

# ByteFlow: A Byte-Level LLM for Deep Packet Inspection and Network Intelligence

Abanisenioluwa Orojo

Baylor University

Email: abanisenioluwa\_oroj1@baylor.edu

Erika Leal

Baylor University

Email: Erika\_Leal@baylor.edu

Emmanuelli El-Mahmoud

Baylor University

Email: emmanuelli\_elmahmou1@baylor.edu

Pablo Rivas

Baylor University

Email: Pablo\_Rivas@baylor.edu

**Abstract**—In recent years, the complexity and volume of network traffic have increased significantly, making it challenging to effectively identify malicious activities. While machine learning (ML) and Large Language Models (LLMs) have emerged as powerful tools, existing approaches often suffer from key limitations. They frequently rely on converting raw packets into less space efficient text or hexadecimal formats; a process that can increase the input size by 100% in hexadecimal case and over 500% in the text case. There is also the use of biased instruction datasets with leading questions that hinder genuine analysis. To overcome these challenges, we propose ByteFlow, a byte-based network intelligence LLM framework designed for efficiency and deep understanding of network traffic. ByteFlow operates directly on the raw byte stream and features three core innovations: tokenization strategy for effectively processing nuanced network flows in their native format; a 2D positional embedding (2DPE) that captures the structural layout of network flows; and a unique pre-training strategy that uses an open-ended, non-leading question-and-answer format to build true contextual and semantic awareness of network traffic. We have successfully developed a 200M parameter proof-of-concept (PoC) model that validates the effectiveness of our approach. This PoC serves as a foundational framework, demonstrating the viability of ByteFlow and paving the way for training larger, more powerful models. By processing network data more efficiently and intelligently, ByteFlow represents a significant advancement in creating adaptive and robust cyber threat detection systems.

**Index Terms**—Network Traffic Analysis (NTA), Large Language Models (LLMs), Byte-Level Processing, Deep Packet Inspection (DPI), Positional Embedding

## I. INTRODUCTION

Computer networks serve as the backbone of modern-day communication, powering the internet, critical infrastructure systems, and mobile telecommunications. The proliferation of interconnected systems and the growing reliance on distributed computation have introduced complex security challenges, catalyzing a significant surge in the frequency and sophistication of Cybersecurity threats. Adversaries increasingly exploit network traffic as a primary vector to execute a wide spectrum of malicious activities, including phishing, malware transmission, web application exploitation, and distributed network attacks by embedding stealthy signals within legitimate communication flows. Network intrusions remain a pervasive threat to data integrity, system availability, and user privacy, often exploiting

structural weaknesses across large-scale digital infrastructures. A foundational approach to mitigating these threats lies in the analysis of network traffic, a task that requires modeling complex, high-dimensional temporal data streams. Accurate traffic analysis enables not only the detection of anomalous or malicious behaviors but also supports performance optimization and protocol compliance validation across networked systems. Recent advances in machine learning have significantly enhanced capabilities in this domain, enabling the extraction of rich latent representations from traffic data for threat detection and behavioral profiling [1], [2]. Consequently, the design of scalable and robust models for network traffic analysis has become central to modern Cybersecurity research and operations [3].

Recent advances in machine learning, particularly deep representation learning, have shown considerable promise for Network Traffic Analysis (NTA), enabling the detection of complex and subtle patterns in both benign and malicious traffic distributions [4], [5]. These methods excel at uncovering latent structures in high-dimensional network telemetry such as packet captures (PCAPs), often outperforming rule-based and statistical approaches. However, their deployment in practical NTA scenarios remains limited due to several key challenges. First, most ML-based methods lack interpretability, rendering them inadequate for high-stakes security environments where human analysts require semantically meaningful insights [6]. Second, these approaches typically rely on handcrafted or intermediate feature extraction pipelines that may obscure critical temporal and protocol-level semantics [7]. Finally, their generalization to diverse downstream NTA tasks such as anomaly detection, attack attribution, and traffic classification is often poor, limiting their robustness and transferability across network environments [3].

Large Language Models (LLMs) have demonstrated state-of-the-art capabilities across a wide range of natural language processing tasks, including pattern extraction, contextual reasoning, and semantic comprehension [8]. Motivated by these advances, recent studies have begun exploring the applicability of LLMs to domains beyond text, such as encrypted traffic classification [9]–[11], synthetic network traffic generation [12], and packet-level behavioral analysis [13]. Despite this progress,

several critical limitations remain unaddressed:

- 1) The constrained context window sizes of current LLMs hinder accurate modeling of long-range dependencies in network traffic.
- 2) Prevailing tokenization schemes, originally designed for textual data, are ill-suited for structured and byte-level network inputs, leading to inefficiencies across architectures.
- 3) Limited investigation into incorporating higher-level representations, such as multi-packet flow features, restricts model generalization and robustness. Addressing these challenges is essential for harnessing the full potential of LLMs in real-world network environments.

