Latency Prediction in Distributed Control Systems Using FPGA-Accelerated Neural Networks

Lei Zhang[1*], Xiaofei, Li[1], Baozhong, Huang[2], Hao, Wei[2]
[1]Fujian Fuqing Nuclear Power Co., LTD., Fujian, China
[2]Atomhorizon Electric (Jinan) Co., Ltd., Jinan, Shandong, China
E-mail: zhanglei20241219@163.com
[*]Corresponding author

Keywords: field programmable gate array, neural network, distributed control system, latency prediction, model optimization

Received: April 28, 2025

To improve the real-time performance and stability of distributed control systems in complex and dynamic environments, this study introduces a delay prediction and optimization model. The model is built on an integrated architecture that combines Long Short-Term Memory (LSTM) neural networks with Field Programmable Gate Array (FPGA). A sliding window input mechanism is used, where a recent sequence of historical delay data serves as input to forecast short-term system response latency. To support efficient hardware deployment, the LSTM model was quantized to 8-bit fixed-point precision. Additionally, the FPGA implementation was optimized through the design of a parallel pipelined architecture and an on-chip cache scheduling mechanism. These enhancements significantly improve inference speed and resource utilization. Experiments were conducted using the Electric Transformer Temperature (ETT) time-series dataset series. The proposed model was compared against several representative approaches. Evaluation metrics included prediction accuracy, response latency, system throughput, resource consumption, task success rate, and overall stability. On the ETT-small-m3 dataset, the optimized model achieved a task completion rate of 99.699%, a system throughput of 1,424.082 tasks per second, and an average response time of 0.247 seconds. These results surpassed those of the baseline models across most performance indicators. To evaluate generalization, five-fold cross-validation was performed. Analysis of variance (ANOVA) was also conducted to confirm the statistical significance of the results, with all p-values below 0.05, ensuring the reliability of the experimental findings. Despite its strengths, the model has limitations in certain reliability metrics. For example, the mean time between failures was slightly lower than that of the Multi-Agent System-Based Distributed Control Model (MAS-DCM), suggesting reduced stability under high-pressure or high-load conditions. Moreover, the model's adaptability to scenarios involving multi-source heterogeneous data has not been comprehensively tested. In summary, this study presents a deployable, efficient, and scalable architecture for intelligent delay prediction. The proposed solution provides a practical approach to delay modeling and performance optimization in smart control systems. It holds strong potential for real-world applications and lays a solid foundation for future research and development in this area.

Povzetek: Opisan je FPGA-pospešen model z LSTM za napovedovanje zakasnitev v porazdeljenih krmilnih sistemih, ki izboljša odzivnost, točnost napovedi in pretočnost pod industrijskimi pogoji.

1 Introduction

As industrial automation systems, power dispatch networks, intelligent manufacturing lines, and smart transportation infrastructure continue to evolve rapidly, Distributed Control Systems (DCSs) have become essential for enabling multi-point coordinated control. These systems are now widely deployed in critical industrial sectors, including chemical process management, power grid operations, and rail transit systems [1–3]. DCSs operate by distributing sub-control units across various locations to handle data acquisition, real-time computation, and control signal transmission. This decentralized architecture enhances system flexibility and scalability. However, it also introduces a more complex control environment. One of the most significant challenges lies in managing the uncertainty and dynamic variability of communication delays and task scheduling latency. These factors have become major bottlenecks affecting system stability, response time, and control accuracy [4]. In response, some advanced DCSs have begun incorporating basic delay detection and management strategies. Common approaches include threshold-based response mechanisms and static delay modeling. However, these methods typically rely on fixed parameters or simple linear models, making them insufficient for predicting nonlinear or sudden delay fluctuations in complex network environments [5]. Improving the accuracy and timeliness of delay prediction has therefore emerged as a key objective in the ongoing effort to optimize DCS performance.

