exceeding sub-millisecond control budgets. By curbing runtime overhead while maintaining accuracy over extended RRC sessions, these measures move LAM-based emulation from proof-of-concept toward deployment inside modular, AI-native base stations, thereby realizing the introductory claim that LAM can serve as integral design primitives of the AI-AI rather than merely auxiliary optimizers.

Looking beyond the 5G horizon, further acceleration and robustness will be pursued through computation-efficient backbones such as state-space models, aggressive quantisation, speculative decoding, and sparse activation. When combined with the aforementioned toolkit, these advances are projected to elevate LAM-driven RRC from laboratory prototype to real-time, energy-aware operation in ultra-dense, low-latency 6G networks where multi-modal, self-explaining agents co-design physical- and MAC-layer procedures. In this way, the research trajectory outlined here closes the loop between the conceptual promise of an AI-AI and its concrete instantiation in next-generation RAN deployments.

APPENDIX A
SYSTEM INTERRUPTIONS AND RELIABILITY BOTTLENECKS.

Out of 73 full training attempts totaling **178.6 GPU-hours**, only 8 completed without interruption. The remaining **65 failures**—recovered via checkpoint resumes—introduced significant wall-clock delays and impacted epoch scheduling variance.

As shown in Table VI, the most common failure categories were host-level faults—including SSD stalls and NCCL watchdog timeouts—followed by GPU memory overflows and hardware issues. SSD interruptions, accounting for 16.9% of failures, resulted from limited local disk capacity during frequent incremental checkpointing and data flushing. We mitigated this by expanding storage in later runs. GPU-side issues, including CUDA out-of-memory (OOM) errors and device instability, made up 21.6% of interruptions. These were generally easier to isolate and recover from. A further 35.4% of cases could not be conclusively categorized and are labeled as "Other."

TABLE VI
INTERRUPTIONS BY COMPONENT

Component	Category	Interruptions	%
Software Bug	Host	8	12.3
Faulty GPU	GPU Maintenance	7	10.8
SSD issue	Maintenance	11	16.9
NCCL Watchdog Timeout	Host	9	13.8
CUDA OOM	GPU	7	10.8
Other	Maintenance	23	35.4

REFERENCES

[1] A. Valcarce and J. Hoydis, "Toward joint learning of optimal mac signaling and wireless channel access," *IEEE Trans. Cogn. Commun. and Networking*, vol. 7, no. 4, pp. 1233–1243, Dec. 2021.

[2] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6G AI-Native Air Interface," *IEEE Communications Magazine*, vol. 59, no. 5, pp. 76–81, May 2021.

[3] OpenAI, "Gpt-4o system card," https://openai.com/index/gpt-4o-system-card/, 2024, accessed: 2025-05-13.

[4] G. Team, R. Anil, and S. Borgeaud, "Gemini: A family of highly capable multimodal models," 2025.

[5] ETST, "Experiential Networked Intelligence (ENI); Study on AI Agents based Next-generation Network Slicing," ETSI, Tech. Rep. GR ENI 051 V4.1.1, Feb. 2025.

[6] 3GPP, "NR; Radio Resource Control (RRC); Protocol specification," 3GPP, Tech. Rep. TS 38.331, 2024, available at https://www.3gpp.org.

[7] H. Touvron, T. Lavril, G. Izacard, and et al., "LLaMA: Open and Efficient Foundation Language Models," *arXiv:2302.13971*, 2023.

[8] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," *arXiv:2106.09685*, 2021.

[9] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv:1508.07909*, 2016.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv: 1412.6980*, 2017.

[11] AI@Meta, "Llama 3 model card," *Unpublished work*, 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[12] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[13] Z. Liu, "nrRRC_Simulator: A RRC protocol emulator and data collection toolkit," https://github.com/EE-zim/nrRRC_Simulator, 2025, git commit main@HEAD; accessed 2025-05-13.

[14] ——, "RRC dataset for radio resource control conversations," https://huggingface.co/datasets/EEzim/RRC, 2025, accessed 2025-05-13.

[15] OpenAI, "Chatgpt-4o model card (gpt-4o-2024-11-20)," https://platform.openai.com/docs/models/gpt-4o, 2024, accessed: 2025-05-13.

[16] OpenAI, "o3-mini: A Cost-Efficient Reasoning Model," https://openai.com/index/openai-o3-mini/, Jan 2025.

[17] Google DeepMind, "Gemini 1.5 Flash (`gemini-1.5-flash-002`) Model Card," https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/1-5-flash, Sep 2024.

[18] Anthropic, "Claude 3.5 Sonnet v2 Documentation," https://www.anthropic.com/news/claude-3-5-sonnet, Jun 2024.

[19] J. W. Rae, A. Potapenko, C. Fernando, and T. Lillicrap, "Compressive transformers for long-range sequence modelling," in *International Conference on Learning Representations (ICLR)*, 2020.

[20] J. Zhang, K. A. Afshar, A. Trischler, and J. C. K. Cheung, "Hierarchical transformers are more efficient for long document processing," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[21] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," in *Findings of ACL*, 2020.

[22] M. Zaheer, G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Onta n'on, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big Bird: Transformers for longer sequences," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[23] J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," in *Findings of ACL*, 2021.

[24] O. Press, N. Smith, and O. Levy, "Train short, test long: Attention with linear biases enables input length extrapolation," *arXiv:2108.12409*, 2022.

[25] R. Zhang, H. Du, Y. Liu, D. Niyato, J. Kang, S. Sun, X. S. Shen, and H. V. Poor, "Interactive AI with retrieval-augmented generation for next generation networking," *IEEE Network*, vol. 38, no. 6, pp. 414–424, Nov. 2024.

[26] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, and S. Tang, "Graph retrieval-augmented generation: A survey," *arXiv:2408.08921*, 2024.

[27] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv:1503.02531*, 2015.

[28] Z. Dai, Z. Yang, Y. Yang, W. W. Cohen, R. Salakhutdinov, and Q. V. Le, "Transformer-xl: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 2978–2988.