These challenges are compounded by two prevailing methodological shortcomings in recent literature. First, many current frameworks sidestep the complexity of raw network data by converting it into less space efficient, intermediate formats. For instance, models such as NetGPT, Lens, and TrafficFormer translate raw packet bytes into hexadecimal strings before tokenization [14]–[16]. This approach significantly inflates the input sequence length, often by more than 100% which consumes valuable context window space and treats the data as a simple character stream, obscuring the rich structural and positional semantics inherent in packet layouts. Second, the training methodologies employed can inadvertently limit a model’s analytical depth. The TrafficLLM framework, for example, parses network flows into human-readable key-value pairs and utilizes instruction-based fine-tuning [17]. While this adapts LLMs to network tasks, it often frames the analysis around leading questions (e.g., “instruction”: “I have a copy of traffic that may be botnet traffic, but I don’t know what type of network it belongs to. Can you give me some information?”, “output”: “Botnet Detection”). Such a paradigm risks teaching the model to perform superficial pattern matching on text rather than developing a genuine, generative understanding of the underlying byte-level evidence, hindering its ability to detect novel or complex threats.

In this paper, we propose ByteFlow, an approach centered around the transformer architecture [18] to perform network traffic analysis directly on the raw byte representation of network packet capture files. Our key contributions include a tokenization scheme, a positional embedding strategy, a hybrid attention mechanism, and tailored training techniques for PCAP and text inputs. Specifically, our tokenization approach considers network flows and introduces special tokens such as  $\text{;sep}_i$ , which help in delineating individual header fields within packets, preserving both temporal and semantic information. Building on this structured, tokenized input, we implement a 2DPE strategy. This allows the model to understand the inherent row-and-column nature of packet data. This 2D understanding is directly leveraged within our attention layer, which incorporates a multi-layer perceptron to dynamically combine attention scores calculated across both standard sequential (row-wise i.e packets) and permuted structural (column-wise i.e flow) views of the network data. This allows ByteFlow to capture complex

dependencies within the structured and sequential aspects of network traffic. This enhanced, structural understanding aims to directly bolster security operations. For instance, accurate, deep parsing can unmask evasion techniques hidden within protocol nuances or deliberately malformed packets, which simpler models might miss. Furthermore, enabling analysts to perform natural language queries against raw traffic captures can significantly accelerate incident response, forensic analysis, and proactive threat hunting efforts, providing a more intuitive and powerful interface for network intelligence.

## II. RELATED WORK

### A. Network Traffic Analysis

Supervised deep learning has become a popular approach for traffic classification by using automatic feature extraction to capture underlying traffic patterns [7], with methods such as converting traffic into visual representations for CNNs [19], combining CNNs with RNNs to capture spatial and temporal features [20], and employing BLSTM networks on byte sequences [21], all achieving high accuracy. To handle encrypted traffic, ET-BERT utilizes pre-training transformers on large-scale unlabeled data with self-supervised tasks, demonstrating significant F1 score improvements over existing methods [9]. While these approaches are promising, particularly for classification, ByteFlow distinguishes itself by integrating a novel 2D positional encoding and a hybrid attention mechanism within a byte-level transformer, aiming to build a more foundational model that understands the intricate 2D structure of packet fields for diverse tasks like parsing and question-answering, not just classification.

### B. Byte LLMs

Character-level modeling has long served as a benchmark for neural architectures, from early recurrent models [22] to foundational contributions like ByT5, the first large-scale byte-level Transformer built on the T5 architecture that operates directly on UTF-8 bytes to remove subword tokenization [23], [24]. While ByT5 matches performance on many multilingual benchmarks, it can struggle on English classification tasks when scaled beyond 1 billion parameters. Building on these token-free advancements [24], [25], the Byte Latent Transformer (BLT) introduced a novel architecture that encodes raw byte sequences into dynamically sized latent patches, demonstrating competitive performance and inference benefits [26]. For our model, ByteFlow, we chose an encoder-decoder structure, which provided a solid foundation for our primary research goal: developing novel, model-agnostic tokenization and 2D-aware attention mechanisms specifically tailored for the unique structure of network traffic.

### C. Network Traffic LLMs

While several transformer-based models have been developed for network traffic analysis, they exhibit notable limitations. For instance, one model applies linear attention for efficient processing of raw PCAP data for generation and classification

but under-represents hierarchical semantics and uses a questionable data split that risks overfitting (**99:1**) [12], [18], [27], [28]. Another model, NetGPT, adapts the GPT-2 architecture by converting bytes to a hex format but is hampered by a limited training set suboptimal for traffic understanding [14]. Similarly, TrafficLLM employs a dual-stage fine-tuning framework with PEFT to outperform many generation methods, yet it fails to capture contextual flow information for practical querying, it also heavily relies on a leading dataset in its Q/A portion [29]. In contrast, ByteFlow offers a distinct approach by operating directly on raw bytes to avoid the sequence length expansion of hex or text-based encodings [24], and it intentionally retains quadratic self-attention [18] to support complex reasoning for tasks like question-answering [8]. Most significantly, ByteFlow introduces a 2D positional encoding and hybrid attention mechanism to explicitly capture the hierarchical, field-level structure within packets, addressing a core limitation of prior models and aiming to provide a more versatile foundation for diverse network analysis tasks.

#### D. Comparative Analysis with Existing Models

To position ByteFlow within the broader landscape of network traffic analysis models, we compare its characteristics with several existing approaches. Table I summarizes key attributes of these models.

TABLE I  
COMPARISON OF NATIVE MODEL ARCHITECTURES AND SCALE.  
APPLICATION(S) LEGEND: CLS (CLASSIFICATION), GEN (GENERATION),  
QA (QUESTION ANSWERING).

