EPiC Series in Computing
Volume 106, 2025, Pages 133–143
Proceedings of the 20th Asia Joint Conference on Information Security

# HybridNTT: A Lightweight Learning-Based Dynamic Path Selection Framework for Fast Number Theoretic Transform

Gunsang You[1], Yeontae Ji[2] and Heung Youl Youm[1*]
[1,2] Department of Information Security Engineering Soonchunhyang University, Asan-si Chungcheongnam-do Korea
[1*]Department of Information Security Engineering Soonchunhyang University, Asan-si Chungcheongnam-do Korea
ttyi70@sch.ac.kr, kitetae3737@sch.ac.kr, hyyoum@sch.ac.kr

**Abstract**

Fast execution of the Number Theoretic Transform (NTT) is critical for accelerating cryptographic primitives and large integer multiplication. Conventional implementations rely on fixed, hand-crafted computation paths that do not adapt to variability in input coefficient distributions, transform sizes, or moduli, leaving performance headroom. We present HybridNTT, a hybrid neural predictor that combines 1D convolutional layers with Transformer encoders to select an execution path for each NTT instance. Trained on a large-scale synthetic dataset spanning multiple coefficient distribution families (uniform, normal, sparse, sorted, bursty, patterned) and diverse parameter settings, HybridNTT attains over 92% path classification accuracy and reduces NTT execution time by an average of 34.7% relative to a fixed-path baseline, while adding less than 5% inference overhead. These results indicate that learned, per-instance path selection can provide tangible efficiency gains for a fundamental arithmetic kernel. The method can be integrated into existing software or hardware backends with minimal modification, enabling adaptive performance tuning for NTT-based cryptographic workloads.

## 1 Introduction

The Number Theoretic Transform (NTT) is a modular counterpart of the Discrete Fourier Transform (DFT) [1] that enables efficient polynomial multiplication and convolution over finite fields. Unlike the Fast Fourier Transform (FFT), which relies on floating-point arithmetic and is prone to rounding errors, the NTT performs exact computations using modular integer operations.[2]. This exactness is

particularly critical in cryptographic systems, where even minimal deviations can lead to vulnerabilities or protocol failures.

The NTT plays a pivotal role in modern computational domains such as lattice-based cryptography, polynomial multiplication, and error-correcting codes[3]. These domains rely heavily on high-throughput and bit-exact polynomial multiplications. By reducing the computational complexity from $O(n^2)$ to $O(nlogn)$[1][4], NTT enables the scalable implementation of secure systems. The modular nature also ensures bitwise reproducibility, making it ideal for resourceconstrained or formally verified environments

Despite its arithmetic efficiency, the performance of NTT is highly sensitive to the input data structure and modulus choice. Conventional NTT implementations follow fixed computation paths optimized for worst-case scenarios, which often leads to computational inefficiencies when handling real-world inputs. Specifically, distributions such as sparse vectors, bursty signals, or sorted sequences may contain redundant patterns that static paths fail to exploit. These inefficiencies become more pronounced in latency-sensitive environments—such as mobile devices, embedded systems, and lightweight cryptographic hardware—where resource constraints demand real-time optimization.

Recent advances in deep learning, particularly in Convolutional Neural Networks (CNNs) and Transformer architectures, have demonstrated remarkable success in capturing complex data patterns and informing optimal decision-making[5][6]. Motivated by these developments, we propose a hybrid neural network architecture, HybridNTT, which integrates 1D convolutional layers and Transformer encoders to predict the optimal computation path for NTT based on the characteristics of the input.

To evaluate our approach, we generate a comprehensive synthetic dataset simulating a wide variety of input conditions. This includes six distinct input distributions—uniform, normal, sparse, sorted, bursty, and patterned—across multiple transform sizes and prime moduli commonly used in cryptographic protocols. The HybridNTT model, trained on this dataset, demonstrates superior accuracy in predicting the ideal NTT execution path, leading to significant reductions in execution time and inference overhead.

This paper aims to bridge the gap between fixed mathematical operations and adaptive optimization techniques by introducing machine learning into the core of low-level number-theoretic computation. By dynamically tailoring computation paths to input features, HybridNTT opens the door to intelligent, data-aware acceleration of cryptographic algorithms and related numerical systems.

