A Comparative Study of Backbone Architectures for Language Model-Based Intrusion Detection

Benedikt Pletzer and Jürgen Mottok
Laboratory for Safe and Secure Systems (LaS³)
OTH Regensburg
Regensburg, Germany
email: {benedikt.pletzer, juergen.mottok}@oth-regensburg.de

***Abstract*—Network based Intrusion Detection Systems (NIDS) have recently been shown to benefit from techniques developed for Natural Language Processing (NLP). Specifically, pretrained models based upon the ubiquitous Transformer backbone architecture have been shown to outperform other approaches. In recent months, promising research aimed at improving the aforementioned Transformer backbone or even replacing it all together has been published. This includes low bit quantization techniques like BitNet, as well as new model types like Mamba. This study, therefore, evaluates the potential of emerging foundation models, such as BitNet and Mamba, as backbones for NIDS. For this purpose, a comparative study of these models as backbone of an otherwise unchanged Language Model (LM) based NIDS algorithm is performed. Our results indicate that Mamba outperforms all other models in terms of classification performance, as well as in inference latency, if Graphics Processing Unit (GPU) acceleration is available. We also establish that low-bit-quantized models are able to achieve good classification accuracies, making them an auspicious option if their potential in computational efficiency are reached.**

Keywords- IDS*; *Transformer*; *BitNet*; *Mamba*; *MatMul-Free LM.

I. Introduction

In times of ever-increasing cybersecurity threats, emanating from state-sponsored entities, as well as criminal groups motivated by profit, countermeasures are in high demand. One of these countermeasures are Network based Intrusion Detection Systems (NIDS). Most NIDS currently deployed in industry use pattern matching methods to compare past attacks in order to detect them and notify system administrators when an attack occurs. These systems, however, struggle with detecting new or evolving threats, making them vulnerable to zero-day exploits or attackers changing their approach to avoid detection. Therefore, various machine learning based approaches to NIDS are researched at present. In [1], Ferrag et al. propose a NIDS that borrows the concepts currently applied in natural language processing to analyze network traffic. The authors report that their approach, based on Language Models (LM), outperforms the state of the art in attack classification accuracy. Aside from its performance in detection and classification, there are other potential upsides of using language modeling techniques in NIDS. These include the semantic interpretation of threat detections and the generation of threat responses in natural language [1][2]. Ferrag et al. utilize the Bidirectional Encoder Representations from Transformers (BERT) [3] Transformer as the backbone of their intrusion detection model. BERT, which has been proposed in 2018, shows strong performance in sequence classification tasks, but a lot of research has been done to improve on the standard Transformer architecture since then. This poses the question whether substituting the BERT backbone of the IDS model with a more modern one could lead to improvements either in classification accuracy, inference, speed or memory consumption. In recent months, various new model architectures and quantization techniques have been proposed to either replace the Transformer or improve it in key characteristics. This publication, therefore, aims to provide a summary of recent advancements in language modeling and to examine the validity of these new architectures and techniques in the field of NIDS. The following four architectures are chosen for this study:

- Transformer++ [4], [5]
- BitNet [6], [7]
- Mamba [8], [9]
- MatMul-Free LM [10]

Transformer++ serves as baseline, representing the state-of-the-art Transformer model. This publication provides reference points of existing implementation of the aforementioned models. Because some of these techniques can not yet meet their full potential, i.e., due to unoptimized implementations in software or the lack of dedicated hardware, the comparison of actually realized results is followed by a discussion of untapped potential. The contributions of this publication can be summarized as follows:

C1: A summary of recent advancements regarding LM backbones is given.
C2: The performance of these backbones in a NIDS task are studied comparatively
C3: The (to our knowledge) first NIDS models using BitNet, and MatMul-Free LM are presented

These three contribution aim to aid with answering the question: Which backbone architecture is most promising for the development of language model based NIDS?

This work is structured into a review of current challenges and recent advances regarding the backbones of language models, aiming to improve Transformers or replace them with more capable models, see Section II. Following this, Section III provides a description of the methodology used to train our models and evaluate them in regard to classification

ISBN: 978-1-68558-206-7

performance and inference speed. In Section IV the results are stated, once again subdivided into classification performance and inference speed. The results are then discussed in Section V, followed by a conclusion and the proposal of future work regarding LM based NIDS and LMs in cybersecurity in general in the final Section VI.

