

(RESEARCH ARTICLE)



GPU-ENABLED FFANN INFERENCE FOR LOW-LATENCY VNF ORCHESTRATION IN MMWAVE 5G CORE NETWORK

Patrick Kipyegon Koech *, Livingstone Mwalugha Ngoo and Henry Macharia Kiragu

Faculty of Engineering and Technology, Multimedia University of Kenya, Nairobi, Kenya.

* Corresponding Author

ORCID Details

Patrick Kipyegon Koech: <https://orcid.org/0009-0004-2390-1554>

Livingstone Mwalugha Ngoo: // <https://orcid.org/0000-0003-0009-9701>

Henry Macharia Kiragu: <https://orcid.org/0009-0008-9976-2713>

World Journal of Advanced Engineering Technology and Sciences, 2026, 20(03), 051–066

Publication history: Received on 27 July 2026; revised on 05 September 2026; accepted on 07 September 2026

Article DOI: <https://doi.org/10.30574/wjaets.2026.20.3.0435>

Copyright © 2026 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0

ABSTRACT

Ultra-Reliable Low Latency Communication (URLLC) services in Millimeter-Wave Fifth Generation (mmWave 5G) core networks require efficient processing and real-time orchestration of network functions. Conventional Central Processing Unit (CPU)-based processing can introduce computational delays as telemetry volumes and network workloads increase. This study proposes a Graphics Processing Unit-enabled Feedforward Artificial Neural Network (GPU-FFANN) to predict Network Function (NF) overload and processing latency for proactive network orchestration. The proposed 11-64-32-2 FFANN architecture jointly performs NF overload classification and processing latency regression using a synthetically generated telemetry dataset of 300,000 samples, partitioned into 80% (240,000 samples) training, 10% (30,000 samples) validation, and 10% (30,000 Samples) testing. The model was implemented using PyTorch in a Computer Unified Device Architecture (CUDA)-enabled GPU computing environment. The GPU-FFANN performance was compared to a CPU/Virtual Central Processing Unit (VCPU)-based FFANN. The GPU-FFANN achieved 99.86% accuracy, 99.89% precision, 99.65% recall, and 99.77% F1-score for NF overload prediction, with a Mean Squared Error (MSE) of 0.2589 and a Mean Absolute Error (MAE) of 0.4055. Computational Inference performance was separately evaluated across inference batch sizes from 256 to 1,000,000 samples. At a batch size of 30,000, GPU-FFANN per-sample latency was approximately 5.0×10^{-8} s compared with 1.23×10^{-7} s for CPU/VCPU-FFANN, corresponding to a 59.35% reduction in per-sample inference latency and approximately 2.46 times faster inference. These results indicate that GPU-FFANN improves computational efficiency and supports responsive, proactive real-time VNF orchestration under high-traffic in mmWave 5G core networks.

Keywords: mmWave 5G, GPU-FFANN, Artificial Neural Network, Virtual Network Function (VNF) Orchestration, Latency reduction, URLLC

1 INTRODUCTION

Fifth-generation (5G) mobile networks are designed to support diverse applications that require high data rates, massive connectivity, and stringent latency and reliability requirements. Millimetre-wave (mmWave) communication

is an important enabler of high-capacity 5G services because of its wide available bandwidth and high spectral efficiency. However, supporting Ultra-Reliable Low-Latency Communication (URLLC) in mmWave 5G networks requires not only high-speed radio transmission but also efficient processing and resource management throughout the 5G Network Core (5GC) [1], [2]. Processing delays introduced by network functions virtualization, queuing, and service orchestration can therefore become significant contributors to overall service latency.

Network Functions Virtualization (NFV) enables network functions to be implemented as software-based Virtual Network Functions (VNFs), allowing network resources to be dynamically allocated according to traffic demand [3]. In a 5G core network, functions such as the Access and Mobility Management Function (AMF), Session Management Function (SMF), and User Plane Function (UPF) process large volumes of control and user-plane traffic. Under increasing or bursty traffic conditions, resource utilization may approach critical levels, resulting in queuing, processing delays, and network-function overload. Conventional orchestration approaches commonly rely on threshold-based reactive mechanisms, in which scaling or VNF instantiation is initiated after predefined resource thresholds have been exceeded. Such approaches can introduce additional delay because corrective actions occur after network degradation has already begun [4], [5].

Artificial Intelligence (AI) and Machine Learning (ML) have increasingly been investigated for predictive network management, resource allocation, traffic prediction, and network slicing [6–9]. AI-based approaches can analyse network telemetry and identify emerging congestion or resource-demand patterns before they result in severe performance degradation. In particular, Artificial Neural Networks (ANNs) can model nonlinear relationships between multiple network variables and performance indicators, making them suitable for predicting network-function overload and processing latency. Recent studies have demonstrated the application of AI for intelligent network slicing, VNF scaling, traffic management, and zero-touch network orchestration [6–9]. However, many existing approaches focus primarily on prediction accuracy or resource-management decisions, with comparatively less attention to the computational latency associated with executing the predictive models themselves.

Specific empirical studies illustrate both the progress and limitations of existing approaches. Alawe et al. developed a machine-learning-based traffic forecasting mechanism for 5G core-network scalability which demonstrated that forecast-based scaling can react to traffic changes more effectively than threshold-based approaches [10]. This work demonstrates the value of predictive intelligence for proactive VNF resource management, but it does not investigate GPU-accelerated neural-network inference as part of the orchestration process. Shafiei Ardestani et al. investigated Network Data Analytics Function (NWDAF)-enabled analytics and closed-loop automation, demonstrating how network data and ML services can support automated management in 5G networks [11]. Nevertheless, the computational latency of the predictive inference stage under changing processing workloads remains insufficiently characterized.

Traffic variability presents a further challenge. Li et al. examined 5G network-service resilience against short-timescale traffic variation and showed that abrupt traffic changes can threaten service requirements and resilience [12]. Such findings emphasize the importance of timely prediction and response when network conditions change rapidly. Ros et al. proposed priority-aware resource allocation for VNF deployment using graph reinforcement learning and demonstrated improvements in resource allocation for service-function chains [13]. Although this approach addresses intelligent VNF deployment, it does not focus on GPU-accelerated inference or quantify the computational latency of the intelligence used to make orchestration decisions.

GPU acceleration provides a potential solution to the computational component of this problem because GPUs provide highly parallel execution for matrix and tensor operations used by neural networks [15], [16]. However, GPU acceleration is workload-dependent. Jayanth et al. compared CPU, GPU, and Neural Processing Unit (NPU) execution for edge-AI workloads and reported that GPUs can provide substantial acceleration for sufficiently parallel workloads, including approximately $2.7\times$ faster than Long Short-Term Memory (LSTM) inference in their benchmark, but CPUs can remain competitive for less-parallel operations [14]. This finding indicates that the benefit of GPU inference should be evaluated as a function of workload size rather than assumed for every inference task.

Taken together, existing studies have established the value of predictive traffic management, network analytics, intelligent VNF deployment, and accelerated AI processing, although these capabilities have largely been investigated independently. A specific gap remains in integrating network-function overload and processing-latency prediction with GPU-accelerated inference for latency-aware orchestration in the mmWave 5G core. In particular, limited attention has been given to the computational latency inference of the predictive model for identifying the workload region in which GPU execution becomes advantageous relative to CPU-based neural inference.