To address this challenge, this study proposes a prediction acceleration framework designed for embedded distributed control scenarios. The framework integrates two complementary technologies: the nonlinear modeling capabilities of neural networks for time-series prediction, and the high-performance, low-latency characteristics of Field Programmable Gate Array (FPGA). Neural networks have already demonstrated superior performance compared to traditional models and support vector machines in various time-sensitive domains such as financial forecasting and fault prediction [6–8]. Their primary advantage lies in their ability to automatically learn long-term dependencies and adapt to non-stationary trends in sequential data. This makes them particularly effective at handling diverse and heterogeneous delay patterns. Meanwhile, FPGA offers a powerful solution for accelerating neural network inference through parallel pipelined hardware architectures. Unlike software-based deployments, which often suffer from computational bottlenecks, FPGA implementations enable real-time processing and are especially well-suited for industrial environments with stringent response time requirements. By combining these two technologies, the proposed framework not only enhances the real-time performance of delay prediction but also supports lightweight and efficient deployment on resource-constrained control nodes. This integration forms a closed-loop optimization system that spans from model design to runtime execution.

Accordingly, this study presents an FPGA-accelerated delay prediction system based on Long Short-Term Memory (LSTM) neural networks. Its effectiveness is validated through experiments on several representative control datasets, with comprehensive evaluations focusing on prediction accuracy, latency, and resource efficiency.

2 Related work

Numerous scholars have explored delay-related challenges in DCSs from various perspectives. Dai et al. identified that communication delays in DCSs primarily stem from network transmission and task scheduling, noting that the unpredictability of these delays significantly undermines the timeliness and precision of control tasks [9]. Caballero-Águila and Linares-Pérez examined the impact of delay on the consistency of industrial automation control systems. They proposed a delay-tolerant task scheduling algorithm, which notably enhanced system robustness within specific operational limits [10]. FPGA has gained popularity in control systems due to their low latency and versatile capabilities. Tiong et al. highlighted the advantages of FPGA's parallel processing and reconfigurability, demonstrating their effectiveness in meeting the stringent real-time requirements of embedded control applications [11]. Similarly, Sridevi and Saifulla implemented a real-time signal processing architecture using FPGA technology. Their approach significantly improved the operational efficiency of industrial control systems by enabling immediate data stream processing [12]. Neural networks have shown strong performance in time-series prediction and are considered promising tools for addressing delay issues in DCSs. Herrera et al. demonstrated that LSTM neural networks effectively captured temporal dependencies in delay data, leading to substantial improvements in prediction accuracy [13]. Likewise, Olabi et al. proposed a latency prediction model based on convolutional neural networks (CNNs), which achieved real-time performance by extracting multi-dimensional features from the delay sequences [14].

Although many researchers have explored delay prediction in control systems, most existing methods suffer from limited accuracy, poor real-time performance, or challenges in deployment on embedded systems. Some studies have employed traditional statistical models or machine learning approaches and have achieved certain results on small-scale datasets. However, these methods often exhibit poor robustness and weak generalization when applied to large-scale, non-stationary, and noise-prone DCSs. In recent years, deep neural networks—particularly recurrent neural networks—have emerged as mainstream tools for time-series prediction and have been adopted in some edge computing tasks. Nonetheless, their deployment efficiency and real-time inference capabilities remain constrained by hardware limitations. A comparative overview of existing delay prediction models is presented in Table 1:

Table 1: Comparison of existing delay prediction models

Model Name	Core Method	Dataset Used	Performance Metrics	Key Limitations
AutoRegressive Integrated Moving Average (ARIMA)	Linear time-series model with autoregressive and moving average components	Power load monitoring data	MAE, MSE	Suitable only for stationary time series; struggles with nonlinear or abrupt delay patterns
Support Vector Machine (SVM)	Support vector regression; suitable for small-sample prediction	Industrial control network delay data	RMSE, R^2	Strong dependence on feature selection; complex hyperparameter tuning; limited generalization
Random Forest for Latency Prediction (RF-LP)	Ensemble regression using multiple decision trees	Industrial control task scheduling logs	MAE, MAPE	High training overhead; sensitive to noise and outliers
Deep Autoregressive Model (DeepAR)	Probabilistic model based on LSTM	E-commerce server request data	ND, NRMSE	Complex structure; long inference time;

				high deployment barrier
Temporal Convolutional Network (TCN)	One-dimensional causal convolutional neural network for modeling long-term dependencies	Communication network link delay sequences	MAE, Latency Error	High accuracy but rigid architecture; not suitable for embedded deployment

Current mainstream methods generally fall into three categories:

(1) Statistical approaches, such as ARIMA, which are effective for linear and stationary data but fail to capture complex patterns;

(2) Traditional machine learning methods, such as SVM and RF, which are sensitive to input features and lack real-time capabilities;

(3) Deep learning models, including LSTM and TCN, which offer high prediction accuracy but pose significant challenges in deployment and demand substantial hardware resources.