II. Related Work

This section will summarize some chosen advances in language modeling that have been published in recent months. The following paragraphs aim to inform about the basic concepts that drive the recent surge of change in foundational language model architecture. In our view, there are two foundational concepts that drive the publications described in this Section. On the one hand there is the low-bit quantization, on the other there is the emergence of parallelizable recurrent architectures that blend the properties of classical Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) [8]. These concepts will now be examined, exploring their implementation in recent models and their potential impact on the field of language modeling.

As language models are being released to the public and used by a wide range of individuals privately, as well as professionally, the cost associated with their inference becomes an increasingly urgent topic. Reducing this cost is therefore a central topic of current language model research. One approach to this topic is reducing the cost of inference by using less resource intensive operations within an otherwise unchanged Transformer model. This process, called quantization, has been standard practice for years. Customary quantization techniques involve replacing the floating point values (for example 16 bit precision floating point) used as weights or activation with integer values, e.g., 8-bit integers. This achieves a reduction in computational cost, and therefore energy consumption, as well as a reduction in consumed memory. Tao et al. show in [11] that low-bit quantized models can maintain good performance while reducing cost, provided quantization-aware training is done right.

The idea of increasing inference efficiency by applying low-bit quantization to the tried and tested Transformer architecture has recently been advanced by H. Wang and S. Ma et al. by proposing BitNet in [7] and further improved in [6]. BitNet uses the same basic building blocks as standard Transformer networks, while replacing its linear feedforward components with the BitLinear layer. This layer uses 1-bit binarized weights, quantizing every element in the weight matrix to either one or minus one, while quantizing the activation to 8-bit. Multiplication of these weight matrices with input vectors can therefore be performed using addition and subtraction only. These changes are not applied outside the Transformer block, meaning input and output components of the network remain unchanged. Wang and Ma show in [6] that the 1 bit-quantized BitNet can produce competitive results with regular Transformers, while significantly improving memory consumption and computational cost of inference.

S. Ma and H. Wang et al. further improve upon the aforementioned findings in [6] proposing BitNet 1.58, a BitNet variant that utilizes ternary instead of binary weights. This extends the values in the weight matrices from $\{-1, 1\}$ to $\{-1, 0, 1\}$. This change increases the information entropy of each weight matrix entry to 1.58 bits, hence the name BitNet 1.58, while maintaining BitNet's main advantage, the omission of multiplication operations. The authors state the ability to filter features as an additional advantage of adding zeros to the weight values. BitNet 1.58 is therefore more expressive than BitNet without additional computational cost.

The previously mentioned Transformer model [12] with Attention [13] as its foundational mechanism has been the dominating network architecture for sequence to sequence tasks in recent years. This success is a consequence of its ability to encode complex relations within a set reference window while maintaining a dense representation of information within its weights and computational efficiency in inference, as well as in training. The Transformer architecture, however, is not without its drawbacks. One of them is the fixed maximum length of its reference window. This drawback is further aggravated by the fact that the computational cost attached to the Transformer scales quadratically with the length of its reference window. A Transformers' ability to extract information from longer sequences is therefore limited, especially in resource-constrained environments.

Another heavily researched area of improvements to language models is the quest for a replacement for the Transformer architecture that remedies the shortcomings of the latter while maintaining the abilities that made Transformers successful. One of the most promising of these alternatives has been proposed by A. Gu and T. Dao in [9]. The authors present Mamba, a variant of State Space Models that scales almost linearly with sequence length while also showing an ability to reason on long sequences that rivals that of Transformer Networks. This model architecture also fulfills the requirement for efficient computations in training and inference because it can be computed either as linear recurrence (for inference) or global convolution (for highly parallelized training) [9]. A NIDS model based on the mamba is described in [14] by Wang et al.