To address this gap, this study proposes a GPU-enabled Feedforward Artificial Neural Network (GPU-FFANN) for latency-aware VNF orchestration in the mmWave 5G core network. The model jointly predicts network-function overload and processing latency from telemetry representing traffic, queue, resource utilization, and VNF-related conditions. The framework is implemented using PyTorch in a CUDA-enabled GPU platform, and GPU-FFANN inference

is compared with an equivalent CPU-FFANN implementation across different workload batch sizes [16]. The objective is to determine whether GPU acceleration can reduce computational inference latency sufficiently to support more responsive predictive orchestration. The study evaluates predictive performance separately from computational performance so that any benefit attributed to GPU execution is not confused with changes in the underlying neural-network predictive capability.

2 MATERIALS AND METHODS

2.1 Proposed GPU-FFANN Framework

The proposed framework integrates a Graphics Processing Unit-enabled Feedforward Artificial Neural Network (GPU-FFANN) into the management and orchestration process of a virtualized mmWave 5G core network. The framework uses network telemetry to predict Network Function (NF) overload and processing latency, thereby providing information that can support proactive VNF orchestration before severe congestion or processing degradation occurs [4], [5].

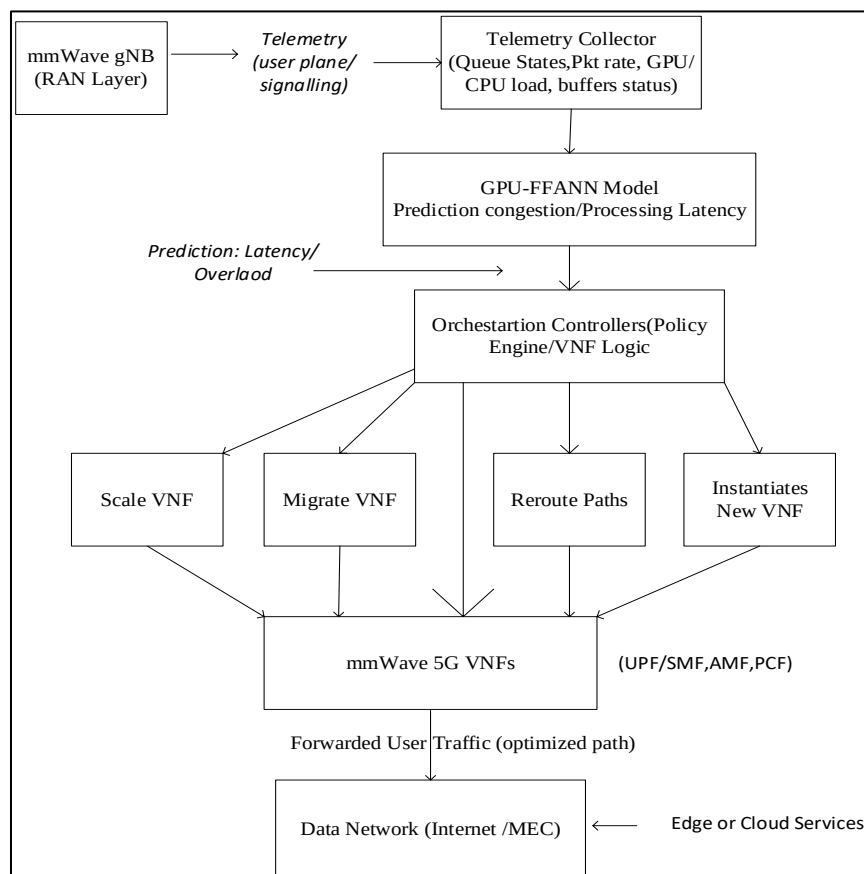


Figure 1: Proposed GPU-FFANN Framework for Proactive VNF Orchestration in mmWave 5G Core Networks

As illustrated in Figure 1, the framework comprises four principal components: telemetry collection, the GPU-FFANN prediction engine, the orchestration controller, and the virtualized NF infrastructure. The telemetry module collects operational measurements describing traffic conditions, queue status, resource utilization, and VNF processing characteristics [17]. These measurements are normalized and supplied to the GPU-FFANN prediction engine.

The prediction engine jointly estimates the probability of NF overload and the expected processing latency. The resulting predictions are provided to the orchestration controller, which can use them to support actions such as VNF scaling, instantiation, migration, and traffic redistribution. This predictive approach differs from conventional threshold-based orchestration, which generally initiates corrective actions after resource utilization or queue conditions have already deteriorated [4], [5]. GPU acceleration is applied to the neural-network computation, particularly the matrix operations performed during the forward pass.

The objective is to reduce the computational time required to obtain predictions while maintaining the predictive capability of the underlying FFANN. The framework therefore combines predictive network management with accelerated AI inference for latency-sensitive core-network applications [18].

2.2 FFANN Architecture and Mathematical Formulation

The proposed model is formulated as a multi-task supervised-learning FFANN that simultaneously performs NF-overload classification and processing-latency regression. Eleven telemetry variables are used as input features, while the two outputs represent the probability of NF overload and the predicted processing latency, respectively. Figure 2. The network adopts an 11-64-32-2 architecture, consisting of an 11-neuron input layer, two hidden layers

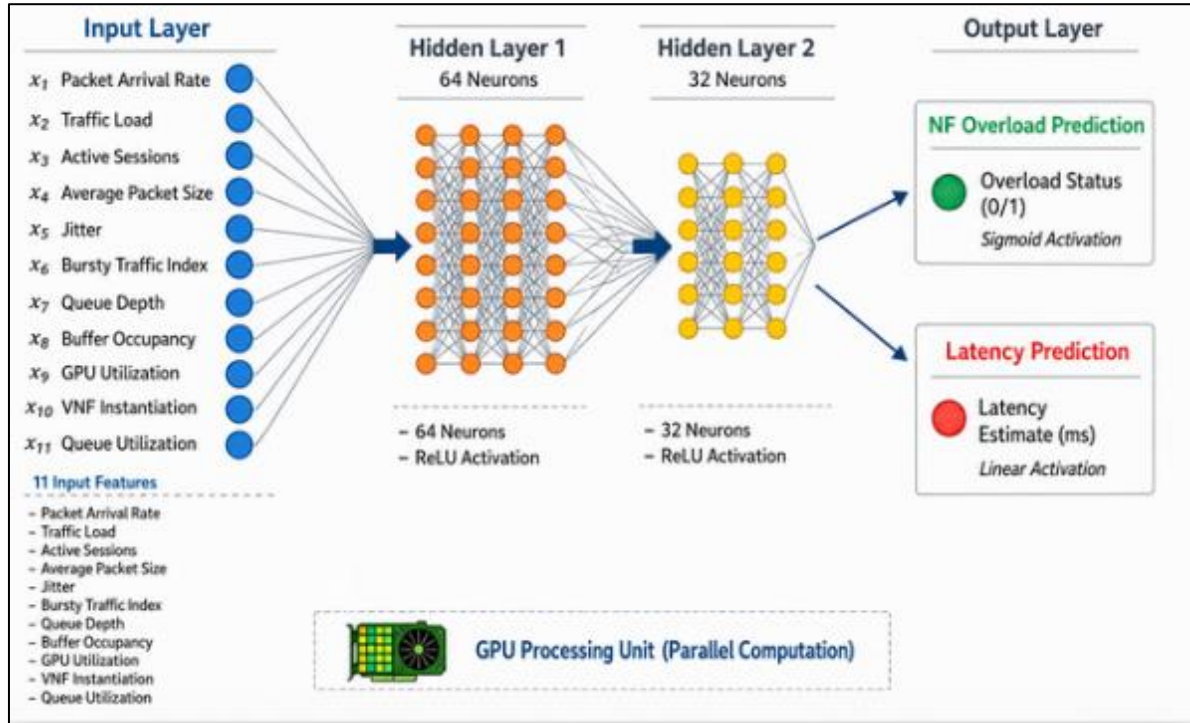


Figure 2: GPU-enabled FFANN Model Architecture

With 64 and 32 neurons, and a two-neuron output layer.

