Long Range Dependency Understanding in State Space Models

Anonymous Authors[1]

Abstract

Although state-space models (SSMs) have demonstrated strong performance on long-sequence benchmarks, most research has emphasized predictive accuracy rather than interpretability. In this work, we present the first systematic kernel interpretability study of the SSM kernel trained on a real-world task. We present time and frequency domain analysis of the SSM kernel, and show that the long-range modeling capability of SSM varies significantly under different model architectures, affecting model performance. We assess the long and short range dependency understanding of the models through their filter behavior. For instance, SSM kernel can behave as low-pass, band-pass or high-pass filter. The insights from our analysis can guide the future work in designing better SSM based models.

1. Introduction

Structured State Space Models (SSMs) (Gu et al., 2021) have been introduced as efficient architectures for sequence modeling, showing competitive results in language (Gu et al., 2022b), time series, and program analysis tasks (Verma et al., 2025). By parameterizing input–output mappings through long convolutional kernels, they provide a scalable way to capture long-range dependencies. Gu et al. (2022b) demonstrated this potential on long-sequence benchmarks, positioning SSMs as strong alternatives to RNNs and transformers. The diagonal variant of SSM, S4D (Gu et al., 2022a), further improved computational efficiency and stability by simplifying kernel generation, making it more practical for large-scale applications. Recent work on understanding source code has explored both transformer-based approaches and state-space variants, with CodeSSM (Verma et al., 2025) indicating that SSMs can effectively model the syntactic and semantic structure of code.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Despite these advances, the mechanisms underlying the behavior of SSM remain insufficiently understood. Previous research has examined stability (Wang & Li, 2024) and generalization properties (Liu & Li, 2024b) of SSMs, but the practical implications for kernel behavior are less clear. In particular, it is not well established whether kernels emphasize local transitions, long-range dependencies, or specific spectral bands, nor how these properties vary across architectures. The lack of interpretability limits the ability to examine the strengths and failures of SSMs.

In this work, we study the behaviour of SSM kernels using the diagonalized variant, S4D, which provides computational efficiency and stable kernel generation while preserving the modeling capacity of SSMs (Gu et al., 2022a). We analyze CNN–S4D hybrid architectures in both the time and frequency domains to examine how architectural choices shape kernel responses and influence the modeling of dependencies. For our study, we used single-layer models to maintain architectural simplicity and interpretability, enabling detailed analysis of how individual kernels respond to specific code patterns. We trained the models on a real-world task (vulnerability detection in source code) instead of synthetic datasets used in previous works on SSM analysis. Single layer 1D CNNs have already been shown to perform well on this task (Seas et al., 2024). So, we use a single-layer CNN baseline to evaluate the performance of different S4D-based models. Using a single layer model also helps to quickly study the behavior of the model under various architectural changes.

Our observations depict that the standalone S4D model performs worse than CNN. Consistent with this observation, our kernel analysis reveals that standalone S4D model only learns short-range dependencies, despite its theoretical long-range modeling capacities (Gu et al., 2021; 2022b). On the other hand, the S4D model preceded by a state memory replay (SMR) (Qi et al., 2024) block performs the best and also shows long-range understanding in our analysis.

2. Methodology

In this section, we explain the different model architectures we have trained and the proposed methodology to analyze the behavior of the SSM kernel.

Figure 1. Common model architecture used across all experiments. The Feature extraction block represents different feature transformation strategies across six model variants.

2.1. Model architectures

We implemented six model architectures to evaluate performance and study kernel behavior in vulnerability detection. All models share a common architecture (Figure 1), where tokenized source code is mapped to dense embeddings and processed through a feature extraction block. The outputs are passed through ReLU activation, max pooling, concatenation, dropout, and a fully connected layer for binary classification. The feature extraction block is the only component that varies between models.

Convolution 1D baseline model: A single one-dimensional convolutional layer that applies multiple kernel sizes in parallel to extract local features similar to CNN architectures used in sentence classification tasks (Kim, 2014).