Zhu et al. combine ideas from the previously described papers in [10] to create a language model that eliminates the need for matrix multiplication in inference completely. They achieve this by adapting the ternary BitLinear layer from [6] and combining this with the fundamental concept of Mamba, i.e., replacing the Transformer architecture with a parallelizable recurrent net. Thus avoiding the fundamental scaling issue of attention layers. In addition to their conceptual contribution, they also describe an optimized variant of the BitLinear layer that fuses the RMS (Root Mean Square) Norm with the activation function to be executed as a single block in the faster Static Random-Access Memory (SRAM) of GPUs reducing latency and memory consumption in comparison to the original BitLinear implementation.

Following this summary of chosen publications promising to improve the current state of language models (**C1**), a description of the methodology used to determine the potential of the aforementioned techniques is given.

III. Test Setup and Methodology

This publication aims to provide a comparison between the aforementioned innovations with the language model like NIDS algorithms in mind. The two most significant criteria in this field are, according to the authors' opinion, the classification performance that can be achieved and the computational cost at inference needed to do so. The former is the most commonly described using metrics like precision, recall and F1-Score as shown in surveys regarding the topic [15][16]. While the latter is especially crucial in application areas with constrained computational resources like the Internet of Things (IoT) [17]. This section is, therefore, structured into a description of the training setup and the classification task used to benchmark model accuracy, and the setup used to measure inference speed.

A. Training and Classification

The dataset used for our tests is the Edge-IIoTset published by Ferrag et al. in [18]. We employ the readily extracted features provided in the CSV format. Due to alignment issues, the rows with the attack labels Man In The Middle (MITM) & Distributed Denial Of Service for User Datagram Protocol (DDOS-UDP) are removed from the dataset, reducing the number of labels, including normal, from fifteen to thirteen. This dataset is chosen to facilitate comparisons with similar, previously published models.

The structure of the NIDS algorithm, as well as the implemented training process, are borrowed from [1]. The model consists of an embedding layer that gets tokens as input, followed by a sequence to sequence model, that is referred to as backbone in this paper. The output of the sequence to sequence model is fed into two different Multi Layer Perceptron (MLP) heads. The first one is a Masked Language Model (MLM) head with an output shape fitting to reproduce the input tokens. The other one is a 13 class softmax head used for attack type classification. Figure 1 shows the previously described structure.

The training process is accordingly split into two stages. The first one, in the following referred to as pretraining, is an unsupervised training stage in which the network is fed with sequences of partially masked tokens from the training set. That means that 15 percent of the tokens are replaced with a mask token and the network is tasked with reproducing the original unmasked sequence. This allows the network to learn inner dependencies of the dataset and thus gain an understanding on how certain tokens relate to each other. In the second training step, the MLM head is replaced with a classification head and trained to classify the attack label associated with the last package in the input sequence in a supervised manner. To combat the heavily biased dataset, random oversampling is deployed, resulting in better classification results for classes with less support. Both training stages take sequences of tokens as input. The process of obtaining those tokens is the Privacy-Preserving Fixed-Length Encoding (PPFLE) algorithm described in [1]. Each entry in the CSV table is concatenated with the associated column name. The resulting string is then hashed, creating a fixed length string for each column name value combination in each line of the CSV. These are then converted into tokens using the Byte-level-Byte-Pair-Encoding.

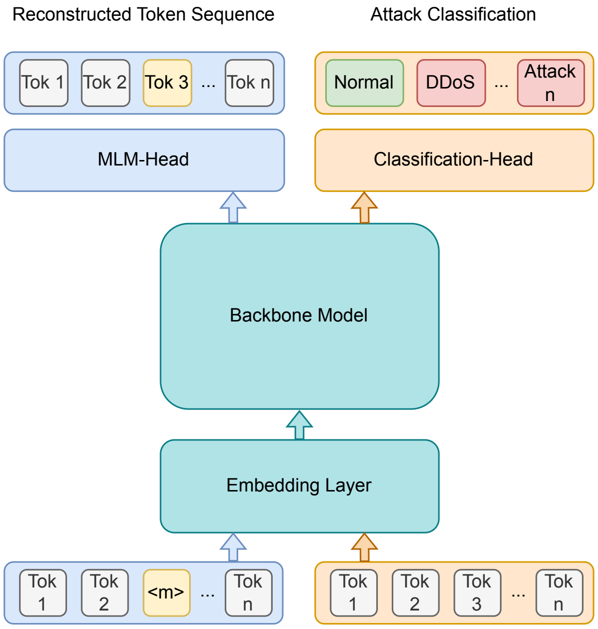


Figure 1. Structure of the Network in pretraining (left) and finetuning (right).