Furthermore, most studies rely on static logs or simulated datasets for evaluation, lacking support from industrial-grade, dynamically changing data sources. This limits the objectivity and generalizability of model assessments. More critically, there remains a gap in the literature regarding a comprehensive delay prediction system that integrates neural network models with hardware platforms—one that balances accuracy, real-time performance, and resource efficiency. To address this gap, this study proposes an LSTM-based prediction model deployed on an FPGA platform. The goal is to provide high-accuracy, low-latency, and embedded-compatible delay prediction capabilities for DCSs.

3 Related theory and model design

3.1 Latency prediction

Latency refers to the time it takes for a message or data packet to travel from one end of a network to the other. It typically includes four components: propagation latency, processing latency, transmission (sending) latency, and queuing latency [15]. The overall latency can be expressed as the sum of these four elements. In most practical scenarios, propagation latency and transmission latency are the primary contributors. For longer messages, transmission latency becomes the dominant factor, while propagation latency plays a greater role for shorter messages. More precisely, latency is defined as the time interval between the arrival of the first bit of a packet at a router and the departure of the last bit from that router [16–18]. During testing, latency is commonly measured as the time interval from when a test instrument sends a data packet to when the packet is received. It is closely related to the packet length and is typically evaluated within the throughput range of the router port. Testing beyond this range is considered uninformative and not reflective of actual performance conditions [19]. Time-series prediction is a critical technique for latency forecasting. Its theoretical foundation includes feature analysis of historical data, the design of prediction models, and the selection of appropriate evaluation metrics. Latency data often exhibit complex temporal patterns, such as trends, seasonality, and random fluctuations. By analyzing these patterns, it is possible to uncover the underlying structure of delay variations [20–22]. Various models are available for time-series forecasting, including both traditional statistical methods and modern machine learning approaches. Among these, neural networks have emerged as the preferred choice for latency prediction due to their ability to model nonlinear temporal dependencies effectively. Common evaluation metrics for latency prediction models include Mean Squared Error (MSE), Mean Absolute Error (MAE), and prediction accuracy, all of which help quantify model performance in terms of accuracy and robustness [23].

The theoretical framework for latency prediction plays a pivotal role in optimizing the performance of DCSs. In this study, a neural network-based latency prediction model is proposed to meet the practical demands of distributed systems. The model leverages hardware acceleration to improve both the real-time performance and the predictive accuracy of latency estimation.

3.2 Design of the DCS

The design theory of DCS) encompasses several key areas, including system architecture, communication mechanisms, and task scheduling strategies [24–26]. To meet the stringent demands of complex control environments—such as real-time responsiveness, reliability, and adaptability—multi-level theoretical models and technical frameworks are integrated to construct a highly coordinated and efficient system structure. Among these components, the communication mechanism plays a central role in enabling collaborative operations among distributed nodes [27]. Its theoretical foundation primarily involves real-time communication protocols, network scheduling and priority management, and distributed consistency models. Real-time control applications impose strict latency requirements on communication mechanisms. To address this, protocols such as Controller Area Network (CAN), Ethernet for Control Automation Technology (EtherCAT), and Industrial Ethernet are commonly employed, offering high reliability and low-latency data transmission. Network scheduling and priority management theories focus on assigning appropriate priorities to tasks and efficiently allocating data transmission resources. This ensures that time-critical tasks are executed promptly, maintaining system performance under varying workloads. Distributed consistency theory emphasizes the importance of maintaining data consistency across nodes. In scenarios where multiple nodes share task states and data, ensuring consistency is critical to preventing execution failures and systemic errors.

3.3 Neural networks and FPGAs integration architecture

To achieve high-precision, low-latency prediction of task response delays in DCSs, this study proposes a prediction model based on a LSTM neural network architecture accelerated by an FPGA platform. A comprehensive co-optimization of both software and hardware components has been carried out. The model not only focuses on accurately predicting latency but also emphasizes its feedback effect on task scheduling and overall system stability, thereby facilitating holistic DCS performance optimization.