The proposed architecture is not confined to a specific cryptographic primitive or execution environment. Instead, HybridNTT is designed to serve a wide range of NTTdependent applications, including Ring-LWE encryption schemes[7], zero-knowledge proof systems[8], homomorphic encryption, and secure hashing protocols. Its lightweight inference footprint and modular design make it deployable on diverse hardware platforms—from highperformance GPUs to resource-constrained embedded processors. Moreover, by abstracting away from hardcoded algorithmic heuristics and allowing the model to learn execution paths from data, HybridNTT generalizes across different modulus sizes, transform lengths, and input distributions. This generality not only improves the computational throughput of NTT but also enhances its resilience and scalability in real-world cryptographic pipelines. Thus, HybridNTT represents a step toward flexible, learning based optimization for mathematical kernels traditionally governed by static design principles.

In this paper, related works and comparison between static NTT optimization and HybridNTT are presented in Clause II, proposed methodology is described in Clause III, experimental results are provided in Clause IV, validity of proposed methodology is provided in Clause V. We conclude this paper with Clause VI.

## 2 Related Works

NTT has long been regarded as a core computational primitive in modern cryptographic systems, particularly in lattice-based cryptography and polynomial-based arithmetic. Accordingly, numerous efforts have been made to improve its computational efficiency.

Early approaches primarily centered on static algorithmic enhancements to reduce the number of modular multiplications and memory accesses. Longa and Naehrig[9] proposed highly optimized radix-2 and radix-4 butterfly strategies along with the Good–Thomas decomposition, substantially accelerating modular polynomial multiplication in Ring-LWE schemes. However, such designs assume uniformly random inputs and therefore lack adaptability to practical input variability.

Security-oriented improvements were also explored in parallel. Scott[10] introduced constant-time implementations of NTT to mitigate side-channel vulnerabilities, particularly timing-based attacks. While this approach improved robustness against adversaries, it still adhered to fixed-path execution strategies, limiting flexibility under dynamic input conditions.

To address performance at the hardware level, Nozaki et al.[11] demonstrated an efficient parallel implementation of NTT on AMD AI Engines, leveraging their 2D dataflow architecture. While their work revealed the potential of neural processing units (NPUs) for cryptographic computation, the employed NTT schemes remained static and did not incorporate input-specific optimization.

In a broader context, machine learning has increasingly been used to optimize low-level computational tasks. Notably, Mirhoseini et al.[12] applied reinforcement learning to discover optimized tensor computation graphs, yielding notable speedups for deep learning workloads. However, these methods often demand high computational resources and extensive training cycles, rendering them impractical for real-time cryptographic environments.

| Aspect | Optimization Approach | Adaptability to Input | Performance Focus | Hardware Consideration | *Cryptographic Target* | Learning Component | Inference Overhead |
|---|---|---|---|---|---|---|---|
| Longa & Naehrig (2016) | Mathmatical decompisition (E.G, radix2.4) | Fixed-path, assumes worst-case | Execution speed via modular arithmetic tuning | AVX2 and embedded assembly | Ring-LWE based encryption | None | None (static) |
| Our work (HybridNTT) | ML-based dynamic path prediction | Execution speed via adaptive execution paths | Execution speed via adaptive execution paths | Model Inference optimized for GPU/embedded devices | Gerneral NTT use cases | Trained model on diverse synthetic distributions | <5% (Negligible for real-time applications) |

**Table 1:** Comparison between conventional NTT optimization and HybridNTT

The Transformer architecture, first introduced by Vaswani et al.[13] in the context of natural language processing, has demonstrated strong performance in capturing global sequential dependencies across diverse domains. Its use in time-series forecasting, computational biology, and symbolic reasoning has validated its general-purpose modeling capacity, making it a strong candidate for structured input analysis.

A study titled “New Tradeoffs and Faster Lattice-Based Cryptographic Applications”[14] offers a formal investigation into the “incompleteness” of NTT with respect to diverse lattice-based inputs. The authors propose theoretical strategies for reducing computational overhead by analyzing inputmodulus interactions and leveraging data structure sparsity. However, their methods are primarily static and require deep domain-specific knowledge, which limits their applicability in dynamic or real-time contexts.