The dataset is split into train, test, and validation splits with the respective sizes of 1272242, 381700, and 163600 entries each. The test split is used during training to select the most capable models, while the validation set is used to perform a final assessment of the models' performance, creating the numbers presented in Section IV. Each of the training stages consists of 1200 batches with a batch size of 256. Each element of a batch consists of a sequence of 900 tokens. The learning rate follows a sinus shape overlaid with an exponential decay. This training regime is chosen because it delivers good results with all tested backbone architectures.

Tuning the hyperparameters of each model is done by choosing smaller network sizes as initial reference points. The parameters of the network are later tweaked to enlarge the networks' parameter count. The variations in hyperparameters that result in noticeable improvement in classification performance are kept, and the process is repeated. The enlargement of the networks is stopped when improvements in classification performance are negligible. This approach is derived from LM scaling laws that suggest that a model's performance will increase with increasing parameter count as it's ability to encode complex information increases. For fixed data sizes, this trend can be expected to continue until a threshold based on the available data is reached and a phase of diminishing returns begins. We try to detect this threshold and determine the optimal model size based on the point of diminishing returns.

The backbones included in this comparative study are selected based on the Related Work presented in Section II. The baseline of this comparison will be a standard implementation of a Transformer network, including the improvements described in Transformer++ [4]. The specific implementation of this backbone is taken from the repository published with Meta's LLaMA [5]. Transformer++/LLaMA is commonly used as baseline in other publications, i.e., [6], [7], [9].

For the implementation of the low-bit quantization proposed in [6], [7] the process described in [7] is followed. Therefore, the LLaMA model used as baseline is adapted by interchanging the linear layers within the attention and the feed-forward blocks of the Transformer model with BitLinear layers. As BitLinear layer, the ternary variant described in [6] is used, as it outperforms or ties the binary variant in all regards except for information density in the weight matrices. The implementation of the BitLinear layer is taken from [10], as it is more efficient and already part of the project.

For Mamba, two different implementations are used. The first one is provided by the authors and contains a computationally efficient selective-scan written in CUDA. This, however, limits the algorithm to be used on systems that have GPU acceleration available. In order to also perform tests using CPU, a PyTorch only implementation is used [19].

The implementation for the final backbone architecture examined in this study is provided by Zhu et al., the authors of the paper describing it [10].

B. Inference Benchmarking

The inference speed test runs with CUDA GPU are executed on a virtual machine hosted on a laboratory server. The virtual machine has the following specifications:

- 12 core Intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz CPU
- NVIDIA A5000 with 24GB GDDR6 memory
- 128GB system memory

All models are implemented in PyTorch and executed using version 2.2.1 in half precision mode.

The implementation of Mamba provided by A. Gu and T. Dao utilizes a memory optimization that reduces copies between the slower HBM (High-Bandwidth-Memory) and the faster but smaller SRAM of GPUs. This optimization speeds up training and inference but requires access to a GPU, preventing the model from being inferred on CPUs. As many NIDS applications rely on low-cost devices, that usually do not have access to a GPU, to perform the classification of network data we deploy a pure python implementation to compare CPU inference latencies of Mamba and Transformer++. These tests are performed on the virtual machine described above but without access to the A5000 GPU. BitNet and MatMul-Free LM are omitted in this test the reasoning for this decision is given in Section IV-B.

IV. Results

This section describes results obtained using the setups described in the previous section. The results are, like the test-setup description, structured into classification performance and inference speed.

A. Classification Results

A collection of metrics comparing the classification performance of the models trained as described in the previous section is shown in Table I. For a more visual comparison, selected key metrics are shown in the radar chart depicted in Figure 2.