Mathematical formation

Let the normalized input vector be:

$$\mathbf{X} = [x_1, x_2, x_3 \dots \dots x_{11}]^T \quad \dots \dots \dots (2.1)$$

Where $[x_1, x_2, x_3 \dots \dots x_{11}]$ denote the normalized i^{th} features.

The Z-score normalization was used, and its formula is:

$$x_i^{norm} = \frac{x_i - \mu_i}{\sigma_i} \quad \dots \dots \dots (2.2)$$

Where x_i^{norm} the normalized feature, x_i is the unnormalized feature, μ_i is the mean of the feature i^{th} and σ_i is the standard deviation of the feature i^{th} .

The first hidden layer consist of 64 neurons that perform a nonlinear transformation of normalized telemetry input vectors. The transformation of the first hidden layer computed as:

$$z_j^{(1)} = \sum_{i=1}^{11} w_{ji}^{(1)} x_i^{(1)} + b_j^{(1)}, \quad j = 1, 2, 3 \dots 64 \quad \dots \dots \dots (2.3)$$

Where $w_{ij}^{(1)}$ is the weight connecting input neuron i^{th} to the hidden neuron j^{th} , and b_j is the bias of neuron j^{th} . The hidden neurons employ the Rectified Linear Unit (ReLU) activation function:

$$f(z_j^{(1)}) = \max(0, z_j^{(1)}) \quad \dots\dots\dots (2.4)$$

The second hidden layer contains 32 neurons; the transformation function is defined as:

$$z_k^{(2)} = \sum_{j=1}^{64} w_{kj}^{(2)} x_j^{(2)} + b_k^{(2)}, \quad k = 1, 2, 3, \dots, 32 \quad \dots\dots\dots (2.5)$$

where $w_{kj}^{(2)}$ is the weight connecting the first hidden layer neurons to the input $x_j^{(2)}$ of the second hidden layer, and b_k is the biasing term. The activation in the second hidden layer still uses ReLU activation defined as in equation (2.4).

The output layer contains two neurons corresponding to overload prediction and processing latency estimation.

$$z_m^{(3)} = \sum_{k=1}^{32} w_{mk}^{(3)} a_k^{(2)} + b_m^{(3)}, \quad m = 1, 2 \quad \dots\dots\dots (2.6)$$

The first output estimate network function overload using a sigmoidal activation function and the mathematical formulation is defined as:

$$z_{Overload}^{(3)} = \sum_{k=1}^{32} w_{1k}^{(3)} a_k^{(2)} + b_1^{(3)} \quad \dots\dots\dots (2.7)$$

It employs binary classification in the out predict network function overload. The activation function used is the sigmoidal function defined as:

$$\hat{y}_{Overload}^{(3)} = \sigma(z_{Overload}^{(3)}) \quad \dots\dots\dots (2.8)$$

Where: $\sigma = \frac{1}{1+e^{-z}}$ (2.9)

Where: $\hat{y}_{overload} \in [0, 1]$ represents the likelihood of overload. Values closer to 1 indicate a high probability of overload. The second output predicts the processing latency using a linear activation function. Linear transformation

$$z_{Latency}^{(3)} = \sum_{k=1}^{32} w_{2k}^{(3)} a_k^{(2)} + b_2^{(3)} \quad \dots\dots\dots (2.10)$$

The activation function is linear:

$$\hat{y}_{latency}^{(3)} = (z_{Latency}^{(3)}) \quad \dots\dots\dots (2.11)$$

proposed GPU-FFANN jointly optimizes overload classification and latency regression using a composite loss function. Binary Cross-Entropy (BCE) loss was employed for overload prediction.

$$L_{BCE} = -\frac{1}{N} \sum_{i=1}^N [y_{Overload} \log(\hat{y}) + (1 - y_{Overload}) \log(1 - \hat{y})] \quad \dots\dots\dots (2.13)$$

Where L_{BCE} is the Binary Cross-Entropy loss; $y_{overload} \in [0, 1]$ is the true overload label, $\hat{y}_{overload}$ represents the likelihood of overload, and N is the number of training Samples.

While Mean Square Error (MSE) was used for latency estimation.

$$L_{MSE} = \frac{1}{N} \sum_{i=1}^N (y_{latency} - \hat{y}_{latency}^{(3)})^2 \quad \dots\dots\dots (2.14)$$

The overall training objective of the FFANN was formulated as a weighted combination of the two loss functions:

$$L_{TOTAL} = L_{BCE} + L_{MSE} \quad \dots\dots\dots (2.15)$$

Where L_{TOTAL} is weight combination of the two loss functions.

During training, gradients of the composite loss were computed through backpropagation and optimized using Adaptive Moments Estimate (Adam) algorithm. The joint optimization strategy enables the network to learn both classification and regression tasks simultaneously, improving computational efficiency and reducing overall model complexity [19].

2.3 GPU-Based Inference

Following training, the learned FFANN parameters are transferred to the GPU for inference. Given a normalized telemetry batch, the forward-pass operations are executed using parallel GPU computation. Matrix multiplication and activation operations can therefore be processed concurrently, providing an opportunity to reduce inference time for sufficiently large workloads.

For comparison, an equivalent CPU-FFANN implementation was evaluated using the same network architecture and input representation. The comparison was designed to isolate the effect of the computational platform on inference performance. GPU execution was implemented using PyTorch and a CUDA-enabled NVIDIA T4 GPU platform [15], [16].

An important distinction was made between training batch size and inference workload batch size. Mini-batches were used during model training, whereas the inference experiment varied the number of samples processed in a workload batch to characterize scalability and identify the GPU/CPU performance crossover point. Thus, the large inference batches used for performance characterization represent workload-scaling experiments rather than a claim that every operational orchestration decision requires processing thousands of samples simultaneously.

2.4 Synthetic Telemetry Dataset Generation

A synthetic telemetry dataset was generated using Python and NumPy to emulate operational conditions in a virtualized mmWave 5G core network. The simulation represented three representative core-network functions: the Access and Mobility Management Function (AMF), Session Management Function (SMF), and User Plane Function (UPF). The generated telemetry comprised 11 variables describing traffic characteristics, queue conditions, resource utilization, and VNF processing behaviour. These variables include Packet arrival rate, traffic load, active sessions, packet size, jitter; burstiness index, queue depth, buffer occupancy, GPU utilization, VNF instantiation delay, and queue utilization.

These features represent network conditions that can influence congestion, resource utilization, and processing latency. Synthetic data generation provides a controlled environment in which different traffic and resource conditions can be systematically represented while avoiding dependence on operational network data that may be difficult to obtain. Random network load conditions provide variable telemetry dataset, and enable the model to learn nonlinear relationships between network state and the two prediction targets. Each generated sample represents a network-state snapshot containing the telemetry features and corresponding target values.