Model	Parameters	Max Tokens	Application(s)
ET-BERT [9]	~110M	512	CLS
LENS [15]	~60M-220M	512	GEN, CLS
NetGPT [14]	~124M+	512	GEN, CLS
YaTC [30]	~110M	512	CLS
TrafficGPT [12]	~100M-300M*	3k / 12k	GEN, CLS
<b>ByteFlow (Ours)</b>	<b>200M</b>	<b>1024</b>	GEN, QA

### III. APPROACH

Our approach uses a byte-level Transformer (Figure 1) based on the ByT5-small model [24]. By operating directly on UTF-8 byte sequences, this token-free model avoids the information loss that conventional tokenization causes on inherently byte-structured network data [24]. This makes the architecture robust to the noisy and unconventional inputs common in network traffic [24]. While originally for natural language, ByT5 provides a strong foundation for our work. The subsequent sections detail our modifications for deep packet inspection.

#### A. Tokenization Strategy for Network Packets

Tokenizing network traffic for our ByT5-based model requires preserving both raw byte content and structural context. Our methodology addresses this by representing packet bytes directly while introducing special tokens. These tokens delineate crucial structural elements, such as header fields and

payloads, and encode temporal relationships between packets in a flow. This hybrid approach ensures complete information fidelity from the raw bytes while explicitly adding the semantic and temporal context necessary for effective analysis. Our tokenization process represents network traffic by combining direct byte encoding with a set of custom special tokens. Following the ByT5 framework, we map each of the 256 possible byte values to a unique token ID. To provide structural and semantic context, we introduce the following tokens: FLOW\_START, PACKET\_START, FIELD\_SEP, TIME\_DELTA (timestamp 8-byte exponential representation), LINK\_TYPE, EOS\_PACKET, FLOW\_END, TEXT\_START, TEXT\_END. The final vocabulary comprises these custom tokens, the 256 byte representations, and standard model tokens such as <PAD>, <EOS>, and <UNK>. This tokenization strategy aims to provide a rich, structured, yet flexible input representation to the model. By explicitly marking boundaries, we hypothesize that the model can better learn the syntax and semantics of packet headers, which is crucial for fine-grained traffic analysis and identifying anomalous field values or combinations. The direct byte representation ensures that the model has access to the complete, unaltered packet information, including potentially encrypted or obfuscated payloads.

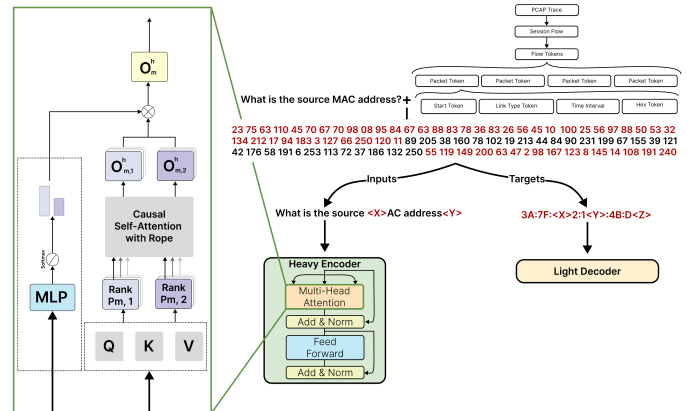


Fig. 1. An overview of the ByteFlow methodology. The main diagram shows the process from raw byte input and a natural language query (top) to the model, producing structured output (bottom). The inset (left) details the 2D-PE hybrid attention mechanism with its MLP and dual attention views.

#### B. Hybrid Attention with 2D Table Positional Encoding

Network traffic has an inherent two-dimensional (2D) structure—a sequence of packets (rows), each containing a sequence of fields (columns). Flattening this for standard 1D attention mechanisms, like those using RoPE, results in the loss of crucial spatial information. To address this, we incorporate a modified attention mechanism that uses Two-Dimensional Positional Encoding (2D-PE) for packet data, while retaining standard 1D attention for any accompanying text. This allows attention heads to learn relationships both across packets (row-wise) and across fields within packets (column-wise), preserving the traffic’s natural 2D structure.

The core idea is that the attention score  $a_{m,n}^h$  between a query token  $x_m$  and a key token  $x_n$  for head  $h$  becomes a function of their positions in the dynamically chosen 2D permutation,  $p_{m,j}$  and  $p_{n,j}$  for permutation  $j$ :

$$o_m^h = \sum_{j=1}^J r_{m,j}^h \cdot \text{Attention}(q_m^h, K_j^h, V_j^h; P_j) \quad (1)$$

where  $r_{m,j}^h$  is the weight for permutation  $j$ , and  $\text{Attention}(q_m^h, K_j^h, V_j^h; P_j)$  is the attention output calculated using query  $q_m^h$  and keys/values  $K_j^h, V_j^h$  based on the positional encoding  $P_j$  from the  $j$ -th permutation order.