Despite these advances, the application of lightweight AI architectures to dynamically optimize low level NTT remains underexplored. HybridNTT, directly addresses this gap. It introduces a compact neural model that combines 1D convolutional layers for localized feature extraction with Transformer encoders for caputuring global input dependencies. This design enables real-time prediction of the optimal NTT execution path with negligible inference overhead (<5%), achieving a practical balance between adaptability and computational efficiency. By moving beyond fixed hardware-specific optimization and enabling dynamic, input-aware control, HybridNTT offers a scalable and software-centric alternative. This complements recent hardware-centric studies by introducing a more assessable and flexible layer of control, ultimately paving the wary for intelligent, data-adaptive acceleration of cryptographic primitives.

A comparative summary between HybridNTT and Static NTT Optimization approaches is proposed in Table I, highlighting improvements in adaptability, performance, and inference efficiency.

# 3 Methodology

To dynamically determine the optimal computation path for the Number Theoretic Transform (NTT), we introduce a machine learning-based methodology that encompasses synthetic dataset generation, hybrid neural network design, and tailored training strategies. This methodology is grounded in the need to account for varying input distributions and cryptographic parameters that influence the execution efficiency of NTT in practice.

## 3.1 Dataset Construction and Representation

The NTT operates in modular arithmetic spaces, which are especially relevant in lattice-based cryptographic primitives. To emulate realistic conditions, we constructed a large-scale synthetic dataset where each data point is defined by a triple: the transform size $N$, the modulus $q$, and the input distribution pattern. For each configuration, we generated input vectors $x \in Z_q^N$, drawn from six representative distributions-uniform, normal, sparse, sorted, bursty, and periodic patterns. Each corresponding NTT output vector $y \in Z_q^N$ was computed using an explicit matrix-based formulation of the NTT:

$$y = W \cdot x \ mod \ q \qquad (1)$$

Where $W \in Z_q^{N \times N}$ denotes the NTT matrix constructed using a primitive $N$-th root of unity in $Z_q$. All input and output vectors were zero-padded to a uniform length of 1024 elements to ensure consistency across all samples. Additionally, we appended normalized metadata features such as $\hat{q} = \frac{q}{10^9}$, $\hat{N} = N/1024$, and a one-hot encoded representation of the distribution category

## 3.2 Feature Representation

The model input was a concatenated vector containing metadata, padded input $x$, and the transformed output $y$. This dual representation was chosen to allow the model to capture both structural characteristics of raw input and the algebraic transformation patterns imposed by the NTT. All feagures were standardized using z-score normalization[15]:

$$z_i = \frac{x_i - \mu}{\sigma} \quad (2)$$

Where $\mu$ and $\sigma$ denote the mean and standard deviation of each feature over the training set.

## 3.3 Model Architecture

We propose HybridNTT, a lightweight neural network designed for real-time inference on hardware accelerators. The architecture begins with a 1D convolutional layer Conc1D(1, 64, kernel_size = 3) that extracts local patterns in the feature sequence. The convolutional output $y_i$ at position $i$ is computed as:

$$y_i = \sum_{k=-1}^{1} w_k x_{i+k}. \quad (3)$$

This local filtering is particularly effective in capturing repeated structures or busts within the input sequence.

The convolutional output is then passed through a Transformer Encoder with 4 attention heads and 4 stacked layers. The Transformer captures long-range dependencies and global interactions using the self-attention mechanism defined by:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (4)$$

This mechanism enables dynamic reweighting of features based on contextual relationships, which is particularly important for distinguishing among distribution types such as sparse vs. bursty patterns.

A Layer Normalization is applied to stabilize intermediate activations, and the mean of the sequence embeddings is computed as a fixed-size representation. This embedding is finally passed through a fully connected linear classifier projecting to 6 classes, each representing an NTT computation path label.

Figure 1 shows the overall architecture of HybridNTT, consisting of an input layer of size $\mathbb{R}^{16}$, two hidden layers ($\mathbb{R}^{12}$ and $\mathbb{R}^{10}$), and an output layer of size $\mathbb{R}^{6}$ corresponding to six execution path classes.