2.5 Data Processing and Partitioning

Before model was trained, the telemetry features were standardized using Z-score normalization [20]. The dataset was randomly divided into training, validation, and testing subsets using an 80:10:10 partition. The training subset was used to optimize the model parameters, while the validation subset was used to monitor model convergence, and support model-development decisions. The testing subset remained unseen during training and was reserved for final performance evaluation.

2.6 GPU-FFANN Training Configuration

The GPU-FFANN was implemented using PyTorch and trained using the Adam optimization algorithm. The model was configured as a multi-task network, using BCE loss for NF-overload classification and MSE loss for processing-latency regression. Adam was selected because of its adaptive learning-rate mechanism and established suitability for neural-network optimization [15].

The principal training parameters are summarized in Table 1.

Table 1: GPU-FFANN Training Configuration

Parameters	Configuration
Programming Language	Python 3.11
Framework	PyTorch
GPU Platform	Google Colaboratory (CUDA-enabled NVIDIA GPU T4)
ANN Architecture	11-64-32-2
Hidden Layer Activation Function	ReLU
Overload Output	Sigmoidal

Latency Output	Linear
Optimizer	Adam
Classification Loss	Binary Coded Entropy
Regression Loss	Mean Squared Error
Feature Normalization	Z-score Standardization
Data Split	80% training, 10 % Validation, 10 % Testing
Epochs	50
Training Batch Size	64

2.7 Inference Performance Evaluation

The computational performance of GPU-FFANN inference was evaluated against an equivalent CPU-FFANN implementation under identical model and data conditions. In addition, a traditional CPU-based rule-driven orchestration approach was included as a reference for computational decision overhead.

Inference workloads were evaluated over batch sizes ranging from 256 to 1,000,000 batch sizes. For each batch size, the average inference latency per sample and total inference time were measured. These measurements were used to characterize the computational scalability of the three approaches and determine the workload region in which GPU acceleration provides an advantage over CPU-based neural inference.

The average inference latency per sample was defined as the total inference time divided by the number of samples in the batch:

$$L_{avg}(B) = \frac{C_{fixed}}{B} + C_{variable}, \quad \dots\dots\dots (2.16)$$

$L_{avg}(B)$ is the average inference latency per sample, B denotes the batch size, C_{fixed} represents the fixed execution overhead per batch, including kernel launch, memory allocation, and host-to-device data transfer, and $C_{variable}$ denotes the average computation time required for processing a single sample.

To characterize the observed scalability, average latency was represented using the model

Similarly, total inference time was represented by

$$T_{total}(B) = C_{fixed} + B C_{variable} \quad \dots\dots\dots (2.17)$$

Where

$T_{total}(B)$ denotes the total inference time for a batch size B , C_{fixed} represents the fixed initialization overhead, and $C_{variable}$ (Intercept) represents the average computational cost per sample. This formulation separates workload-independent overhead from the variable computational cost associated with processing additional samples.

2.8 Performance Evaluation Metrics

Model prediction performance was evaluated separately from computational inference performance. For NF-overload classification, accuracy, precision, recall, and F1-score were calculated. Accuracy measures the proportion of correctly classified samples, precision measures the proportion of predicted overload cases that were correctly identified, recall measures the proportion of actual overload cases correctly detected, and the F1-score provides the harmonic mean of precision and recall. Processing-latency prediction was evaluated using MSE, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). Lower MSE, MAE, and RMSE indicate lower prediction error, whereas a higher R^2 indicates better agreement between predicted and observed latency values.

Computational performance was evaluated using average inference latency per sample, total inference time, latency reduction, and inference speedup. Formulation to calculate percentage latency reduction achieved by GPU-FFANN relative to CPU/VCPU-FFANN is derived from the model equation (2.16) for average latency per sample as function of batch size for each setup. The percentage latency reduction is defined as:

$$LR(B) = \frac{\left(\frac{C_{fixed}^{CPU}}{B} + C_{Variable}^{CPU}\right) - \left(\frac{C_{fixed}^{GPU}}{B} + C_{variable}^{GPU}\right)}{\left(\frac{C_{fixed}^{CPU}}{B} + C_{Variable}^{CPU}\right)} \times 100 \quad \dots\dots\dots (2.18)$$

Where $LR(B)$ represent latency reduction percentages, $L_{CPU}(B)$, and $L_{GPU}(B)$ denotes the average inference latency per sample of the GPU-FFANN and CPU-FFANN as a function of batch size, respectively.

Using the latency model development in equation 2.16, substitute for each case,

The latency reduction

$$LR(B) = \frac{L_{CPU}(B) - L_{GPU}(B)}{L_{CPU}(B)} \times 100\% \quad \dots\dots\dots (2.19)$$

Where:

$$L_{CPU}(B) = \frac{C_{fixed}^{CPU}}{B} + C_{Variable}^{CPU} \quad \dots\dots\dots (2.19a)$$

$$L_{GPU}(B) = \frac{C_{fixed}^{GPU}}{B} + C_{variable}^{GPU} \quad \dots\dots\dots (2.19b)$$

These measures allow the predictive performance of the proposed model to be considered separately from the computational benefit obtained through GPU acceleration.

3 RESULTS AND DISCUSSION

The performance of the proposed GPU-enabled Feedforward Artificial Neural Network (GPU-FFANN) was evaluated based on training loss, network-function (NF) overload prediction, processing-latency prediction, and inference scalability. The GPU-FFANN performance was compared with a CPU-based FFANN to determine the computational benefit of GPU acceleration under different inference batch sizes. The conventional rule-based CPU/vCPU orchestration approach was also considered as a baseline for computational comparison. The results show that GPU acceleration provides the greatest benefit when the inference batch size is sufficiently large to amortize GPU initialization, memory-transfer, and kernel-launch overheads.

3.1 GPU-FFANN Training Performance

The training performance of the GPU-FFANN and CPU-FFANN implementations was evaluated over 50 training epochs. Figure 3 presents the training-loss profiles obtained during model training.

For both implementations, the loss decreased rapidly during the initial training epochs, followed by gradual stabilization. In the GPU-based experiment, the loss decreased from approximately 5.92 at the beginning of training to about 0.31 by the second epoch. The CPU-based implementation similarly decreased from approximately 5.70 to 0.52 over the same period. After approximately 10 epochs, both implementations approached a stable loss level of approximately 0.26.