During DCS operation, system response latency can be broken down into the following components:

(1) Sending latency – the time from when a task is dispatched by the scheduler to when it reaches the network module;

(2) Propagation latency – the time required for a data packet to travel through the communication channel, influenced by bandwidth and network topology;

(3) Queuing latency – the waiting time a task experiences in the scheduling buffer before processing;

(4) Processing latency – the time a node takes to compute and complete the assigned task.

These delays directly impact task completion times, which in turn affect scheduling order, priority allocation, and the overall stability of the system. In this work, total response latency is modeled as a unified prediction target, with its constituent components implicitly embedded within the input features. The neural network is designed to automatically learn the underlying relationships among these factors. The goal of the model is to predict the total latency of a task, which is defined as the time from its reception to completion. This prediction is based on historical task data and system states collected within a sliding time window. The results can support decisions on task prioritization and resource allocation.

To simulate the dynamics of task loads and delay variations in a DCS, this study employs the publicly available Electricity Transformer Temperature (ETT) - small dataset. Features such as timestamps, voltage, current, and load power serve as proxies for the state variations in a real-world task scheduling system. The response delay is defined as the difference between the task's reception time and its feedback time, and this value serves as the prediction target. Input features include a 10-dimensional sequence of historical task response times, task type and priority (encoded via one-hot encoding), and current system metrics such as GPU utilization, queue length, and network congestion level. Samples are constructed using a sliding window approach, where each sample input consists of 10 time steps of sequential information used to predict the latency at the next time step. To enhance training stability and improve model generalization, all feature values are normalized to the [0,1] range using Min-Max Scaling. Additionally, outliers are removed using the Interquartile Range (IQR) method to reduce noise and improve robustness.

The neural network developed in this study adopts a single-layer LSTM architecture to model the dynamic behavior of sequential tasks. The input layer has 16 dimensions, comprising 10 dimensions of historical delay data and 6 dimensions representing system state. The LSTM hidden layer consists of 128 units, and the output layer includes a single neuron with a linear activation function to produce continuous prediction values. Internally, the LSTM uses Sigmoid and Tanh activation functions to regulate the behavior of the forget gate, input gate, and output gate. The model is trained using the MSE as the loss function and the Adam optimizer with a learning rate of 0.001. Training is performed in mini-batches of size 64 for up to 100 epochs. An EarlyStopping strategy is applied to halt training if validation performance fails to improve over consecutive epochs. The model outputs the predicted delay for the next time step, which can be used to support task scheduling, task ordering, and node priority adjustment in distributed control environments. To ensure efficient deployment and real-time performance in practical DCS scenarios, the trained LSTM model is deployed on the Xilinx Alveo U250 FPGA acceleration platform. Logic synthesis, placement and routing, and bitstream generation are conducted using the Vivado platform. The model structure is translated into multiple hardware modules capable of parallel execution. The matrix multiplications within the LSTM are accelerated using the FPGA's internal Digital Signal Processing (DSP) blocks, while the nonlinear activation functions are implemented using lookup tables to minimize resource consumption. Intermediate state variables are stored in Block RAM (BRAM) to support time-step state propagation. Neural network weights are quantized into 8-bit fixed-point representation using the following quantization equation:

$$\hat{\mathrm{W}} = \mathrm{round}(\mathrm{W} \cdot 2^{q})/2^{q} \tag{1}$$

W refers to the weight matrix of the neural networks; $\hat{\mathrm{W}}$ represents the quantized weight matrix; q is the number of quantization bits, and then the model mainly performs matrix multiplication. This quantization strategy effectively reduces the bit width and latency of multiplication operations, thereby improving inference efficiency. Furthermore, the data input/output interface and control modules are connected to the main control module via a shared bus, enabling real-time interaction between the input prediction stream and the feedback results.