### C. Pre-training Strategy

We use a self-supervised pre-training phase to give our model a foundational understanding of both general language and network traffic structures. This phase utilizes a large, unlabeled corpus composed of plain text and raw PCAP data. Our primary pre-training objective is the span corruption task from [24], where the model learns to reconstruct corrupted input spans. The training objective is to minimize the negative log-likelihood of the target tokens, as defined in Equation 2.

$$\mathcal{L}_{SC} = - \sum_{t=1}^T \log P(y_t | Y_{in < t}, X_{corrupt}; \theta) \quad (2)$$

When applying 2D-PE to PCAP data, we introduce an auxiliary loss ( $\mathcal{L}_{ENT}$ ) to regularize the attention mechanism’s network. This loss encourages the weights ( $r_{m,j}^h$ ) to become more discriminative, favoring specific permutation orders over a uniform blend. The loss is the average over the weights for each token, as defined in Equation 3.

$$\mathcal{L}_{ENT} = \frac{1}{N_{layers} N_{heads} M} \sum_{l=1}^{N_{layers}} \sum_{h=1}^{N_{heads}} \sum_{m=1}^M H(R_{l,h,m}) \quad (3)$$

This auxiliary objective is combined with the primary span corruption loss to form the total pre-training loss,  $\mathcal{L}_{total}$ , weighted by a hyperparameter  $\lambda_{ENT}$  (Equation 4). This combined objective trains the model to master byte-level sequence modeling while effectively leveraging the 2D structure of PCAP data.

$$\mathcal{L}_{total} = \mathcal{L}_{SC} + \lambda_{ENT} \mathcal{L}_{ENT} \quad (4)$$

## IV. EXPERIMENTS

We conducted experiments to validate our ByteFlow framework, particularly its 2D-aware tokenization and hybrid attention mechanism. Our evaluation assesses the model’s byte-level understanding via pre-training and its performance on downstream network tasks through multi-task fine-tuning. This section outlines our experimental methodology, detailing the datasets in Section IV-A, data pre-processing in Section IV-C, training setup in Section IV-D, and the downstream tasks and evaluation metrics in Section IV-E.

### A. Datasets

Our experiments leverage public datasets for pre-training and multi-task fine-tuning. For the initial pre-training phase, we use approximately 189 gigabytes of PCAP data compiled from five sources: ISCX-Tor2016 [31], USTCTFC2016 [32], ISCXVPN2016 [33], DoHBrw2020 [34], and CICIOt2022 [35]. These datasets provide diverse TCP/IP traffic (general, VPN, Tor, and IoT), from which we exclusively use the raw packet data. For general textual understanding, we also incorporate the English portion of the Colossal Clean Crawled Corpus (mc4-en) [23].

### B. Multi-Task Fine-Tuning and Downstream Tasks

In our multi-task fine-tuning phase (Phase 2), we adapt the pre-trained ByteFlow model to five downstream tasks simultaneously, leveraging shared representations from a combined, curated PCAP dataset. To direct the model, each input receives a task-specific prefix (e.g., BMLM) (See A), and training optimizes a summed loss function. The five tasks are: 1) Byte Masked Language Modeling (BMLM) on PCAP Data, which predicts masked bytes to evaluate contextual understanding; 2) Autoregressive Next-Byte Prediction on PCAP Data, which assesses sequential dependency knowledge; 3) Supervised Packet Parsing (Wireshark JSON Prediction), where the model translates raw packets into structured JSON to test protocol comprehension; 4) Question Answering (QA) on Packets, which requires generating natural language answers to questions about packet data (see B); and 5) Field Finding in a Specific Packet, a targeted retrieval task to extract a specific field’s value. Training on these diverse tasks concurrently enhances ByteFlow’s ability to map byte sequences to various semantic and structural interpretations, improving its overall network intelligence.

### C. Data Pre-processing and Augmentation

Our pre-processing pipeline begins with Flow Segmentation, where we segment raw PCAP data into five-tuple (source/destination IP, source/destination port, protocol) flows, discarding non-TCP/UDP packets similarly to [9]. After Tokenization with our custom strategy, the data is split into 80/20 training and testing sets. To improve model robustness and simulate real-world network conditions, we apply PCAP Data Augmentation to 20% of the training samples. These techniques include Duplication of packets to simulate retransmissions, Permutation of packet order to simulate reordering, and Translation of TIME\_DELTA values to simulate latency jitter. This process helps the model learn fundamental traffic patterns rather than overfitting to idealized data, thereby enhancing its generalization to live network environments.

### D. Training Setup

Our two-phase model training was conducted over 14 days on 16 NVIDIA H200 GPUs, with configurations based on the ByT5-small architecture [24]. In Phase 1 (Initial Pre-training), the model was trained for 1,000,000 steps on a combined PCAP and text corpus using the span corruption objective (Section III-C) with a 1024-token sequence length

and a 0.0001 learning rate. This phase activated our 2D-TPE and incorporated a regularization loss ( $\mathcal{L}_{ENT}$ ) with an optimal weight of  $\lambda_{ENT} = 0.01$ . In Phase 2 (Multi-Task Fine-Tuning), we adapted the model to five downstream tasks by jointly optimizing a combined loss function with a reduced learning rate of **0.00001**. These values represent the optimal hyperparameters selected after a grid search over ranges for the pre-training learning rate ( $10^{-5}$  to  $5 \times 10^{-4}$ ), fine-tuning learning rate ( $5 \times 10^{-6}$  to  $5 \times 10^{-5}$ ), and the  $\lambda_{ENT}$  weight ( $10^{-3}$  to  $10^{-1}$ ).