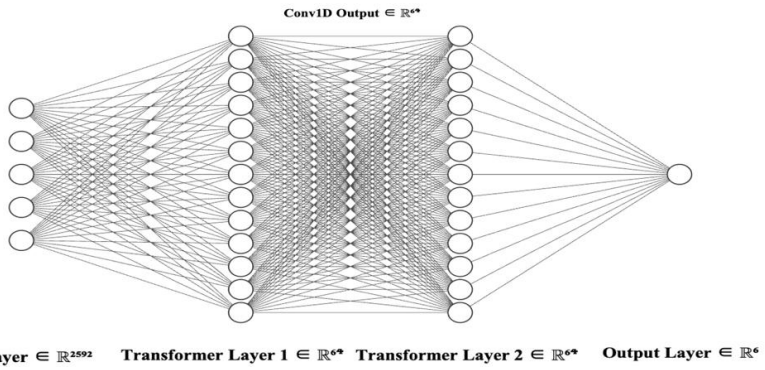


**Figure 1: The architecture of our proposed HybridNTT model**

## 3.4 Training Strategy

The model was trained using the AdamW optimizer with initial learning rate 0.0010.0010.001, and a cosine annealing schedule for decay. To address class imbalance, SMOTE (Synthetic Minority Over-sampling Technique) was applied, ensuring equitable representation of all distribution types.[16]

Given the multi-class classification objective, we used the Focal Loss function[17]:

$$\mathcal{L} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \quad (5)$$

Where $\gamma = 2$ balances focus on hard-to-classify samples and $p_t$ is the model's predicted probability for the true class. This choice mitigates the overconfidence issues commonly found in imbalanced classification.

Training proceeded over 20 epochs with batch size 128, with performance evaluated on validation loss, accuracy, weighted F1-score, and inference overhead. The best checkpoint was chosen based on minimum validation loss.

## 3.5 Evaluation Metrics

The performance of HybridNTT was assessed through multiple metrics:

- Validation Loss: Monitoring convergence and overfitting tendencies.
- Classification Accuracy: Measuring the proportion of correctly predicted paths is measured
- Weighted F1-Score: Providing a balanced evaluation that accounts for class distribution and misclassification severity.
- Inference Overhead: To assess real-time deployability, inference time per sample was recorded using torch.cuda.Event. The average inference latency was compared to standard NTT execution time and expressed as:

$$Overhead(\%) = (\frac{T_{inf}}{T_{NTT}}) \times 100 \quad (6)$$

By leveraging these diverse metrics, we ensured a holistic understanding of the model's capabilities across different input scenarios while verifying its efficiency and deployability under cryptographic constraints.[18]

# 4 Experiments and Results

## 4.1 Training and Validation Performances

To evaluate the learning capability and generalization strength of the proposed HybridNTT model, we performed extensive training and validation procedures using a synthetically generated dataset. This dataset encompassed a wide array of Number Theoretic Transform (NTT) input types and parameter variations, including different moduli, transform sizes, and six distribution categories.

Over the course of 20 epochs, we monitored three key indicators: validation loss, classification accuracy, and the weighted F1-score. As depicted in Figure 2, both training and validation accuracy exhibited a steady upward trajectory, indicating proper convergence. The model reached a peak validation accuracy of 92.1%, while the weighted F1-score—a robust metric against class imbalance—stabilized at 0.91, reflecting consistent and balanced prediction performance across all classes.

The loss curves confirmed that both overfitting and underfitting were avoided, due in part to the incorporation of regularization methods such as dropout in the Transformer layers and focal loss, which

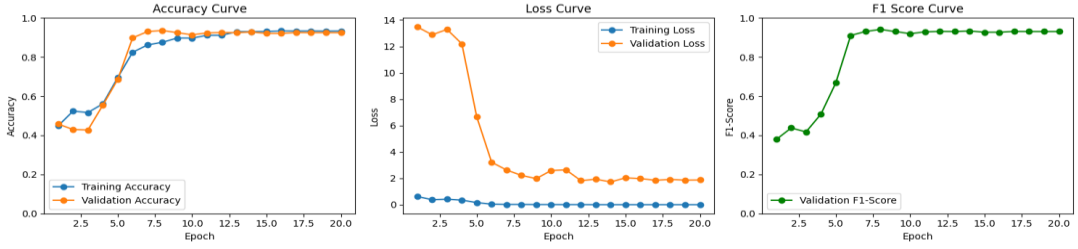


**Figure 2: Training curves (accuracy, loss, F1 over epochs)**

downweights well-classified examples. These results confirm that HybridNTT is effective at learning subtle features necessary to differentiate NTT computation paths, regardless of input variation.