The optimized model architecture in this study is presented in Figure 1:

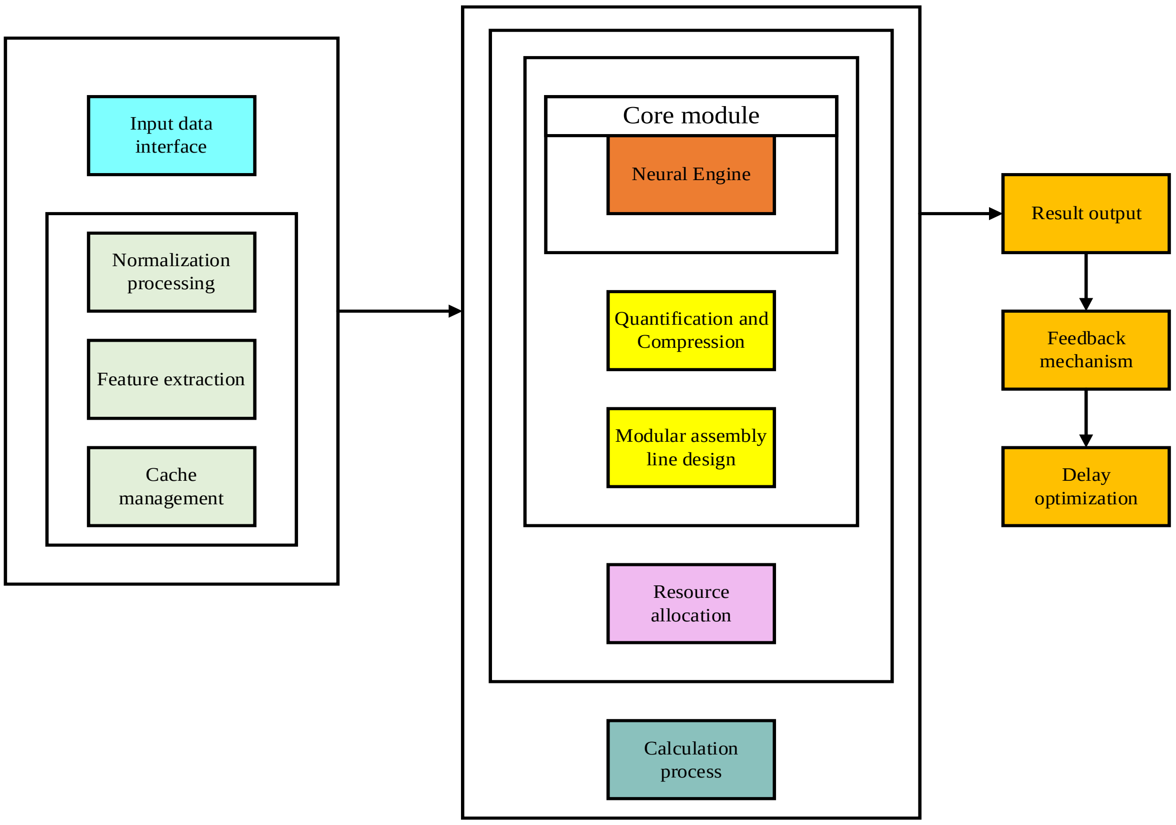

Figure 1: Architecture of the optimized model

The "Neural Engine" is the core computational unit of the system. Its internal structure includes the LSTM gating unit computation module, quantization and dequantization modules, state register module, and activation function look-up table (LUT) module. The LSTM Core Unit performs time-step calculations in a pipeline, with the gating units operating in parallel. The Quantization Module compresses the input signals and weights. The State Register caches the target for the current step, enabling state transitions across time steps. The activation function is quickly implemented by the LUT module to perform nonlinear mapping. The entire module is connected to the external bus, supporting high-speed data exchange with the controller and scheduler. The module structure is highly configurable, supporting parameter reuse and adjustable precision settings, making it adaptable to various deployment scenarios.

To further enhance the model's adaptability and the system's intelligence, an online update mechanism is introduced after the model is deployed. When the real response latency from the control system deviates significantly from the model's prediction, the system will locally fine-tune the model parameters via backpropagation, using the Adam optimization algorithm for incremental updates. The update strategy is described by Equations (2) to (4).