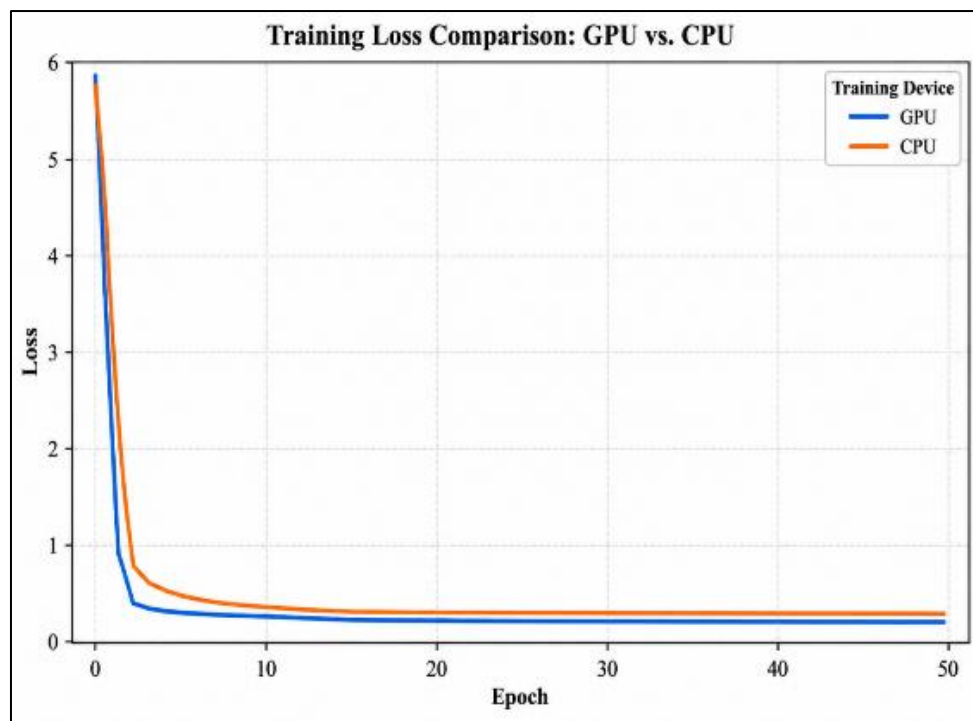


Figure 3: Training loss comparison of the GPU-FFANN and CPU-FFANN over 50 epochs.

The rapid decrease in loss during the early epochs indicates that the FFANN was able to learn the relationship between the telemetry features and the target variables effectively. The subsequent stabilization indicates convergence of the optimization process. The observed difference between the two training curves should not be interpreted as evidence that GPU execution inherently improves model accuracy. GPU acceleration primarily reduces computational execution time; differences between independently trained models can also result from initialization, numerical operations, and training-run variability.

Overall, the results indicate that the proposed architecture trained successfully using on both GPU and CPU platform, while GPU computing provides a more suitable computational platform when larger datasets or repeated model training are required.

3.2 Network-Function Overload Prediction

The classification capability of the proposed model was evaluated using accuracy, precision, recall, and F1-score. The results are summarized in Table 2.

Table 2: Classification performance of the FFANN implementations

Model Performance	GPU-ANN	CPU-ANN
Accuracy	99.86%	99.78%
Recall:	99.65%	99.35%
Precision:	99.89%	99.90%
F1-Score:	99.77%	99.62%

The GPU-FFANN achieved an accuracy of 99.86%, precision of 99.89%, recall of 99.65%, and F1-score of 99.77%. The CPU-FFANN achieved corresponding values of 99.78%, 99.90%, 99.35%, and 99.62%, respectively. Thus, both implementations provided highly accurate overload classification.

The confusion matrix for the GPU-FFANN is shown in Figure 4. The model correctly classified 21,180 non-overload samples and 8,779 overload samples. Only 10 false-positive and 31 false-negative predictions were observed among the 30,000 evaluated samples.

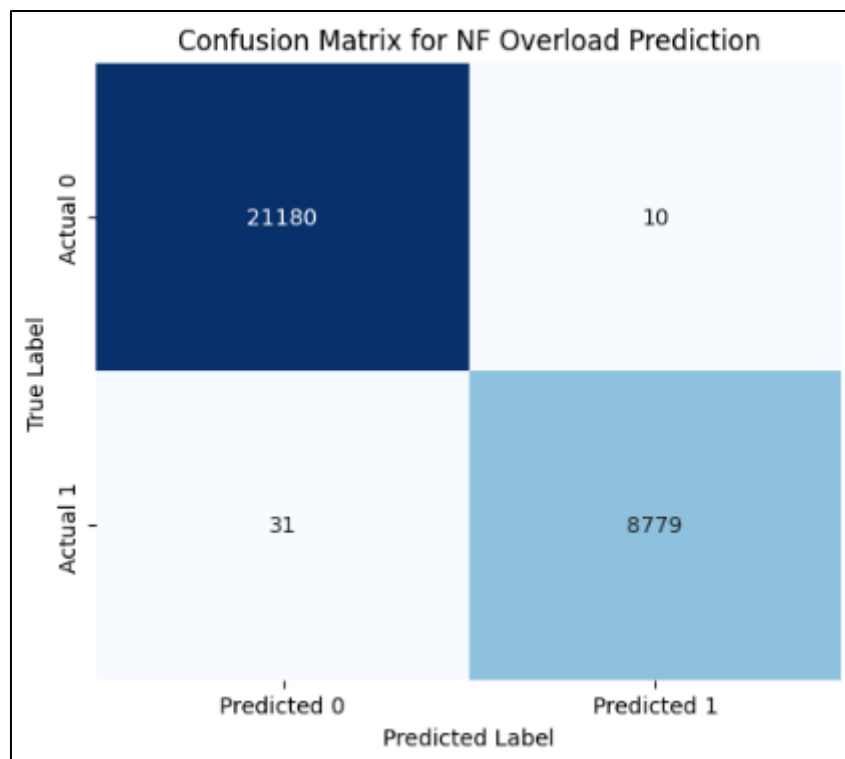


Figure 4: Confusion matrix for NF-overload prediction using GPU-FFANN.

The resulting relatively small number of misclassifications demonstrates that the selected telemetry features provided sufficient information for distinguishing overloaded from non-overloaded network functions.

The high recall is particularly important for proactive orchestration because false negatives represent overloaded network functions that may not be identified in time for corrective action. The GPU-FFANN recall of 99.65% indicates that the model correctly identified the overwhelming majority of overloaded cases. The small differences between GPU-FFANN and CPU-FFANN classification metrics should not be attributed directly to GPU acceleration. Both implementations use the same FFANN formulation; therefore, hardware acceleration primarily affects computational execution rather than the underlying predictive capability.

3.3 Processing-Latency Prediction

In addition to overload classification, the FFANN was evaluated for regression-based prediction of network-function processing latency. Mean Squared Error (MSE) and Mean Absolute Error (MAE) were used as the primary regression metrics. The results are presented in Table 3.

Table 3: Processing Latency Prediction Performance

Metric	GPU Run	CPU Run	Comment
MSE	0.2589	0.2657	Average squared difference between predictions and actual values. Lower is better.
MAE	0.4055	0.4108	Average absolute difference between predictions and actual values. Lower is better.

The GPU-FFANN obtained an MSE of 0.2589 and an MAE of 0.4055, compared with 0.2657 and 0.4108 for the CPU-FFANN. The relatively small difference between the two implementations indicates comparable predictive performance. Figure 5 presents the relationship between predicted and true processing-latency values.