Depthwise separable convolution model: The model factors the standard convolution into two successive stages: a convolution in depth, where a single filter is applied per channel to extract the localized structure, and a point-wise (1×1) convolution, which mixes information between channels, significantly reducing the number of parameters and computational cost while maintaining predictive performance (Chollet, 2017).

S4D (standalone) model: In this architecture the feature extraction block contains a S4D (Gu et al., 2022a) layer. The S4D efficiently computes parameterized kernels in the frequency domain using FFTs.

Depthwise separable with S4D model: This hybrid design first applies a depth-wise convolution for local feature extraction, followed by an S4D layer with global context modeling, and finally a point-wise convolution to integrate information across channels. This model was designed to illustrate how SSMs and CNNs can complement each other in capturing both structural and semantic cues in code.

Conv1D + S4D model: This hybrid model extends the Conv1D baseline by stacking an S4D layer after the convolutional layer. The convolution layer captures fine-grained local features, while the subsequent S4D module models broader temporal dependencies across the sequence.

State Memory Replay(SMR) + S4D model: The final variant introduces a State Memory Replay (SMR) mechanism (Qi et al., 2024) wherein the outputs of the Conv1D layer are multiplied element-wise with the original embedding representations before being passed to the S4D layer, as shown in Figure 2. This fusion mechanism improves the ability of the model to recall contextual memory while preserving local structure.

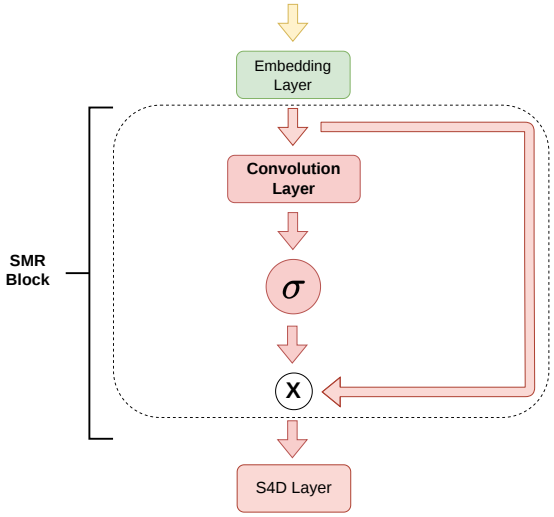

Figure 2. Feature extraction block for SMR + S4D model.

2.2. Training

We train all models on the ReVeal dataset (Chakraborty et al., 2020), a real-world benchmark for vulnerability detection. The dataset was constructed from two large-scale open source projects, the Linux Debian kernel and Chromium, covering diverse security issues across operating systems and browsers. It contains 22,734 functions, of which 2,240 are vulnerable and 20,494 are nonvulnerable, formulated as a binary classification task. To address class imbalance, we used BCEWithLogitsLoss with class weights.

2.3. Kernel analysis

To interpret the internal behavior of the trained models, we performed a kernel-level analysis in both the time domain and the frequency domain.

Time-domain Analysis. The convolutional and S4D kernels were extracted after training and were visualized in the

temporal domain. We evaluated two key properties:

- Peak sensitivity: maximum positive and negative amplitudes, indicating the strongest excitations of the kernel.
- Transition sharpness: first-order differences that highlight abrupt changes in kernel response.

This analysis reveals whether a kernel is dominated by sharp local transitions or smooth long-range dependencies.