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L(W) \tag{2}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla L(W))^2 \tag{3}$$

$$\mathrm{W} = \mathrm{W} - \alpha \frac{m_t}{\sqrt{v_t}+\epsilon} \tag{4}$$

The exponential decay average of momentum and gradient squared is denoted as m_t and v_t ; $\nabla L(W)$ indicates the gradient for weights; α refers to the learning rate; β_1 and β_2 are exponential decay coefficients; ϵ represents a random number, to prevent the denominator from becoming 0. It can be observed that the integrated design of FPGA and neural networks has established a solid mathematical foundation at both the theoretical and implementation levels, ensuring the real-time, efficient, and accurate performance of the system. The pseudo code of the optimized model is as follows:

```
# Pseudocode: LSTM-Based Latency Prediction with FPGA Deployment
Initialize:
  Load quantized LSTM weights W_q from on-chip memory
  Initialize hidden state h_0 and cell state c_0 to zeros
  Set time window size w = 10
  Define learning rate α, momentum decay β1, β2, ε
# Inference Loop (on FPGA)
Function LSTM_Predict(x_seq):
  h, c ← h_0, c_0
  for t in 1 to w:
    # Gate calculations (in parallel on FPGA)
    f_t ← sigmoid(W_f * x_t + U_f * h + b_f)
    i_t ← sigmoid(W_i * x_t + U_i * h + b_i)
    o_t ← sigmoid(W_o * x_t + U_o * h + b_o)
    g_t ← tanh(W_c * x_t + U_c * h + b_c)
    # Cell and hidden state update
    c ← f_t ∘ c + i_t ∘ g_t
    h ← o_t ∘ tanh(c)
    # Final prediction
  y_pred ← W_out * h + b_out
```

```
        return y_pred
    # On-chip Online Update (optional)
    Function Online_Update(y_true, y_pred):
        loss ← (y_true - y_pred)^2
        # Compute gradients (simplified)
        grad_W ← ∂loss/∂W using backpropagation
        m_t ← β1 * m_{t-1} + (1 - β1) * grad_W
        v_t ← β2 * v_{t-1} + (1 - β2) * (grad_W)^2
        # Update weights
        W_q ← W_q - α * m_t / (sqrt(v_t) + ε)
        Store updated W_q to on-chip memory
    # Main control loop
    While system is running:
        Collect current input sequence x_seq from sensors
and system state
        y_pred ← LSTM_Predict(x_seq)
        Send y_pred to scheduling module
        Receive feedback y_true
        If |y_true - y_pred| > δ:
            Online_Update(y_true, y_pred)
```

4 Experimental design

To validate the applicability and performance of the proposed neural network and FPGA-accelerated delay prediction model in DCSs, this study uses the ETT dataset as the experimental data source. This dataset was collected by the power monitoring systems of two regions in a Chinese province. It records the oil temperature, load conditions, and external disturbance features of power transformers during actual operation, spanning two years with good continuity and high-frequency characteristics. It is a typical industrial-grade time series dataset.

The original features of the ETT dataset include: date, timestamp, load power, oil temperature, ambient temperature, target forecast values, and six external load influence factors (such as power flow direction, voltage level, etc.). These variables are highly correlated with task scheduling delays, response time fluctuations, and environmental disturbance intensity in DCSs, and thus serve as a good proxy for the operational state of a real DCS. Therefore, the ETT dataset structurally aligns with and maps to the delay prediction task of a DCS, making it suitable for training neural network prediction models. The ETT dataset is available for download from the official website (https://github.com/zhouhaoyi/ETDataset/blob/main/README_CN.md). Additionally, the dataset is clearly divided into multiple scenarios and supports sequence modeling at different time granularities (minute-level, hour-level), facilitating comparative research on algorithms with varying complexities and precision requirements. For this study, three subsets at the minute-level were selected, as described below:

(1) ETT-small-m1: Region 1, record per minute.

(2) ETT-small-m2: Region 2, record per minute.

(3) ETT-small-m3: Region 3, record per minute.