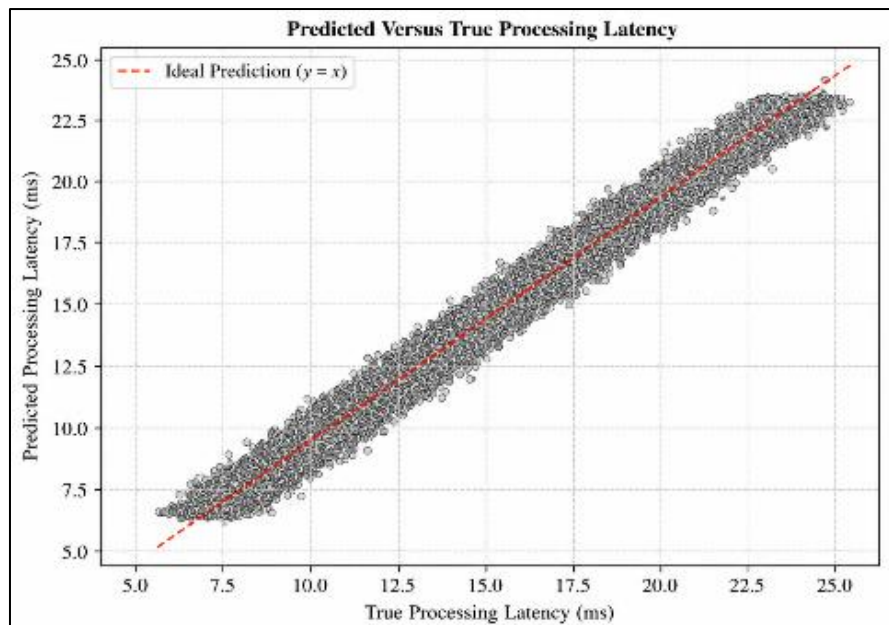


Figure 5: Predicted versus true processing latency

The close correspondence between the predicted and observed values indicates that the FFANN successfully captured the relationship between network telemetry and processing latency.

The regression results further demonstrate that the proposed multi-task architecture can simultaneously support overload classification and latency estimation. This is important for proactive orchestration because an orchestration controller can use both predicted overload state and estimated processing latency when determining whether additional network-function resources should be instantiated or allocated.

It should be noted that GPU acceleration itself does not intrinsically improve the predictive accuracy of a neural network. The primary contribution of GPU execution in this study is computational acceleration. Any small difference in prediction metrics between the GPU and CPU implementations may result from independent training runs and numerical differences rather than from the GPU hardware itself.

3.4 Average Inference Latency

The inference scalability of the FFANN was evaluated using progressively larger inference workloads. Figure 6 shows the average inference latency per sample as a function of inference batch size.

At relatively small batch sizes less 10^3 batch size, the GPU-FFANN did not immediately provide lower per-sample latency than the CPU implementation. This behaviour is attributable to fixed GPU-related overheads, including kernel-launch, memory-management, and data-transfer operations. When only a small number of samples are processed, these fixed costs are distributed over fewer samples and therefore have a greater influence on the measured latency.

As the inference workload increased, the average GPU inference latency decreased substantially because the fixed computational overhead was amortized over a larger number of samples. The GPU-FFANN approached and subsequently surpassed the CPU-FFANN at a crossover point of approximately 2,048 samples. Beyond this region, the GPU implementation demonstrated a progressively greater computational advantage.

This result is important because it demonstrates that GPU acceleration is workload dependent. The benefit is limited for isolated or very small inference tasks but becomes increasingly significant when large volumes of telemetry must be processed. Such conditions are relevant to high-traffic 5G core environments in which telemetry observations may be processed in batches.

The traditional rule-based CPU/vCPU baseline exhibited low computational overhead because it performs simple threshold-based operations rather than neural-network inference. Consequently, it can have lower raw execution time than either FFANN implementation. However, the rule-based approach does not provide the predictive classification and regression capabilities of the proposed GPU-FFANN. The comparison therefore highlights a trade-off between simple reactive processing and computationally accelerated predictive intelligence.

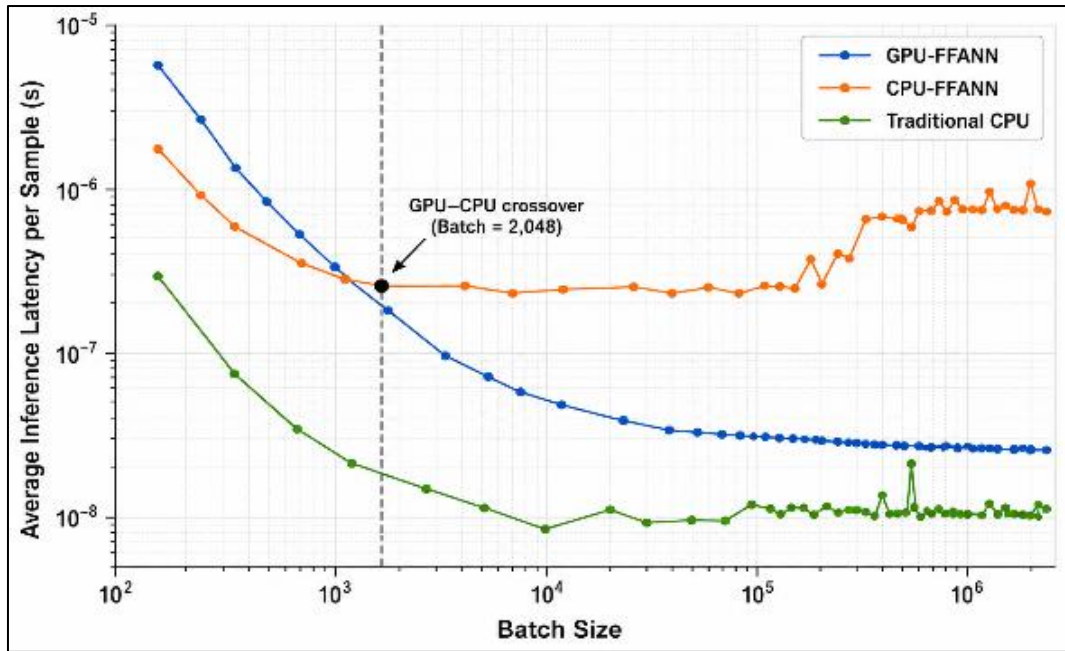


Figure 6: Average Inference Latency per Sample versus Batch Size

3.5 Total Inference Time

Figure 7. presents the total inference time as the inference batch size increases from 256 to 1,000,000.

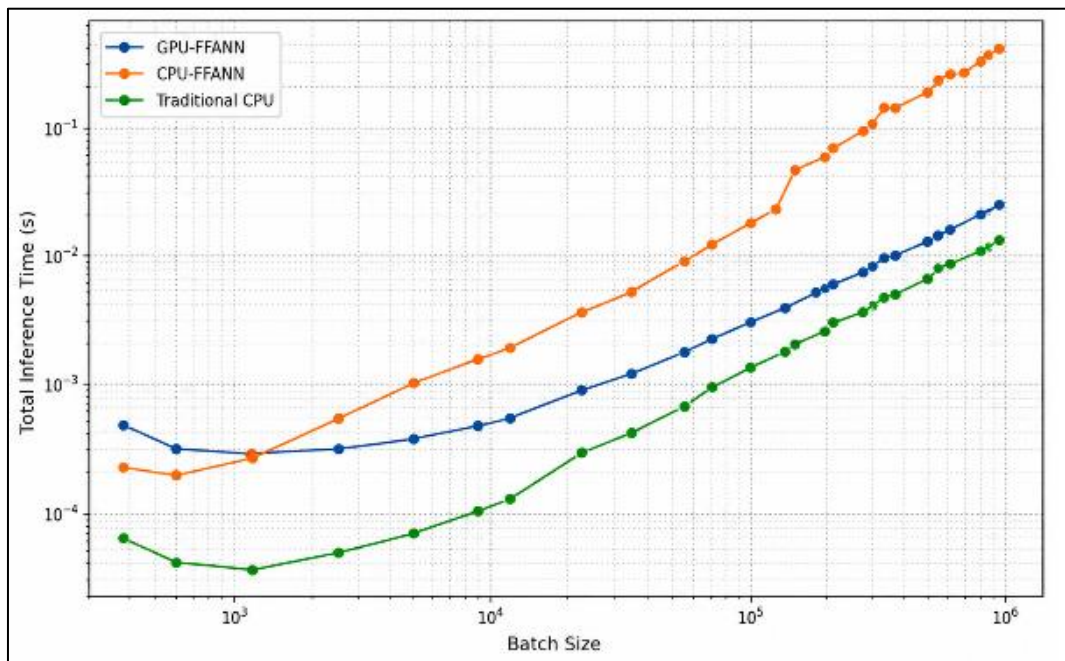


Figure 7: Total Time Inference as Function of Batch Size for GPU-FFANN, CPU-FFANN and Traditional CPU-Based Orchestration.