### E. Evaluation Metrics

The ByteFlow model’s performance is assessed across its tasks using tailored automated metrics. For the byte-level tasks, Byte Accuracy measures the percentage of correctly predicted masked bytes for BMLM and the percentage of correctly predicted next bytes for Autoregressive Next-Byte Prediction. The Supervised Packet Parsing; Wireshark JSON Prediction task is evaluated on both JSON Accuracy, which assesses structural and content correctness (often via Exact Match or tree-edit distance), and Field Accuracy, which calculates the percentage of correctly identified key-value pairs. For QA on Packets, we use a standard suite of three metrics: the Token F1 Score for token overlap, the ROUGE-L Score for the longest common subsequence, and Exact Match (EM) for the percentage of perfect answer matches. Finally, the Field Finding in a Specific Packet task is evaluated solely by the EM metric, measuring the percentage of precisely matching extracted field values.

## V. ANALYSIS AND RESULTS

This section evaluates the ByteFlow framework’s performance across five distinct network intelligence tasks. Our analysis focuses on how the phased training strategy, crucial domain-specific pre-training, and novel 2D Table Positional Encoding (2D-PE) each contribute to the model’s success. To isolate these contributions, we compare four distinct model variants:

- **Vanilla ByT5-small:** The un-trained baseline model.
- **Fine-tuned ByT5-small:** The baseline model after receiving only multi-task fine-tuning.
- **ByteFlow Base:** The model with domain-specific pre-training but without 2D-PE.
- **ByteFlow (Full):** The final proposed model, which incorporates both pre-training and 2D-PE.

This comparison allows us to position ByteFlow against existing network traffic analysis models and highlight its value for applied security.

### A. Impact of Phased Training and 2D Positional Encoding

While direct numerical comparison with existing literature is challenging due to differing task definitions, datasets, and the novelty of our QA and fine-grained parsing tasks on raw bytes, the ByteFlow framework—with its 200M parameter ByT5-small base and unique combination of byte-level processing

TABLE II  
COMBINED PERFORMANCE OF BYT5-SMALL

Task	Metric	Base Value	Finetuned Value
BMLM	Byte Accuracy	0.0000	0.0178
Next-byte	Byte Accuracy	0.0026	0.1230
Parse	JSON Accuracy	0.0426	0.7595
	Field Accuracy	0.0000	0.5867
QA	Token F1	0.0000	0.2266
	Rouge-L	0.0000	0.2840
Find-byte	Exact Match	0.0000	0.0000

TABLE III  
COMPARISON OF BYTEFLOW BASE AND BYTEFLOW (FULL) PERFORMANCE

Task	Metric	ByteFlow Base (Value)	ByteFlow (Full) (+ 2D-PE Value)
BMLM	Byte Accuracy	0.6173	0.5924
Next-byte	Byte Accuracy	0.5286	0.5829
Parse	JSON Accuracy	0.8055	0.8125
	Field Accuracy	0.6520	0.6610
QA	Token F1	0.2730	0.4800
	Rouge-L	0.3215	0.3980
Find-byte	Exact Match	0.2250	0.4700

and 2D Positional Encoding (2D-PE)—is architecturally positioned to excel in deep packet inspection beyond the scope of classification-focused models (e.g., ET-BERT, YaTC) or generative models based on abstracted representations (e.g., NetGPT). Notably, ByteFlow establishes a strong benchmark for direct, natural language Question Answering (QA) on Packets (Token F1 0.4800, Find-byte EM 0.4700), a critical capability for applied security that democratizes network analysis by allowing personnel without deep protocol expertise to rapidly query and understand complex network events. This, combined with robust byte-level understanding (BMLM: 0.5924) and accurate parsing (JSON Acc: 0.8125), results in manifold security benefits, including enhanced threat detection against evasion techniques, accelerated incident response by reducing Mean Time To Respond (MTTR), and an overall improved Security Operations Center (SOC) workflow through an intuitive interface for complex data (Table II & III).

## VI. COMPARISON WITH TEXT-BASED LLMs

To rigorously validate the efficiency and structural benefits of ByteFlow’s direct byte-level processing and 2D-PE, we conducted a head-to-head comparison against open-source, text-based Large Language Models of a similar size. This experiment isolates the impact of the input format: raw bytes (ByteFlow) versus a text representation of the bytes (competitors).

### A. Experimental Setup

- **Competitor Models:** We selected two open-source, decoder-only and encoder-decoder models: **SmolLM-2**

(300M parameters) and **T5-small** (60M parameters). Both were pre-trained on large text corpora and then fine-tuned on our five downstream network intelligence tasks (Phase 2), mirroring ByteFlow’s fine-tuning setup.

- **Input Conversion:** For the competitor models, the raw PCAP byte sequences were first converted to a text JSON representation. This conversion inflates the input sequence length by  $\approx 500\%$  compared to raw bytes, but is reduced by using the Byte Pair Encoding or Word Piece Tokenizer of the given model. The input length is still well above the byte sequence.
- **Context Normalization:** To ensure a fair comparison based on the amount of raw data processed, we normalized the effective input context. ByteFlow operates with a 1024-token context. Since the text models’ text input uses approximately more tokens per byte, we reduced their maximum sequence length. For ByteFlow to T5-small comparison, we used a 1024-token limit for T5-small on the input. For ByteFlow to SmolLM-2 comparison, we used a 1024-token limit for SmolLM-2 on the input. This ensures both byte-level and text-level models process the same maximum number of tokens.
- **Fine-Tuning:** Both SmolLM-2 and T5-small were fine-tuned for the same number of steps and using the same loss function and learning rate as ByteFlow’s Phase 2 fine-tuning (LR = 0.00001).