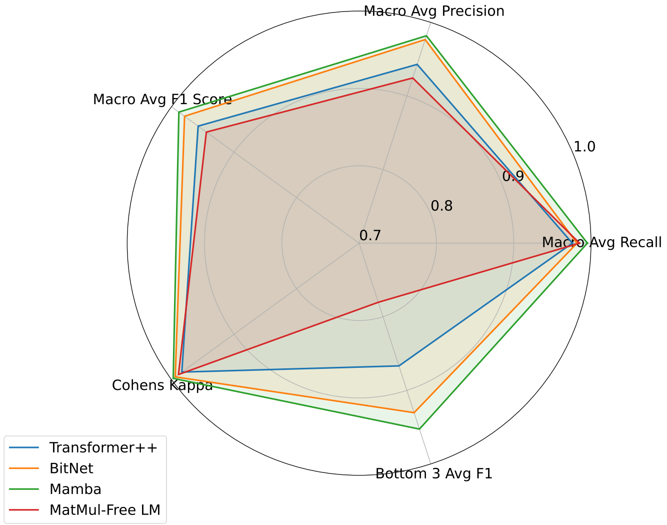


Figure 2. Radar chart showing key classification metrics.

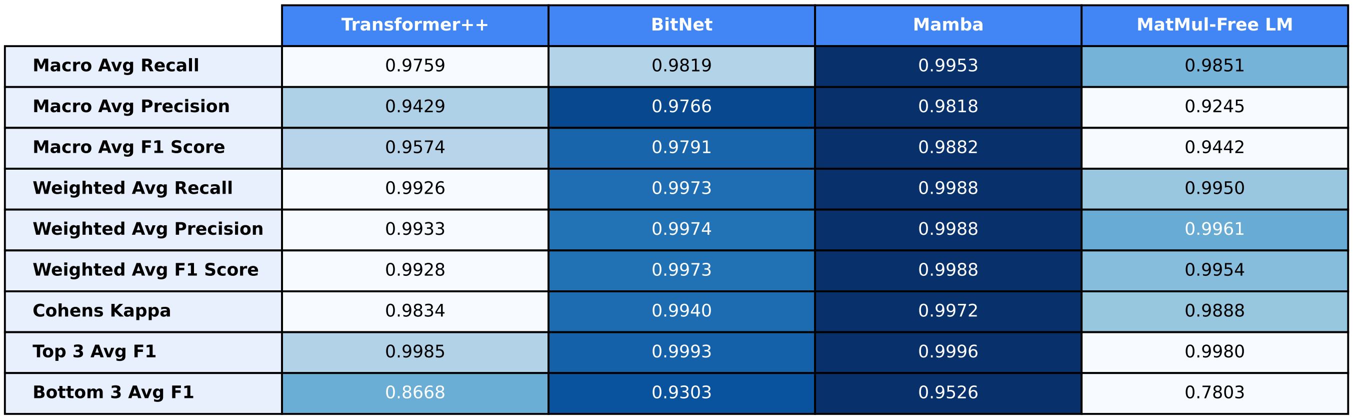
Both charts show that the model using Mamba as backbone performs the best in all chosen metrics. BitNet is second in all metrics except for Macro Avg Recall, in which it comes third. The standard Transformer++ implementation and MatMul-Free LM take last place in all selected metrics. BitNet outperforming Transformer++ is notable as it only differs from Transformer++ by using BitLinear layers instead of 16 bit floating point linear layers. This change is primarily done to reduce the resources necessary to infer it. Comparing the results with the ones published by Ferrag et al. in [1] shows that all backbone models outperform the implementation using BERT as backbone. But it has to be noted that the training process and dataset are not identical to the ones used by Ferrag et al. as for example the sequence window is longer and, as described in Section III-A, two of the classes were removed from our dataset.