Frequency-domain Analysis. Frequency-domain analysis was performed using the Fast Fourier Transform (FFT) to obtain the spectral representation of the kernel.

$$\hat{K}(f) = \mathcal{F}K(t). \tag{1}$$

Using the fourier transform of the kernel, the normalized Power Spectral Density (PSD) was computed as

$$P(f) = \frac{|\hat{K}(f)|^2}{\sum_f |\hat{K}(f)|^2}. \tag{2}$$

The following properties were extracted:

- **Dominant frequency:** the frequency with maximum spectral energy.
- **Secondary peaks:** significant frequency components with magnitudes above 30% of the dominant frequency.
- **Spectral entropy:** measures the concentration or spread of the spectral energy. The spectral entropy is calculated as

$$H = -\sum_f P(f) \log P(f). \tag{3}$$

These measures characterize whether kernels behave as selective narrowband filters or as broadband filters with distributed spectral energy.

3. Results and Discussion

We evaluated six model architectures. Table 1 summarizes the performance in accuracy, precision, recall, and F1 score. The SMR+S4D model achieved the highest F1 score of 88.03, outperforming the baselines of convolution-only and S4D-only. Separable depth-wise CNNs combined with S4D also performed competitively, demonstrating the benefit of integrating local convolutions with structured state-space representations.

We analyzed the learned S4D kernels in both the time and frequency domains to study their dynamic behaviors and complement performance metrics. The time-domain response is shown in Figure 3 and the frequency-domain spectra is shown in Figure 4. The **standalone S4D** model produced oscillatory kernels with peak amplitudes of approximately $+2.6/-6.2$. Its spectrum was dominated by a mid-frequency component around 0.18 cycles/sample ($\approx$ 5–6 token periodicity), with an entropy of $H = 3.94$. This indicates sensitivity to short- to mid-range transitions, consistent with its moderate performance. The **Depthwise+S4D** variant showed stronger early-sequence modulation ($+2.7/-3.8$) and a lower dominant frequency near 0.086 cycles/sample ($\approx$ 12 tokens). Its entropy of $H = 3.97$, together with multiple mid-frequency peaks, suggests a band-pass profile that captures both local and broader structures. The **Conv+S4D** model exhibited the sharpest temporal responses ($+6.7/-5.2$) and a broadband spectrum with dominant frequency ≈ 0.14 cycles/sample (≈ 7 tokens). Its entropy was the highest ($H = 4.21$), reflecting the distributed spectral energy between frequencies, suggesting that it can engage with both fine-grained syntax and longer-range semantics. In contrast, the **SMR+S4D** kernels had balanced amplitudes ($+3.9/-4.7$) and a very low dominant frequency of ≈ 0.02 cycles/sample (≈ 50 tokens). With the lowest entropy ($H = 3.79$), this model displayed a low-pass profile emphasizing smooth long-range dependencies while suppressing rapid variations, aligning with its superior F1 score of 88.03.

Table 1. Model performance comparison on the ReVeal dataset for filter size 6. The best result among all architectures is highlighted in bold.

Model	Accuracy	Precision	Recall	F1 Score
Basic Conv1D	87.64	87.67	87.64	87.66
Depthwise Separable Conv	87.12	**88.08**	87.12	87.57
S4D (Standalone)	87.07	86.97	87.07	87.02
Depthwise Separable + S4D	88.17	86.43	88.17	87.17
Conv + S4D	85.05	87.75	85.05	86.22
SMR + S4D	**88.26**	87.82	**88.26**	**88.03**

Thus our analysis reveals that when S4D is preceded by a SMR block, the S4D kernel can capture long-range dependency and the performance of the model improves while the standalone S4D layer cannot capture long-range dependency. This observation is consistent with findings in S4ND (Nguyen et al., 2022), where band-limiting high-frequency components improved performance. Our analysis suggests that hybrid CNN–SSM architectures can achieve similar band-limiting effects without explicit constraints. Our result is complementary to the previous work (Nishikawa & Suzuki, 2025), which proves theoretically that the performance of SSM improves if SSM layer is preceded by feedforward layer. We also performed token level activation analysis which is explained in Section F. Additionally, we show that separable depth-wise CNNs combined with S4D performs better than Conv 1D combined with S4D and