### B. Comparative Results and Analysis

Table IV presents the performance of ByteFlow (Full) against the fine-tuned text models.

TABLE IV  
PERFORMANCE COMPARISON: BYTEFLOW VS. TEXT-BASED LLMs (JSON)

Task	Metric	T5-small	SmolLM-2	ByteFlow
Byte Modeling	BMLM B-Acc	0.0091	0.0215	<b>0.5924</b>
	Next-byte B-Acc	0.4410	0.3632	<b>0.5829</b>
Parse	JSON-Acc	0.8173	0.7528	<b>0.8125</b>
	Field-Acc	0.7305	0.6894	<b>0.6610</b>
QA	Token F1	0.2762	0.2011	<b>0.4800</b>
	Rouge-L	0.1935	0.1170	<b>0.3980</b>
Find-byte	EM	0.0000	0.0000	<b>0.4700</b>

The results demonstrate the advantage of direct byte-level processing combined with domain-specific pre-training over general-purpose text models, even when compensating for sequence space. In a direct comparison, ByteFlow’s foundational task domination is evident, achieving a nearly higher Next-byte Accuracy and a higher BMLM Accuracy than the best text model, SmolLM-2, which confirms that converting raw bytes to text fails to build the essential byte-level and protocol structure understanding that ByteFlow learns in its Phase 1 pre-training. Crucially, both T5-small and SmolLM-2 score 0.0000 on the challenging Find-byte Exact Match task, demonstrating their complete inability to perform targeted information extraction—a core function for security analysts.

In contrast, ByteFlow achieves 0.4700 EM, showing that its byte-level attention and 2D-PE are essential for mapping complex queries to precise byte locations within the packet structure. For higher-level semantic tasks like QA, ByteFlow’s performance remains vastly superior, achieving higher QA Token F1 score than SmolLM-2, which indicates that the loss of fundamental byte-level fidelity due to hex conversion is a severe handicap that cripples a generic transformer’s ability to correctly interpret and generate structured or natural language output about the network data. In conclusion, this comparison validates ByteFlow’s core premise: for deep packet inspection, the ByteFlow framework’s direct processing of raw bytes, enhanced by domain pre-training and structural encodings, is an indispensable requirement for achieving practical network intelligence capabilities, while the simple translation of bytes to text formats is fundamentally insufficient.

### VII. LIMITATIONS AND FUTURE WORK

Despite the promising results of ByteFlow, several limitations define avenues for future work, primarily concerning the depth of question answering (QA)—as the current model excels at simple extraction but is unproven for complex, security-centric inferential reasoning across multi-packet flows, compounded by the lack of standardized, large-scale public benchmark datasets. Furthermore, while ByteFlow handles raw bytes, its current utility for encrypted traffic is limited to metadata analysis, necessitating future integration with decryption proxies or dedicated Encrypted Traffic Analysis (ETA) methods for deep payload inspection. In terms of performance and scale, the computational intensity of Large Language Models and the long sequences inherent to byte-level processing pose a significant hurdle for real-time, line-rate analysis in high-throughput environments. For security deployment, a rigorous evaluation of adversarial robustness against evasion techniques is critical. Finally, the current framework primarily focuses on single flows and lacks a dedicated mechanism for inter-flow correlation, which we aim to address by exploring advanced representations, such as a Three-Dimensional (3D) Positional Encoding (Flows, Packets, Bytes/Fields), to model crucial temporal and spatial dependencies across the network for detecting complex, distributed threats.

### VIII. CONCLUSION

We introduced **ByteFlow**, a novel framework leveraging a byte-level Large Language Model (ByT5-small) for deep packet inspection and versatile network intelligence directly on raw byte streams, integrating a tailored tokenization scheme, a two-phased training strategy with domain-specific pre-training, and a 2D Positional Encoding (2D-PE) mechanism to capture packet structure. Our evaluation showed that domain-specific pre-training is crucial for foundational understanding, and that 2D-PE dramatically enhances structure-aware and semantic tasks, more than doubling accuracy on the Find-byte task and achieving a  $\sim 76\%$  relative increase in QA F1 scores over the base pre-trained model. These capabilities directly translate to enhanced security operations by enabling faster incident

response through natural language querying and more accurate automated extraction of Indicators of Compromise (IoCs). While ByteFlow pioneers Question Answering (QA) for raw packets, future work must address limitations such as handling encrypted traffic content, scaling to real-time speeds, and modeling complex inter-flow correlations, potentially through 3D positional encoding.

#### IX. ACKNOWLEDGEMENTS

Part of this work was funded by the Department of Education under grant P116Z230151, and by the National Science Foundation under grants CNS-2210091 and CNS-2515265. The authors thank the Rivas.AI Lab (<https://lab.rivas.ai>) for the support and helpful feedback throughout this project.