B. Inference Speed

Figure 3 shows a violin plot of each networks' inference latency on an Nvidia A5000 GPU using CUDA. The plot shows that Mamba beats the baseline set by Transformer++ by a wide margin. The latency of Mamba is 34 percent lower than the one of Transformer++ despite outperforming it in terms of classification, which makes Mamba the fastest as well as the most capable model in this comparison.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

TABLE I
TABULAR OVERVIEW OF THE CLASSIFICATION PERFORMANCE



	Transformer++	BitNet	Mamba	MatMul-Free LM
Macro Avg Recall	0.9759	0.9819	0.9953	0.9851
Macro Avg Precision	0.9429	0.9766	0.9818	0.9245
Macro Avg F1 Score	0.9574	0.9791	0.9882	0.9442
Weighted Avg Recall	0.9926	0.9973	0.9988	0.9950
Weighted Avg Precision	0.9933	0.9974	0.9988	0.9961
Weighted Avg F1 Score	0.9928	0.9973	0.9988	0.9954
Cohens Kappa	0.9834	0.9940	0.9972	0.9888
Top 3 Avg F1	0.9985	0.9993	0.9996	0.9980
Bottom 3 Avg F1	0.8668	0.9303	0.9526	0.7803

The two quantized models, BitNet and MatMul-Free LM, on the other hand, have inference latencies that are larger than Transformer++'s by a factor of 6 and 4, respectively. These results can be explained by the method used to implement the quantized models in PyTorch, which is further discussed in the next section.

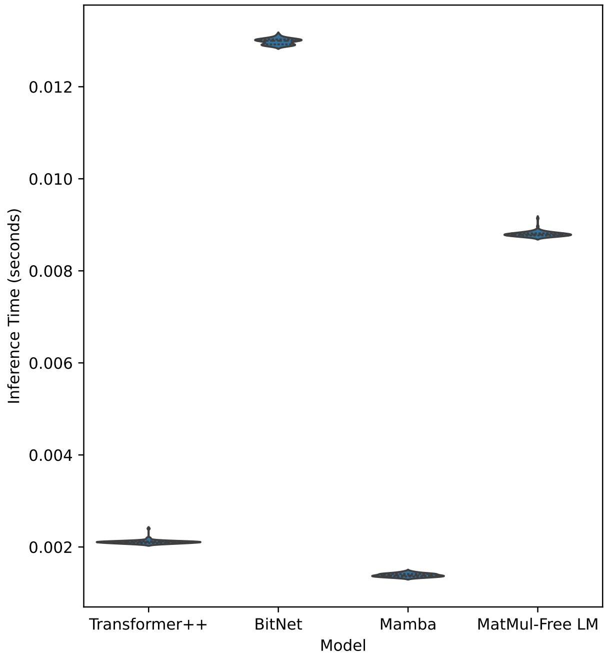


Figure 3. Inference latency on GPU.

Figure 4 shows the violin plots for the CPU inference tests. BitNet and MatMul-Free LM are not included in this test, as the GPU tests have already shown that their implementations are not competitive at the moment. This would not change on CPUs as the underlying cause of it, which is discussed in Section V-B, is the same. The plot shows that the advantage Mamba has over Transformer++ on GPU does not carry over to CPUs.

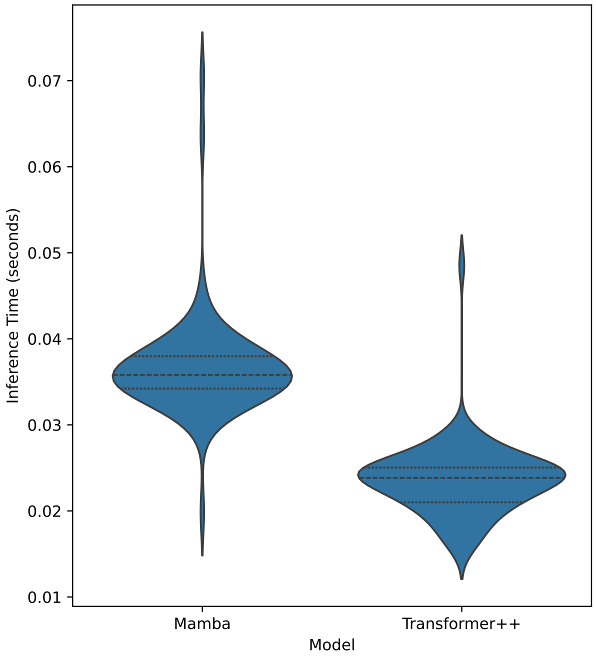


Figure 4. Inference latency without GPU acceleration, using CPU.