For small workloads, the GPU-FFANN incurs a fixed computational overhead associated with GPU execution. As the number of samples increases, however, the GPU becomes increasingly efficient because its parallel processing capability can be utilized more effectively. The total inference-time curve therefore demonstrates greater scalability for the GPU-FFANN than for the CPU-FFANN at large workloads.

The observed behaviour confirms that GPU acceleration is particularly beneficial when a large number of telemetry samples must be processed within a short period. This characteristic is relevant to predictive network management,

where telemetry streams can generate substantial volumes of data that must be evaluated before orchestration decisions are made.

The results also reinforce the distinction between computational inference latency and complete network end-to-end latency. The measurements reported in this study quantify the computational inference component. They therefore demonstrate the potential of GPU acceleration to reduce the processing time associated with predictive orchestration, but they should not be interpreted as a direct measurement of total 5G end-to-end service latency unless the complete network path is explicitly evaluated.

3.6 Statistical Analysis of Inference Performance

To quantify the relationship between inference workload and execution time, the measured results were fitted using fixed-overhead and variable-cost models. For average per-sample inference latency vs batch size, at approximate 30,000 batch size, the following observation deduce in Table 4. The fitted parameters for the average inference latency are presented in Table 4.

Table 4: Regression parameters for average inference latency

Model	Fixed Overhead Per Batch (ms)	Per-Sample Processing Time (μs)	R ²
GPU-FFANN	0.490	0.025	0.9576
VCPU-FFANN	0.093	0.466	0.8906
Traditional VCPU	0.0794	0.023	0.9181

The GPU-FFANN has a larger fixed overhead than the traditional CPU/vCPU approach but a substantially lower variable per-sample cost than the CPU-FFANN. This explains why GPU acceleration becomes advantageous as the workload increases. The high R² value for the GPU-FFANN indicates that the fitted model provides a good representation of the observed latency trend.

Table 5: Regression Parameters for Total Inference Time

Model	Fixed Overhead per Batch (ms)	Processing time per Sample within Batch(us)	R ²
GPU-FFANN	0.406	0.030	0.9991
VCPU-FFANN	0.1679	0.554	0.9647
Traditional VCPU	0.1156	0.016	0.9181

The GPU-FFANN achieved an R² value of 0.9991, indicating an extremely strong linear relationship between total inference time and workload size under the tested conditions. More importantly, its fitted variable processing cost was substantially lower than that of the CPU-FFANN. Consequently, the GPU implementation becomes increasingly advantageous as the number of processed samples grows. The traditional CPU/VCPU implementation retains a low computational cost because its rule-based decision mechanism is considerably simpler than neural-network inference. However, its computational simplicity should not be interpreted as equivalent predictive functionality. The FFANN implementations provide learned prediction of overload and processing latency, which enables predictive rather than purely reactive orchestration.

3.7 Quantitative Inference-Latency Reduction

The computational benefit of GPU acceleration was quantified using the percentage reduction. The percentage latency reduction based on average inference latency per sample is defined in equation (2.19)

$$LR(B) = \frac{L_{CPU}(B) - L_{GPU}(B)}{L_{CPU}(B)} \times 100\% \quad \dots\dots\dots (3.1)$$

Where: For the large batch size ($B \geq 30,000$), representative of high-density URLLC traffic, the measured average inference latency values were

$$L_{GPU} \approx 5.0 \times 10^{-8} \text{ s}$$

$$L_{CPU} \approx 1.23.0 \times 10^{-7} \text{ s}$$

Substituting these values in equation (3.1)

$$LR(\%) = \frac{(1.23 \times 110^{-7} - 5.0 \times 10^{-8})}{(1.23 \times 10^{-7})} \times 100\% \dots\dots\dots (3.2)$$

$$\approx 59.35\%$$

The above expression demonstrated that the latency reduction depends on both the batch size and the difference in fixed overhead and per-sample processing cost between the CPU and GPU implementations.

For small batch size ($B \approx 256$), the fixed GPU overhead associated with kernel launch and memory transfer dominates the execution time, resulting in little or no latency reduction. Consequently, GPU acceleration provides limited benefit under low traffic conditions.

As the batch size increases ($10^3 \leq B \leq 10^4$), the fixed overhead is progressively amortized and inference latency of the GPU-FFANN approaches that of the CPU-FFANN. The crossover point occurs at approximately 2,048 samples, beyond which GPU acceleration becomes advantageous.

For the large batch size ($B > 30,000$), representative of high-density URLLC traffic, the measured average inference latency values were

The corresponding speedup was calculated as:

$$Speedup = \frac{L_{CPU}}{L_{GPU}} \dots\dots\dots (3.3)$$

Which gives approximately 2.46 times faster inference for the GPU-FFANN at this workload.

The reduction becomes more pronounced as the workload increases, confirming that the primary benefit of GPU acceleration arises when sufficient parallel work is available to amortize fixed GPU overheads.

The total-inference-time regression further indicated an asymptotic computational latency reduction of approximately 58.9% for GPU-FFANN relative to CPU-FFANN. The close agreement between this value and the approximately 59.35% reduction measured at a batch size of 30,000 provides additional evidence that GPU acceleration substantially reduces the computational cost of neural-network inference under high-volume workloads.

These findings support the use of GPU-accelerated FFANN inference as a computational component of latency-aware VNF orchestration. Nevertheless, the reported reduction represents inference latency, rather than complete end-to-end 5G network latency. The extent to which this computational improvement translates into end-to-end latency reduction depends on additional factors, including telemetry collection, communication between orchestration components, VNF scaling or migration time, queuing delay, and network transport delay.

4 CONCLUSION

This study proposed and evaluated a GPU-enabled Feedforward Artificial Neural Network (GPU-FFANN) for predictive network-function management in latency-sensitive mmWave 5G core networks. The proposed framework jointly performs network-function overload classification and processing-latency prediction using network telemetry. GPU acceleration was subsequently evaluated to determine its effect on the computational latency of FFANN inference under different workload sizes.

The results demonstrated that the proposed FFANN achieved high predictive performance for network-function overload detection. The GPU-FFANN obtained 99.86% accuracy, 99.89% precision, 99.65% recall, and 99.77% F1-score. Processing-latency prediction also produced low prediction errors, with an MSE of 0.2589 and MAE of 0.4055. These results indicate that the selected telemetry features and FFANN architecture can effectively support predictive network monitoring.