## REFERENCES

- [1] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.
- [2] Z. Li, Y. Liu, C. Zhang, W. Shan, H. Zhang, and X. Zhu, "Trustworthy deep learning for encrypted traffic classification," *Soft Computing*, pp. 1–18, 2025.
- [3] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.
- [4] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2017.
- [5] X. Meng, Y. Wang, R. Ma, H. Luo, X. Li, and Y. Zhang, "Packet representation learning for traffic classification," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3546–3554, 2022.
- [6] X. Yun, Y. Wang, Y. Zhang, and Y. Zhou, "A semantics-aware approach to the automated network protocol identification," *IEEE/ACM transactions on networking*, vol. 24, no. 1, pp. 583–595, 2015.
- [7] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE communications magazine*, vol. 57, no. 5, pp. 76–81, 2019.
- [8] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [9] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proceedings of the ACM Web Conference 2022*, pp. 633–642, 2022.
- [10] Y. Liang, P. Li, Q. Lai, and J. Wen, "Em-bert: A language model based method to detect encrypted malicious network traffic," in *International Conference on Image, Vision and Intelligent Systems*, pp. 580–589, Springer, 2023.
- [11] J. Luo, Z. Chen, W. Chen, H. Lu, and F. Lyu, "A study on the application of the 15 large language model in encrypted traffic classification," *Peer-to-Peer Networking and Applications*, vol. 18, no. 1, pp. 1–13, 2025.
- [12] J. Qu, X. Ma, and J. Li, "TrafficGPT: Breaking the token barrier for efficient long traffic analysis and generation," *arXiv preprint arXiv:2403.05822*, 2024.
- [13] K. B. Kan, H. Mun, G. Cao, and Y. Lee, "Mobile-llama: Instruction fine-tuning open-source llm for network analysis in 5g networks," *IEEE Network*, 2024.
- [14] X. Meng, C. Lin, Y. Wang, and Y. Zhang, "Netgpt: Generative pretrained transformer for network traffic," *arXiv preprint arXiv:2304.09513*, 2023.
- [15] Q. Wang, C. Qian, X. Li, Z. Yao, H. Shao, and R. Zhang, "Lens: A foundation model for network traffic," *arXiv preprint arXiv:2402.03646*, 2024.
- [16] G. Zhou, X. Guo, Z. Liu, T. Li, Q. Li, and K. Xu, "Trafficformer: An efficient pre-trained model for traffic data," in *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 1844–1860, 2025.
- [17] T. Cui, X. Lin, S. Li, M. Chen, Q. Yin, Q. Li, and K. Xu, "Trafficllm: Enhancing large language models for network traffic analysis with generic traffic representation," 2025.
- [18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, Curran Associates, Inc., 2017.
- [19] T. Shapira and Y. Shavitt, "Flowpic: A generic representation for encrypted traffic classification and applications identification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1218–1232, 2021.
- [20] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017.
- [21] X. Xiao, W. Xiao, R. Li, X. Luo, H. Zheng, and S. Xia, "Ebsnn: Extended byte segment neural network for network traffic classification," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3521–3538, 2021.
- [22] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024, 2011.
- [23] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mT5: A massively multilingual pre-trained text-to-text transformer," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 483–498, Association for Computational Linguistics, 2021.
- [24] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel, "ByT5: Towards a token-free future with pre-trained byte-to-byte models," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 291–306, 2022.
- [25] J. Belouadi and S. Eger, "Bygpt5: End-to-end style-conditioned poetry generation with token-free language models," *arXiv preprint arXiv:2212.10474*, 2022.
- [26] A. Pagnoni, R. Pasunuru, P. Rodriguez, J. Nguyen, B. Muller, M. Li, C. Zhou, L. Yu, J. Weston, L. Zettlemoyer, *et al.*, "Byte latent transformer: Patches scale better than tokens," *arXiv preprint arXiv:2412.09871*, 2024.
- [27] Y. Yu, H. Yan, H. Guan, and H. Zhou, "Deephttp: semantics-structure model with attention for anomalous http traffic detection and pattern mining," *arXiv preprint arXiv:1810.12751*, 2018.
- [28] M. R. Shahid, G. Blanc, H. Jmila, Z. Zhang, and H. Debar, "Generative deep learning for internet of things network traffic generation," in *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 70–79, IEEE, 2020.
- [29] T. Cui, X. Lin, S. Li, M. Chen, Q. Yin, Q. Li, and K. Xu, "Trafficllm: Enhancing large language models for network traffic analysis with generic traffic representation," *arXiv preprint arXiv:2504.04222*, 2025.
- [30] R. Zhao, M. Zhan, X. Deng, Y. Wang, Y. Wang, G. Gui, and Z. Xue, "Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Washington, DC, USA, February 7-14, 2023*, pp. 5420–5427, AAAI Press, 2023.
- [31] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 253–262, SciTePress, 2017.
- [32] W. Wang and Y. Lu, "USTC-TFC2016 Dataset." <https://github.com/yungshenglu/USTC-TFC2016>, 2016. Accessed: May 6, 2025.
- [33] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 407–414, SciTePress, 2016.
- [34] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. H. Lashkari, "Detection of DoH tunnels using time-series classification of encrypted traffic," in *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 63–70, IEEE, 2020.
- [35] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K.-N. Truong, and A. A. Ghorbani, "Towards the development of a realistic multidimensional IoT profiling dataset," in *2022 IEEE International Conference on Privacy, Security and Trust (PST)*, pp. 1–10, IEEE, 2022.