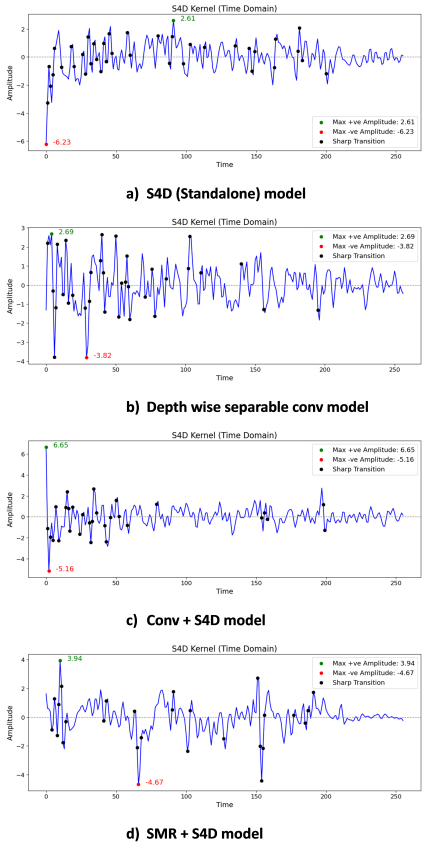

Figure 3. Time-domain impulse responses of S4D kernels across architectures. Green: maximum positive amplitude, red: maximum negative amplitude, black: sharp transitions.

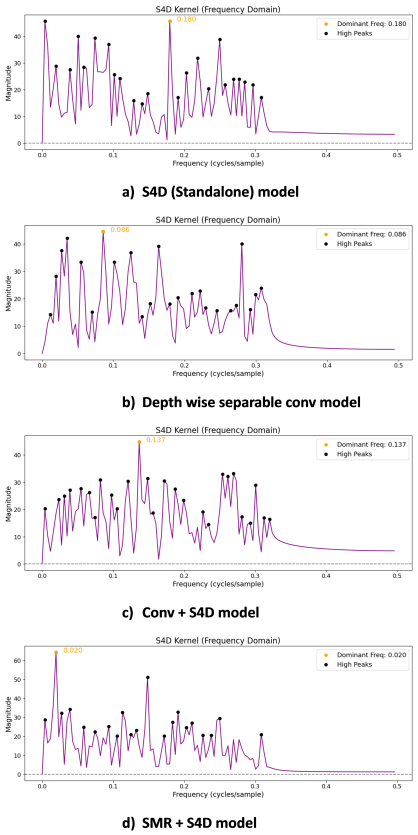

Figure 4. Frequency-domain spectra of S4D kernels across architectures. Orange: dominant frequency; black: secondary peaks ($\geq$30% of dominant).

captures longer range of tokens.

4. Related Work

Previous work in neural networks shows that models often display a spectral bias, learning smooth low-frequency components before fine-grained high-frequency details (Rahaman et al., 2019). Similarly, in convolutional models, the spectrum leakage is observed, where small kernels spread energy broadly across frequencies, while larger kernels concentrate it in low-frequency bands (Tomen & van Gemert, 2021; Hanin & Nica, 2019). From a signal processing perspective, long-range-dependent processes concentrate energy in low frequencies and exhibit lower entropy rates (Feutrill & Roughan, 2021). Complementing these findings, S4ND (Nguyen et al., 2022) mitigates spectral leakage by band-limiting noisy high-frequency components, thereby constraining the kernel to low-frequency bands and improving stability and performance in multidimensional modeling tasks. These findings suggest that analyzing SSM kernels through impulse responses and spectral distributions provides valuable insight into whether they capture smooth, global patterns or short, localized fluctuations. Deep learning models such as CNNs and transformers have been applied to tasks like vulnerability detection (Chakraborty et al., 2020). More recently, CodeSSM (Verma et al., 2025) demonstrated that SSMs can be promising alternatives for code-related tasks. However, they emphasized on predictive performance, while systematic interpretability analysis of SSM kernels remains limited. Some additional related works are discussed in Section A.