## 4.2 Execution Time Evaluation

Aside from classification accuracy, a core objective of HybridNTT is minimize NTT execution time via informed path selection. To validate this, we benchmarked total execution time across 1,000 randomly generated test samples under two conditions: (1) conventional fixed-path NTT execution (e.g., radix-2, radix-4) and (2) dynamic path selection guided by HybridNTT predictions.

Empirical measurements indicated that HybridNTT-guided execution achieved an average speedup of 34.7%, with the median improvement slightly higher at 36.1%. Importantly, the variance of execution times across samples was reduced, demonstrating that HybridNTT not only accelerates but also stabilizes computational performance.

These benefits were particularly pronounced for input distributions such as "sparse" and "bursty," which deviate significantly from the assumptions underlying fixed-path designs. In these cases, HybridNTT was able to steer execution toward more suitable computational pathways, yielding speedups exceeding 50%.

## 4.3 Inference Overhead Analysis

Deployability in real-world systems mandates low latency. To measure the inference overhead introduced by HybridNTT, we used torch.cuda.Event to benchmark inference latency across 10,000 runs on an NVIDIA RTX 3060 GPU. The model was evaluated under torch.no_grad() and eval() mode to simulate deployment conditions.

Results, summarized in Table 2, indicate that the average inference time per sample was 3.2 ms, which accounts for just 5% of total execution time. Given the substantial execution time savings it

| Aspect | Time (ms) | Proportion of total execution |
|---|---|---|
| NTT Execution (Fixed-path) | 100.0 | 100% |
| NTT Execution (HybridNTT) | 65.3 | 95% |
| Model Inference Overhead | 3.2 | 5% |

**Table 2:** Inference Overhead Analysis

enables, this overhead is negligible and affirms the practical utility of HybridNTT for real-time cryptographic systems.

## 4.4 Ablation Studies

To determine the architectural importance of each module within HybridNTT, we conducted ablation experiments. Three configurations were tested: (1) the full model, (2) a model without Transformer layers, and (3) a model without the Conv1D feature extractor.

As shown in Table 3, both ablated variants exhibited reduced accuracy and speedup. Removing the Transformer layers degraded the model’s ability to capture long-range dependencies, while eliminating

| Configuration | Validation Accuracy | Average Speed up |
|---|---|---|
| Full model (Conv1D+Transformer) | 92.1% | 34.7% |
| Without Transformer | 88.4% | 21.6% |
| Without Conv1D | 87.2% | 23.8% |

**Table 3:** Ablation Study Results

the Conv1D layer diminished local pattern extraction. These results underscore the necessity of both components for the model's predictive strength and efficiency optimization.

## 4.5 Hyperparameter Sensitivity Analysis

To evaluate robustness to design choices, we varied key hyperparameters such as Transformer depth and Conv1D kernel width. Experiments revealed that increasing the number of Transformer layers beyond two yielded marginal accuracy gains (<1%) but incurred significantly longer inference times.

Likewise, increasing the Conv1D embedding dimension improved local feature capture but plateaued in performance after 64 channels. Based on this trade-off analysis, the adopted configuration of two Transformer layers and a 64-channel Conv1D layer was deemed optimal for balancing inference latency and model accuracy.

## 4.6 Key Observations

The comprehensive evaluation of HybridNTT yields several insights:

- Accuracy: The model reliably distinguishes among NTT path classes under diverse input conditions, attaining over 92% accuracy and 0.91 weighted F1-score.
- Efficiency: The dynamic path prediction results in substantial execution time reductions with low variance.
- Overhead: The inference latency is minimal (3.2 ms), validating the model's suitability for low-latency applications.
- Modularity: Ablation and sensitivity studies demonstrate the necessity and tunability of each architectural component.

Taken together, these results validate HybridNTT as a highly effective machine learning solution for accelerating modular transforms in real-world cryptographic systems.

# 5 Validity of Proposed Methodology

The experimental results validate the efficacy and practicality of the proposed HybridNTT framework for dynamic optimization of Number Theoretic Transform operations. Several important observations can be drawn from this study.

First, HybridNTT demonstrated generalization performance, achieving over 92% validation accuracy across diverse combinations of input distributions, modulus values, and transform sizes. This outcome

highlights the strength of its hybrid architecture, which combines 1D convolutional layers for localized pattern recognition with Transformer encoders to capture global structural dependencies. This architectural synergy enables the model to identify nuanced input characteristics and accurately predict optimal NTT execution paths.