## APPENDIX

### A. Fine-Tuning Task Dataset Generation

The datasets for the five fine-tuning tasks were curated from our comprehensive PCAP data corpus, which consists of approximately 189 gigabytes of traffic from 3,771,804 PCAP flow files. Each task's dataset was constructed by processing individual packets or sequences of packets (chunks) based on specific criteria and constraints related to packet length and structure. The generation process and approximate scale for each task dataset are detailed below:

- **Byte Masked Language Modeling (BMLM) on PCAP Data:** For each flow a BMLM task was formulated. Approximately 15% of its byte tokens (excluding special

structural tokens) were randomly selected for masking. This procedure yielded approximately 3.7 million BMLM task instances.

- **Autoregressive Next-Byte Prediction on PCAP Data:** Next-byte prediction tasks were generated from packets whose lengths were between 10 and 1000 bytes. For each qualifying packet, a random truncation point (ensuring at least one byte preceded it) was selected. The input to the model was the sequence of byte tokens leading up to this truncation point, and the objective was to predict the single original byte token that immediately followed. This process resulted in approximately 3 million next-byte prediction instances.
- **Supervised Packet Parsing (Wireshark JSON Prediction):** This task aimed to translate raw packet data into a structured format. For individual packets up to 500 bytes in length, we generated corresponding ground truth JSON representations. These were obtained by executing the `tshark` utility (part of the Wireshark suite) with the command options `-r <temp_pcap> -T json -x` on a temporary PCAP file containing the single raw packet. Each task instance thus paired a raw packet byte sequence with its detailed `tshark`-generated JSON output. Approximately 500 thousand such parsing tasks were created.
- **Question Answering (QA) on Packets:** QA tasks were formulated for packets not exceeding 800 bytes in length. For each suitable packet, multiple QA pairs were generated based on a predefined set of 50 distinct questions covering a wide array of packet attributes (sample questions are provided in Appendix B). The answers to these questions were programmatically derived by parsing the packet data, ensuring the factual correctness of the ground truth. As each eligible packet could generate 50 QA instances, this resulted in a dataset of approximately 100 million QA pairs.
- **Field Finding in a Specific Packet:** This task focused on targeted information retrieval from packet sequences. We processed PCAP data in chunks of up to 10 packets, requiring a minimum of 3 packets per chunk and ensuring the total byte length of the chunk did not exceed 940 bytes. For each packet within such an eligible chunk, a field-finding task was created. This involved randomly selecting a field name from a predefined list of common packet header fields (e.g., 'TCP window size', 'Source IP Address', 'EtherType'). The model was provided with the tokenized chunk, an identifier for the specific target packet within that chunk, and the name of the field, with the objective of extracting the exact value of that field. Ground truth values were programmatically extracted. This methodology yielded approximately 22.9 million field-finding instances.

#### B. Sample Questions for QA on Packets Task

Below are the 50 sample questions used for the Question Answering (QA) on Packets task:

- 1) What is the source MAC address?
- 2) What is the destination MAC address?
- 3) What is the EtherType of this frame?
- 4) Is this an IPv4 or IPv6 packet?
- 5) What is the source IP address?
- 6) What is the destination IP address?
- 7) What is the IP protocol number?
- 8) What is the Time To Live (TTL) value for this IP packet?
- 9) What is the Differentiated Services Code Point (DSCP) value?
- 10) What is the total length of the IP packet?
- 11) What is the IP header length?
- 12) Are IP flags set for 'Don't Fragment'?
- 13) Are IP flags set for 'More Fragments'?
- 14) What is the fragment offset for this IP packet?
- 15) Is this packet a TCP segment?
- 16) Is this packet a UDP datagram?
- 17) What is the TCP source port?
- 18) What is the TCP destination port?
- 19) What is the UDP source port?
- 20) What is the UDP destination port?
- 21) What is the TCP sequence number?
- 22) What is the TCP acknowledgment number?
- 23) Is the TCP SYN flag set?
- 24) Is the TCP ACK flag set?
- 25) Is the TCP FIN flag set?
- 26) Is the TCP RST flag set?
- 27) Is the TCP PSH flag set?
- 28) Is the TCP URG flag set?
- 29) What is the TCP window size?
- 30) What is the UDP length?
- 31) What is the checksum value for the transport layer?
- 32) Does this packet contain a DNS query?
- 33) Does this packet contain a DNS response?
- 34) What is the DNS transaction ID?
- 35) What is the DNS query type (e.g., A, AAAA, MX)?
- 36) What is the domain name being queried in DNS?
- 37) Does this packet contain an HTTP request?
- 38) Does this packet contain an HTTP response?
- 39) What is the HTTP request method (e.g., GET, POST)?
- 40) What is the HTTP request URI?
- 41) What is the HTTP Host header value?
- 42) What is the HTTP User-Agent header value?
- 43) What is the HTTP Content-Type header value?
- 44) What is the HTTP status code?
- 45) Is this packet part of a TLS handshake?
- 46) What is the TLS handshake type (e.g., Client Hello, Server Hello)?
- 47) What is the TLS version proposed or used?
- 48) What is the captured length of this frame?
- 49) What is the time delta from the previously captured packet in this context?
- 50) Which protocol is encapsulated directly within the IP layer?