V. DISCUSSION

This section discusses the results presented in Section IV. In parallel to the preceding section, it is also subdivided into

Copyright (c) IARIA, 2024. ISBN: 978-1-68558-206-7

classification performance and inference latency.

A. Discussion of Classification Performance

The fact that all the models trained for the purpose of this study outperform the model described in [1] should not be solely attributed to the superiority of the more modern backbones used in our models. Attributing this difference to the attack classes removed in our tests seems natural, but does not withstand closer examination. Both classes (MITM & DDOS-UDP) are classified flawlessly by Ferrag et al.'s model. The main differentiator between our models and the reference is the classification performance for the fingerprinting class (F1 score in [1] 0 and .92 in our Mamba model). This discrepancy might be explained by omitting the separation of network traffic into flows. The removal of this step allows networks to draw connections between suspicious packages sent to and from different devices in close temporal proximity. The fingerprinting attack comprised in the Edge-IIoTset [18] aims at identifying the operating systems of potential victims in the Network. The attack therefore likely contains similar messages to different Hosts in the network in short timespans. Thus making detection easier if inputs are not restricted to one flow. The added capability of models creating an understanding of what happens in different parts of a network in combination with the sub-quadratic scaling of models like Mamba in regard to input sequence length suggests that non-flow-based NIDS could be a better choice going forward.

Determining why BitNet outperforms Transformer++ in our tests requires further experimentation. Besides quantization, the most notable difference between BitLinear and nn.Linear layers are the additional RMS Norm added before activation quantization. This added step, originally done for the sake of numerical stability, might contribute to this otherwise counterintuitive finding.

B. Discussion of Inference Latencies

The underwhelming results of BitNet and MatMul-Free LM can be explained by the fact that their default implementation use a technique called fake-quantization. This means that they use standard floating point values that are clipped during inference to behave like ternary/8-bit quantized operation. This is done because it is, due to limitations of the PyTorch library, the most efficient approach. using currently available hard and software. The usage of floating-point operations, however, cancels out the theoretical gains in computational load during inference, and memory requirements. Clipping these floats to get quantized values even introduces additional operations, leading to overall worse performance. In order to harness the full potential, these techniques offer, specialized inference frameworks like BitBlas or even better hardware accelerators tailored to these operations would be necessary.

Zhu et al. describe two approaches to dealing with this problem. One of them is the aforementioned BitBlas library used to generate their performance latency benchmarks, showing a reduction of latency by a factor of 3.65. The fused BitLinear implementation used for these tests are at the moment not publicly available. These results can therefore not be reproduced for this study. It has to be noted that BitBlas is intended for the deployment of large models on powerful hardware. Whether these results are transferable on the much smaller model sizes used in this study is yet to be determined.

The same limitations apply to our BitNet model, as it uses the fused BitLinear layer proposed in [10].

The other approach described in [10] is an implementation on Hardware using Field-Programmable Gate Arrays (FPGA) demonstrating the potential speed up and increase in efficiency that custom hardware tailored to MatMul-Free models would have. Using FPGAs is however out of the scope of this publication.

VI. Conclusion and Future Work

This publication provides a comparison between promising candidates for language model based NIDS backbones. Supposed to give researchers in the field guidance on which backbone to pick for their models. For this purpose, to our knowledge, the first NIDS using BitNet and MatMul-Free LM as backbones are presented (**C3**). A language model based NIDS using Mamba has been proposed by Wang et al. in [14]. Our research confirms their finding that constitutes a capable and efficient backbone for language model NIDS. Making Mamba the foremost choice as backbone if GPU acceleration is available. Our Inference tests on CPU show that basic Transformer++ might still be the best choice if inference on CPU is mandatory (**C2**).

We prove that low-bit quantized backbones can provide good classification performances which makes them an interesting option for low-power or otherwise resource-constrained application, as soon as specialized hardware or better software implementations for inference are available.