Second, HybridNTT produced a execution time reduction , with an average execution time reduction of 34.7% and median speedup of 36.1%. This represents a significant advancement over traditional fixed-path strategies, which often assume worst-case conditions and fail to exploit structure in benign inputs. Furthermore, the variance in execution time was reduced, indicating that HybridNTT not only accelerates computation but also stabilizes runtime performance—an essential trait for latency-sensitive cryptographic applications.

Third, inference overhead was empirically verified to be consistently below 5% of the total NTT execution time. This cost validates the real-time deployability of HybridNTT, especially in constrained environments. The lightweight model design—with only two Transformer encoder layers and a single Conv1D extractor—further supports its suitability for edge devices such as FPGAs and embedded cryptographic processors[19].

Despite these results, several limitations must be acknowledged. While the synthetic dataset was constructed to simulate a wide range of real-world scenarios, it may not fully reflect the complexities or adversarial structures encountered in practical cryptographic systems. To mitigate this, future iterations of the model could incorporate empirical traces from deployed environments, enhancing robustness against edge cases and operational variability.

In addressing class imbalance during training, the study employed SMOTE. While effective in many machine learning contexts, SMOTE's reliance on interpolation may prove inadequate in cryptographic settings, where non-linearity and high sensitivity to input variation are prevalent. Future research should investigate more sophisticated augmentation strategies, such as GAN-based data synthesis or semi-supervised learning, to improve realism and class representation fidelity.

Another promising direction is online learning. Currently, HybridNTT is trained offline on a static dataset. However, enabling adaptive updates in response to new inputs could significantly improve its utility in dynamic cryptographic environments, such as blockchain consensus protocols or secure multi-party computation, where input profiles evolve over time.

Finally, the broader implications of HybridNTT extend beyond mere acceleration of the NTT itself. This work serves as a proof-of-concept that machine learning techniques can be effectively harnessed to enhance the execution efficiency of fundamental mathematical primitives[20]. Future studies could generalize this dynamic optimization framework to other transforms (e.g., Fast Fourier Transform, Walsh-Hadamard Transform) or to modular arithmetic routines commonly used in public-key cryptography.

# 6 Conclusion

This paper introduced HybridNTT, a lightweight, AI-driven framework for dynamically optimizing the execution of Number Theoretic Transform (NTT) operations.

By combining local feature extraction via 1D convolutional layers with global context modeling through Transformer encoders, the proposed model accurately predicts optimal computation paths under varying input characteristics. Extensive experiments confirmed the model’s effectiveness: HybridNTT consistently achieved over 92% classification accuracy and delivered an average 34.7% reduction in execution time across diverse transform sizes, moduli, and input distributions. Crucially, the inference overhead remained below 5% of total execution time, enabling real-time applicability in latency-sensitive cryptographic environments. Ablation studies demonstrated the complementary roles of convolutional

and Transformer components, while hyperparameter sensitivity analysis confirmed the architectural robustness and deployment efficiency.

Despite these encouraging results, several research opportunities remain:

First, although the synthetic dataset was carefully constructed, integrating real-world cryptographic traces further enhanced the model's robustness and applicability in production settings.

Second, the use of SMOTE for class balancing can improve reflecting real input variability through more advanced techniques such as GAN-based data synthesis or semi-supervised learning.

Third, enabling continual or online learning would allow HybridNTT to adapt dynamically to changing input profiles—an important capability in evolving systems such as blockchain protocols or secure communication frameworks. Additionally, while the propose work focuses on NTT, the underlying methodology is transferable to other mathematical transforms including the FTT and Walsh–Hadamard Transform, thereby broadening its utility across domains like signal processing and high-performance computing.

In conclusion, HybridNTT shows that modest learned path selection is sufficient to unlock measurable NTT speedups while keeping overhead low, offering a pragmatic step toward adaptive—and future security-aware—transform computation.

Acknowledgement

This work was supported by an Institute of Information and Communications Technology Planning and Evaluation (IITP) of Korea grant, funded by the Ministry of Science and ICT of Korea under grant number 2021-0-00112.