The computational evaluation showed that GPU acceleration provides its greatest benefit at sufficiently large inference workloads. At a batch size of 30,000, the measured GPU-FFANN per-sample inference latency was approximately 5.0×10^{-8} s compared with 1.23×10^{-7} s for the CPU-FFANN. This corresponds to an approximately 59.35% reduction in per-sample inference latency and approximately 2.46 times faster inference. The regression analysis similarly indicated

an asymptotic computational latency reduction of approximately 58.9%. The results therefore demonstrate that GPU acceleration can substantially improve the scalability of predictive neural-network inference when large volumes of telemetry must be processed.

An important finding is that GPU acceleration does not necessarily provide an advantage for very small inference workloads because GPU initialization, memory-transfer, and kernel-launch overheads can dominate execution time. The measured results showed that the GPU-FFANN became advantageous relative to the CPU-FFANN at approximately the 2,048-sample workload region. This confirms that the effectiveness of GPU acceleration is workload dependent.

The study therefore establishes GPU-FFANN inference as a promising computational mechanism for supporting latency-aware and proactive VNF orchestration in mmWave 5G core networks. However, the measured latency reduction represents neural-network inference latency and should not be interpreted directly as an equivalent reduction in complete end-to-end network latency. Actual end-to-end improvement would additionally depend on telemetry acquisition, orchestration communication, VNF instantiation or scaling, queuing, and network transport delays.

Future studies should validate the proposed framework using real-world 5G core-network telemetry rather than exclusively synthetic data. Experimental deployment on an operational or emulated 5G core network would also allow the complete orchestration pipeline and end-to-end latency to be measured. Further investigation should consider larger and more heterogeneous neural-network architectures, dynamic workload scheduling, multi-GPU or edge-GPU deployment, and comparison with additional AI-based orchestration approaches. Such validation would provide stronger evidence regarding the practical applicability of GPU-accelerated predictive orchestration for future high-throughput and latency-sensitive 5G and beyond-5G networks.

STATEMENTS ON COMPLIANCE WITH ETHICAL STANDARDS

Acknowledgments

The authors would like to express their sincere appreciation to the Faculty of Engineering and Technology, Multimedia University of Kenya, for providing the academic environment and resources that supported this study. The authors also gratefully acknowledge the guidance and constructive contributions of the research supervisors throughout the development of this work.

Disclosure of conflict of interest

All authors declare that there is no conflict of interest.

Statement of Ethical Approval

This study was conducted under a research licence issued by the National Commission for Science, Technology and Innovation (NACOSTI) (License No: NACOSTI/P/25/4179208), Kenya. The study involved simulation-based analysis using synthetically generated network telemetry data and did not involve human participants, animals, biological materials, or identifiable personal data. Therefore, ethical approval from a human-subjects ethics review committee was not applicable.

Statement of Informed Consent

Informed consent was not applicable because the study did not involve human participants or the collection, processing, or use of personal or identifiable human data.

REFERENCES

- [1] X. Lin, "An overview of 5G Advanced evolution in 3GPP Release 18," IEEE Communications Standards Magazine, vol. 6, no. 3, pp. 77–83, 2022, doi: 10.1109/MCOMSTD.0001.2200001.
- [2] 3GPP, System architecture for the 5G System (5GS), TS 23.501 V18.5.0, Release 18, 2024.
- [3] ETSI, Network Functions Virtualizations (NFV); Architectural Framework, ETSI GS NFV 002 V1.2.1, 2014.
- [4] H. O. Otieno, B. Malila, and J. Mwangama, "Deployment and management of intelligent end-to-end network slicing in 5G and beyond 5G networks: A systematic review," IEEE Access, vol. 12, pp. 190411–190433, 2024, doi: 10.1109/ACCESS.2024.3516051.
- [5] R. Verma and K. M. Sivalingam, "Design and analysis of VNF scaling mechanisms for 5G-and-beyond networks using federated learning," IEEE Access, vol. 12, pp. 129826–129843, 2024, doi: 10.1109/ACCESS.2024.3458437.

- [6] R. Mustafovski, G. Marinova, B. Qehaja, E. Hajrizi, S. Gagic, and V. Guliashki, "AI-driven network optimization for the 5G-to-6G transition: A taxonomy-based survey and reference framework," *Future Internet*, vol. 18, no. 3, Art. no. 155, 2026, doi: 10.3390/fi18030155.
- [7] E. Lykakis, I. O. Vardiambasis, and E. Kokkinos, "Data traffic prediction for 5G and beyond: Emerging trends, challenges, and future directions: A scoping review," *Electronics*, vol. 14, no. 23, Art. no. 4611, 2025, doi: 10.3390/electronics14234611.
- [8] F. E. Subhan, A. Yaqoob, C. H. Muntean, and G.-M. Muntean, "A survey on artificial intelligence techniques for improved rich media content delivery in a 5G and beyond network slicing context," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 2, pp. 1427–1487, 2025, doi: 10.1109/COMST.2024.3442149.
- [9] M. A. Hossain, A. Xiang, A. Kiani, T. Saboorian, J. Kaippallimalil, and N. Ansari, "AI-assisted E2E network slicing for integrated sensing and communication in 6G networks," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 10627–10634, 2024, doi: 10.1109/JIOT.2023.3326761.
- [10] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, "Improving traffic forecasting for 5G core network scalability: A machine learning approach," *IEEE Network*, vol. 32, no. 6, pp. 42–49, 2018, doi: 10.1109/MNET.2018.1800104.
- [11] F. Shafiei Ardestani, N. Saha, N. Limam, and R. Boutaba, "Towards NWDAF-enabled analytics and closed-loop automation in 5G networks," *arXiv preprint arXiv:2505.06789*, 2
- [12] R. Li, B. Decocq, A. Barros, Y.-P. Fang, and Z. Zeng, "Estimating 5G network service resilience against short timescale traffic variation," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, 2023, doi: 10.1109/TNSM.2023.3269673.025.
- [13] S. Ros, S. Kang, T. Iv, I. Song, P. Tam, and S. Kim, "Priority-aware resource allocation for VNF deployment in service function chains based on graph reinforcement learning," *Computers, Materials & Continua*, vol. 83, no. 2, pp. 1649–1665, 2025, doi: 10.32604/cmc.2025.062716.
- [14] R. Jayanth, N. Gupta, and V. Prasanna, "Benchmarking edge AI platforms for high-performance ML inference," *arXiv preprint arXiv:2409.14803*, 2024.
- [15] NVIDIA Corporation,, *CUDA C++ Programming Guide*, Release 13.3 ,25 JUN 2026.
- [16] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* 32, pp. 8024–8035, 2019.
- [17] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain, "Network slicing for 5G: Challenges and opportunities," *IEEE Internet Computing*, vol. 21, no. 5, pp. 20–27, 2017, doi: 10.1109/MIC.2017.3481355.
- [18] S. Muruganandam, R. Joshi, P. Suresh, N. Balakrishna, K. Hari, and S. V. Manikanthan, "A deep learning based feed forward artificial neural network to predict the K-barriers for intrusion detection using a wireless sensor network," *Measurement: Sensors*, vol. 25, Art. no. 100613, 2023, doi: 10.1016/j.measen.2022.100613.
- [19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.
- [20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.