Additionally, we report, to our knowledge, the best classification results on the Edge-IIoTset [18] to date, using Mamba as backbone. The other models proposed in this work also outperform the baseline set in [1].

Future work in this direction might include expanding benchmarks by adding different datasets, as well as different representations of network traffic data, i.e., using raw byte data as input for the tokenizer.

Promising directions for future work include the development of better software solutions for low-bit quantized operations. An investigation into the potential upsides of specialized hardware accelerators might unlock new possibilities for the edge deployment of language model based NIDS and language models in general.

Another promising research field is the generation of detailed descriptions of security incidents in natural language. Ferrag et al. propose in [2] a model based on FalconLLM [20] that generates incidence responses using NIDS classification results as prompt. This approach, however, limits the generated incidence responses to general advices on how to counter certain threats. More detailed outputs might be possible if the generating language model has access to information from the hidden states of the NIDS model.

ACKNOWLEDGMENT

The presented work is part of the research project *KRITIS Scalable Safe and Secure Modules* (KRITIS³M), which is funded by the Project Management Jülich (PtJ) and the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under funding code 03EI6089A.

REFERENCES

[1] M. A. Ferrag *et al.*, "Revolutionizing Cyber Threat Detection With Large Language Models: A Privacy-Preserving BERT-Based Lightweight Model for IoT/IIoT Devices", *IEEE Access*, vol. 12, pp. 23 733–23 750, 2024, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3363469.

[2] M. A. Ferrag *et al.*, "Revolutionizing Cyber Threat Detection with Large Language Models", 2023. DOI: 10.48550/ARXIV.2306.14263.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding", in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[4] P. Thapak and P. Hore, *Transformer++*, Mar. 2020. arXiv: 2003.04974 [cs, stat].

[5] H. Touvron *et al.*, *LLaMA: Open and Efficient Foundation Language Models*, Feb. 2023. arXiv: 2302.13971 [cs].

[6] S. Ma *et al.*, *The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits*, Feb. 2024. arXiv: 2402.17764 [cs].

[7] H. Wang *et al.*, *BitNet: Scaling 1-bit Transformers for Large Language Models*, Oct. 2023. arXiv: 2310.11453 [cs].

[8] T. Dao and A. Gu, *Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality*, May 2024. arXiv: 2405.21060 [cs].

[9] A. Gu and T. Dao, *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*, Dec. 2023. arXiv: 2312.00752 [cs].

[10] R.-J. Zhu *et al.*, *Scalable MatMul-free Language Modeling*, Jun. 2024. arXiv: 2406.02528 [cs].

[11] C. Tao *et al.*, "Compression of Generative Pre-trained Language Models via Quantization", in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Dublin, Ireland: Association for Computational Linguistics, 2022, pp. 4821–4836. DOI: 10.18653/v1/2022.acl-long.331.

[12] A. Vaswani *et al.*, "Attention is all you need", in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[13] D. Bahdanau, K. Cho, and Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, May 2016. arXiv: 1409.0473 [cs, stat].

[14] T. Wang *et al.*, *NetMamba: Efficient Network Traffic Classification via Pre-training Unidirectional Mamba*, May 2024. arXiv: 2405.11449 [cs].

[15] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. Khan, "Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review", *Procedia Computer Science*, vol. 171, pp. 1251–1260, 2020, ISSN: 18770509. DOI: 10.1016/j.procs.2020.04.133.

[16] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches", *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, Jan. 2021, ISSN: 2161-3915, 2161-3915. DOI: 10.1002/ett.4150.

[17] A. Heidari and M. A. Jabraeil Jamali, "Internet of Things intrusion detection systems: A comprehensive review and future directions", *Cluster Computing*, Oct. 2022, ISSN: 1386-7857, 1573-7543. DOI: 10.1007/s10586-022-03776-z.

[18] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning", *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3165809.

[19] J. Ma, *Mamba-minimal*, https://github.com/johnma2006/mamba-minimal, Feb. 2024.

[20] E. Almazrouei *et al.*, *The Falcon Series of Open Language Models*, Nov. 2023. arXiv: 2311.16867 [cs].