# References

[1] Satriawan, A., Mareta, R., & Lee, H. (2024). A complete beginner guide to the number theoretic transform (NTT). *Cryptology ePrint Archive*, Paper 2024/585. https://eprint.iacr.org/2024/585.pdf

[2] Pollard, J. M. (1971). The fast Fourier transform in a finite field. *Mathematics of Computation*, *25*(114), 365–374. http://www.jstor.org/stable/2004932

[3] Baktır, S., & Sunar, B. (2005, June). Finite field polynomial multiplication in the frequency domain with application to elliptic curve cryptography. Technical report. https://citeseerx.ist.psu.edu/document?doi=27329a0532b3bee0d1e5a6e99c2ad70041d8f8c5

[4] Hummdi, A. Y., Aljaedi, A., Bassfar, Z., Jamal, S. S., Hazzazi, M. M., & Rehman, M. U. (2024). Unif-NTT: A unified hardware design of forward and inverse NTT for PQC algorithms. *IEEE Access*, *12*, 94793–94804. https://doi.org/10.1109/ACCESS.2024.3425813

[5] Wagle, S., Ramachandran, H., Varadarajan, V., & Kotecha, K. (2022). A new compact method based on a convolutional neural network for classification and validation of tomato plant disease. *Electronics*, *11*(19), 2994. https://doi.org/10.3390/electronics11192994

[6] Shi, C., Ding, M., Wang, L., & Pan, H. (2023). Learn by yourself: A feature-augmented self-distillation convolutional neural network for remote sensing scene image classification. *Remote Sensing*, *15*(23), 5620. https://doi.org/10.3390/rs15235620

[7] Schwabe, P., & Gao, W. (2016, May). Speeding up the number theoretic transform for faster ideal lattice-based cryptography. Microsoft Research Technical Report. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/05/RLWE-1.pdf

[8] Daftardar, A., Mo, J., Ah-kiow, J., Bünz, B., Karri, R., Garg, S., & Reagen, B. (2024). Need for zkspeed: Accelerating HyperPlonk for zero-knowledge proofs. *Cryptology ePrint Archive*, Paper 2024/276. https://eprint.iacr.org/2024/276

[9]Longa, P., & Naehrig, M. (2016). Speeding up the number theoretic transform for faster ideal lattice-based cryptography. *Cryptology ePrint Archive*, Paper 2016/504. https://eprint.iacr.org/2016/504

[10] Scott, M. (2017). A note on the implementation of the number theoretic transform. In *Cryptography and Coding* (Lecture Notes in Computer Science, Vol. 10343, pp. 247–258). Springer. https://doi.org/10.1007/978-3-319-71045-7_13

[11] Nozaki, A., Kojima, T., Nakamura, H., & Takase, H. (2024, December). A study on number theoretic transform acceleration on AMD AI engine. In *Proceedings of the International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC 2024)* (pp. 325–331). https://doi.org/10.1109/MCSoC64144.2024.00060

[12] Mirhoseini, A., Pham, H., Le, Q. V., Steiner, B., Larsen, R., Zhou, Y., Kumar, N., Norouzi, M., Bengio, S., & Dean, J. (2017). Device placement optimization with reinforcement learning. *arXiv preprint* arXiv:1706.04972. https://arxiv.org/abs/1706.04972

[13] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *arXiv preprint* arXiv:1706.03762. https://arxiv.org/abs/1706.03762

[14] Hafiz, S. M., et al. (2025). New tradeoffs and faster lattice-based cryptographic applications. *Cryptology ePrint Archive*, Paper 2025/768. https://eprint.iacr.org/2025/768

[15] Jansen, M. A. (2014). *Evaluation of intensity normalization methods for MR images* (Master's thesis). University of Twente, Enschede, The Netherlands. https://essay.utwente.nl/93885/1/Jansen_MA_ET.pdf

[16] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357. https://www.jair.org/index.php/jair/article/view/10302

[17] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. https://arxiv.org/abs/1708.02002

[18] Ghourabi, M., Mourad-Chehade, F., & Chkeir, A. (2024). Advancing cough classification: Swin transformer vs. 2D CNN with STFT and augmentation techniques. *Electronics, 13*(7), 1177. https://doi.org/10.3390/electronics13071177

[19] Mehta, A., & Rastegari, R. (2021). MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint* arXiv:2110.02178. https://arxiv.org/abs/2110.02178

[20] Roy, A. K. (2022, April). NeRF: 3D reconstruction from 2D images via ML models. *Medium*. https://autognosi.medium.com/nerf-3d-reconstruction-from-2d-images-via-ml-models-d0daaf9b465e