5. Conclusion

In this work, we presented the first systematic interpretability study of SSM kernels. By analyzing SSM-based models, we showed that different model architectures yield distinct filter behavior in the S4D kernel. Our analysis revealed that SMR+S4D exhibits a low-pass bias that emphasizes long-range dependencies, resulting in the best performance. The analysis framework presented in the paper can thus be used for understanding how the SSM model behaves. While we used this analysis framework for single-layer models, this work can be easily extended to multi-layer models. The insights can be used to improve the architecture of SSM-based models.

References

Chakraborty, S., Krishna, R., Ding, Y., and Ray, B. Deep learning based vulnerability detection: Are we there yet?, 2020. URL https://arxiv.org/abs/2009.07235.

Chollet, F. Xception: Deep learning with depthwise separable convolutions, 2017. URL https://arxiv.org/abs/1610.02357.

Feutrill, A. and Roughan, M. Differential entropy rate characterisations of long range dependent processes, 2021. URL https://arxiv.org/abs/2102.05306.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Gu, A., Johnson, I., Goel, K., Saab, K. K., Dao, T., Rudra, A., and Re, C. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=yWd42CWN3c.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=yJE7iQSAep.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Hanin, B. and Nica, M. Finite depth and width corrections to the neural tangent kernel, 2019. URL https://arxiv.org/abs/1909.05989.

Kim, Y. Convolutional neural networks for sentence classification, 2014. URL https://arxiv.org/abs/1408.5882.

Liu, F. and Li, Q. Autocorrelation matters: Understanding the role of initialization schemes for state space models, 2024a. URL https://arxiv.org/abs/2411.19455.

Liu, F. and Li, Q. From generalization analysis to optimization designs for state space models, 2024b. URL https://arxiv.org/abs/2405.02670.

Nguyen, E., Goel, K., Gu, A., Downs, G. W., Shah, P., Dao, T., Baccus, S. A., and Ré, C. S4nd: Modeling images and videos as multidimensional signals using state spaces, 2022. URL https://arxiv.org/abs/2210.06583.

Nishikawa, N. and Suzuki, T. State space models are provably comparable to transformers in dynamic token selection. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=QFgbJOYJSE.

Qi, B., Gao, J., Zhang, K., Li, D., Liu, J., Wu, L., and Zhou, B. SMR: State memory replay for long sequence modeling. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.483. URL https://aclanthology.org/2024.findings-acl.483/.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. On the spectral bias of neural networks, 2019. URL https://arxiv.org/abs/1806.08734.

Seas, C., Fitzpatrick, G., Hamilton, J. A., and Carlisle, M. C. Automated vulnerability detection in source code using deep representation learning. In *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0484–0490, 2024. doi: 10.1109/CCWC60891.2024.10427574.

Tomen, N. and van Gemert, J. Spectral leakage and rethinking the kernel size in cnns, 2021. URL https://arxiv.org/abs/2101.10143.

Verma, S., Anand, A., and Mezini, M. CodeSSM: Towards state space models for code understanding. In *The 2025 Conference on Empirical Methods in Natural Language Processing*, 2025. URL https://openreview.net/forum?id=qKwvxi02Ah.

Wang, S. and Li, Q. Stablessm: Alleviating the curse of memory in state-space models through stable reparameterization. In *ICML*, 2024. URL https://openreview.net/forum?id=nMN5hNZMQK.

A. Additional Related Works

SSMs have been extended in several directions to improve efficiency, stability, and adaptability. Beyond the original S4 and its diagonalized variant S4D, variants such as S4ND (Nguyen et al., 2022), StableSSM (Wang & Li, 2024) and Mamba (Gu & Dao, 2024) have extended the framework to multidimensional data, improved robustness and introduced input-dependent gating. Despite these advances, SSMs face challenges in stability and generalization. Vanilla variants suffer from the curse of memory, where hidden state dynamics collapses into short horizons, preventing effective modeling of long-range dependencies. To overcome this, stability-aware reparameterizations (Wang & Li, 2024) and initialization scaling methods (Liu & Li, 2024b) were proposed. Liu et al (Liu & Li, 2024a) emphasize the importance of autocorrelation and eigenvalue placement in initialization, which directly affect memory timescales and learning dynamics. Although these approaches mitigate instability and improve generalization, they do not fully explain the mechanisms that govern kernel behavior.

B. Data Pre-processing

The overall data pre-processing pipeline consists of loading raw source code functions, tokenizing code sequences, constructing vocabularies, converting tokens into indices, and padding or truncating sequences to fixed lengths. For the ReVeal dataset (Chakraborty et al., 2020), the entries were shuffled and divided into training, validation, and test sets using an 80/10/10 split strategy. A vocabulary was constructed by extracting unique tokens from the training data.

Each source code function was tokenized using regex-based tokenization. The resulting token sequences were mapped to corresponding vocabulary indices and padded or truncated to a maximum sequence length of 256 tokens. The processed samples were wrapped into custom PyTorch Dataset objects and passed into DataLoader instances for mini-batch training and evaluation.

C. Training Details

All models were trained using Adam Optimizer (learning rate = 1×10^{-3}), with dropout 0.5, sequence length capped at 256 tokens, batch size 64, and 10 training epochs. Early stopping was not applied to maintain consistent training duration across runs. The data set was divided into 80% training, 10% validation and 10% testing, and all experiments were implemented in PyTorch and executed on CUDA 11.8.

D. Mathematical Formulation of S4D

D.1. Structured State Space Models (SSMs)

Structured State Space Models (SSMs) are sequence modeling architectures derived from linear time-invariant dynamical systems (Gu et al., 2021; 2022b). Unlike transformer-based models that rely on pairwise attention mechanisms, SSMs process sequences using recurrent state dynamics that can be represented as convolution operations.

A continuous-time SSM is formulated as:

$$x'(t) = Ax(t) + Bu(t) \tag{4}$$

$$y(t) = Cx(t) + Du(t) \tag{5}$$

where:

- $x(t)$ represents the latent state,
- $u(t)$ is the input sequence,
- $y(t)$ is the output sequence,
- A, B, C, and D are learnable matrices.

After discretization, the system can be expressed as a convolution:

$$y = K * u \tag{6}$$

where K is a convolution kernel derived from the state-space dynamics (Gu et al., 2021). This formulation enables efficient long-range sequence modeling using convolutional kernels computed through Fast Fourier Transforms (FFTs).

D.2. Structured State Space Sequence Model (S4)

The Structured State Space Sequence Model (S4) extends traditional SSMs by introducing specialized parameterizations that improve stability and long-range memory retention (Gu et al., 2022b). S4 uses HiPPO-based initialization and structured state matrices to efficiently model long sequences while maintaining stable dynamics. Unlike transformers that rely on self-attention mechanisms with quadratic complexity, S4 models long-range dependencies using structured convolution kernels with linear-time complexity.

D.3. Diagonal Structured State Space Sequence Model (S4D)

The Diagonal Structured State Space Sequence Model (S4D) is a simplified and computationally efficient variant of S4

(Gu et al., 2022a). Instead of using a diagonal-plus-low-rank parameterization as in S4, S4D adopts a strictly diagonal state transition matrix, significantly reducing computational complexity and memory overhead while preserving long-range modeling capability.

The S4D kernel is parameterized using two learnable complex-valued vectors:

- A, which controls the decay and oscillation behavior of the kernel,
- C, which controls the contribution of each temporal mode.

Given a sequence length L, the convolution kernel is computed as:

$$K[n] = \mathrm{Re}(C \cdot e^{nA}), \quad n = 0, 1, \ldots, L-1 \tag{7}$$

The real component of A determines the decay rate of the kernel, while the imaginary component controls oscillatory behavior (Gu et al., 2022a). This formulation enables S4D kernels to capture temporal dynamics such as exponential decay, oscillatory behavior, and smooth transition patterns across sequences. The kernels are efficiently computed using FFTs, enabling scalable sequence modeling with linear-time complexity.

E. Additional Experimental Results

Table 2 presents additional experimental results on the ReVeal dataset using filter size 9 across different model architectures. These experiments were conducted under the same training conditions described for the filter size 6 experiments in the Table 1.

Table 2. Model performance comparison on the ReVeal dataset for filter size 9.

Model	Accuracy	Precision	Recall	F1 Score
Basic Conv1D	86.41	88.07	8641	87.16
Depthwise Separable Conv	87.25	87.84	87.25	87.53
S4D (Standalone)	87.73	87.17	87.73	87.44
Depthwise Separable + S4D	87.69	87.64	87.69	87.66
Conv + S4D	86.94	86.91	86.94	86.93
SMR + S4D	86.98	87.98	86.98	87.45

F. Token Level Activation Analysis

To better understand the internal behavior of the proposed architectures, a sample-level interpretability analysis was conducted on both correctly classified and misclassified examples from the ReVeal dataset (Chakraborty et al., 2020). Rather than relying solely on evaluation metrics, this analysis investigates how the SMR + S4D model responds to different code patterns through token-wise activations in both the time and frequency domains.

The activations were extracted from the S4D layer after inference, and Fast Fourier Transform (FFT) analysis was performed to study the frequency characteristics of the learned representations. Low-frequency components correspond to smooth global contextual modeling, whereas high-frequency components indicate sensitivity to abrupt local fluctuations (Rahaman et al., 2019; Tomen & van Gemert, 2021).

F.1. Correctly classified sample analysis

Figure 6 presents the token-wise activations of the S4D layer for a correctly classified vulnerable sample. The selected function contains memory-related operations and indexing expressions that require contextual reasoning across multiple tokens and statements.

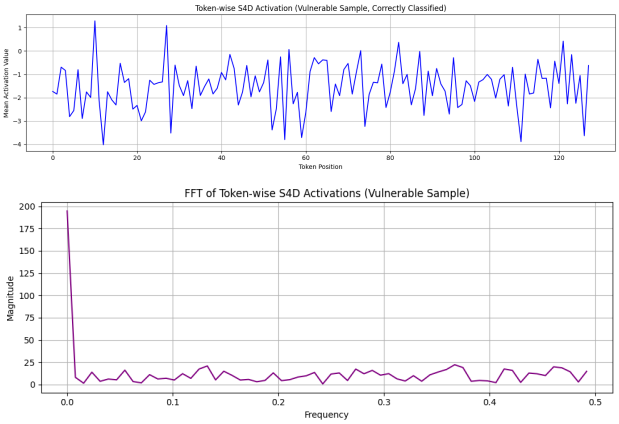

Figure 5. Token-wise activations extracted from the S4D layer for a correctly classified vulnerable sample in both the time and frequency domains.

The time-domain activations exhibit relatively smooth transitions across the sequence without significant oscillatory behavior. Strong activations are primarily concentrated around semantically meaningful operations such as memory accesses, indexing logic, and arithmetic computations associated with vulnerability-relevant behavior. Unlike conventional convolution-only architectures that operate within fixed receptive fields, the S4D layer preserves contextual information over longer token ranges, enabling the model to associate distant operations within the same functional context.

The corresponding FFT spectrum reveals that the majority of the spectral energy is concentrated in the low-frequency region. This low-frequency dominance indicates that the learned representation evolves gradually across the sequence and captures globally coherent semantic structure rather than isolated syntactic artifacts. Such behavior is particu-

larly desirable for vulnerability detection tasks, where vulnerabilities often emerge from interactions between distant code regions instead of individual token patterns (Feutrill & Roughan, 2021).

Furthermore, the absence of significant high-frequency components suggests that the model is not excessively sensitive to repetitive delimiters, variable names, or localized token-level noise. Instead, the learned representation reflects stable long-range dependency modeling, which aligns with the improved predictive performance observed for the SMR + S4D architecture (Qi et al., 2024).

Overall, the correctly classified sample demonstrates that the proposed architecture successfully integrates local feature extraction with long-range contextual reasoning, resulting in semantically stable internal representations.

F.2. Misclassified sample analysis

To better understand the limitations of the proposed architecture, we additionally analyzed a non-vulnerable sample that was incorrectly classified as vulnerable. The analyzed function primarily consists of arithmetic operations, nested loops, and repeated indexing expressions.

```
void vp9_fdct16x16_1_c(const int16_t *input,
                       tran_low_t *output,
                       int stride) {
    int r, c;
    tran_low_t sum = 0;

    for (r = 0; r < 16; ++r)
        for (c = 0; c < 16; ++c)
            sum += input[r * stride + c];

    output[0] = sum >> 1;
    output[1] = 0;
}
```

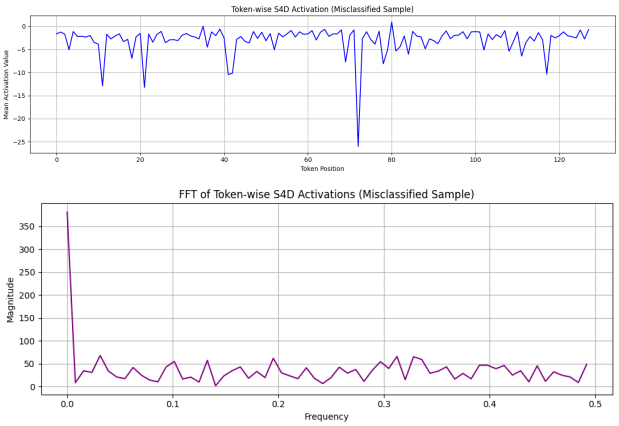

Figure 6. Token-wise activations extracted from the S4D layer for a misclassified non-vulnerable sample in both the time and frequency domains.

In contrast to the correctly classified example, the time-domain activations display abrupt fluctuations and sharp activation spikes across multiple token positions. Particularly strong activations are observed around repetitive syntactic structures such as `;`, `output`, and `sum`, despite the absence of any actual vulnerability in the function.

This behavior suggests that the model has learned strong statistical associations between certain recurring token patterns and vulnerable samples during training. Consequently, benign functions containing similar arithmetic or indexing structures may incorrectly activate vulnerability-related filters, resulting in false positive predictions.

The frequency-domain analysis further supports this observation. Although low-frequency components are still present, the FFT spectrum exhibits substantially higher mid- and high-frequency energy compared to the correctly classified sample. High-frequency components correspond to abrupt local activation changes and indicate that the model is reacting strongly to localized syntactic patterns rather than constructing a globally stable semantic representation.

Such activation instability highlights an important limitation of data-driven vulnerability detection systems. While the S4D layer is capable of modeling long-range contextual information, certain repetitive syntactic structures can still dominate the learned representation and override broader semantic reasoning.

F.3. Key observations

The qualitative analysis reveals several important characteristics of the proposed architecture:

- Correctly classified samples exhibit smooth activation distributions and dominant low-frequency behavior, indicating stable long-range dependency modeling.

- Misclassified samples contain abrupt activation spikes and stronger high-frequency components, suggesting over reliance on local syntactic structures.
- The S4D layer provides interpretable sequence dynamics in both the time and frequency domains.
- FFT-based analysis helps distinguish between globally coherent semantic modeling and unstable local token sensitivity.

These findings demonstrate that although the SMR + S4D architecture effectively captures long-range contextual information, false positives can still arise when the model overfits to repetitive syntactic patterns rather than deeper semantic behavior.