The dataset contains approximately 70,080 records per minute (2 years x 365 days x 24 hours x 60 minutes). The study also set the experimental environment, as exhibited in Table 2:

Table 2: Experimental environment

Configuration	**type**
Computing device	Dell PowerEdge R740
Central Processing Unit (CPU)	Intel Xeon Gold 6230, 2.1GHz, 20 cores
Graphics Processing Unit (GPU)	NVIDIA Tesla V100, 16GB memory
Memory	128GB DDR4
Storage device	Samsung PM981a NVMe SSD
Prediction Target	Task Response Latency (0.01 ms)
Neural Network Output	Predicted Total Response Latency of the Next Task Scheduling Cycle
Prediction Target Deployment Use	Used for Dynamic Task Scheduler Sorting and Control Strategy Optimization, Achieving Real-time Performance Improvement of the DCS System
FPGA development board	Xilinx Alveo U250

The optimized model parameters in this study were determined by considering several factors. These include the dynamic characteristics of task response latency in DCSs, resource limitations in the deployment environment, and the real-time requirements of the prediction target. Firstly, regarding the input features, the model uses 6 dimensions, covering key control variables such as historical task latency, task type encoding, CPU utilization, cache queue length, network link load rate, and scheduling priority. These features provide a comprehensive reflection of the factors influencing response latency during task scheduling and resource allocation in control systems. In terms of network structure, the model adopts a two-layer LSTM architecture, with 128 hidden units per layer. This design is based on the characteristics of task latency in DCSs, which exhibit medium- to short-term temporal dependencies. The two-layer structure effectively models the system's dynamics, while the 128-dimensional state representation achieves a good balance between prediction accuracy and FPGA deployment resource efficiency. The output layer has 1 dimension, using a linear activation function. The output value represents the predicted response latency for the next time step (in milliseconds), making it suitable for direct use by the controller in task prioritization and scheduling strategy decisions. During training, the Adam optimizer is used with an initial learning rate of 0.001, a first-order momentum decay coefficient of 0.9, and a second-order momentum decay coefficient of 0.999. This setting has shown good convergence in various non-stationary time series modeling tasks, making it especially suitable for short-term fluctuations and local trend changes in industrial control data. Additionally, gradient clipping with a threshold set to 5 is applied during training to suppress gradient explosion caused by anomalous data, improving the overall stability and robustness of the training process. The batch size is set to 64, balancing training efficiency and model generalization, while also considering the cache scheduling efficiency and throughput requirements when running on the FPGA platform. In the comparative experiments, to ensure fairness and comparability in model evaluation, the implementation environment and parameter settings for all comparison models were standardized. Except for the proposed optimized model deployed on the FPGA platform, the other models ran on the same hardware and software platform, using the same data processing flow, and the same training and validation dataset splitting strategy.

The Distributed Model Predictive Control (DMPC) model was executed on an Ubuntu 20.04 system environment. The control algorithm was implemented using Python and primarily used to simulate the predictive scheduling process of distributed tasks across multiple nodes. The control horizon was set to 50 time units, with a control step size of 10 steps, and a receding horizon optimization strategy was employed. The model used historical system states, task loads, and resource utilization (RU) as inputs and performed rolling prediction and latency estimation based on a quadratic programming solver. In the experiments, the DMPC model was deployed on a workstation equipped with an Intel Core i7-11700 processor and 16GB of RAM. The Multi-Agent System-Based Distributed Control Model (MAS-DCM) was jointly developed using Python and Java Agent Development Environment (JADE), simulating information exchange and local prediction behaviors among multiple agents. Each agent maintained an independent task estimator, utilizing local observations for latency prediction and scheduling feedback. The model was implemented within the Robot Operating System (ROS) framework, supporting task state broadcasting and multi-node coordination. Communication between agents followed a publish/subscribe mechanism, with each scheduling synchronization cycle set to 500 ms. The model was deployed on an NVIDIA Jetson Xavier NX platform (ARM Cortex-A57 CPU + 8GB RAM), simulating a low-power edge deployment environment. The Field Programmable Gate Array-Based Deep Neural Network Acceleration Model (FPGA-DNNAM) followed recent best practices for edge deployment. It used a four-layer multilayer perceptron (MLP) for latency prediction, with a network structure of [Input dimension 16 → 64 → 32 → 1]. All hidden layers used Rectified Linear Unit (ReLU) activation functions, while the output layer employed a linear function. The model was trained using the PyTorch 1.13 framework over 100 epochs with the support of an NVIDIA RTX 3060 GPU. The Adam optimizer was used with an initial learning rate of 0.001 and a batch size of 64. After training, the model was quantized and deployed using the Xilinx Vitis AI toolchain and was ultimately implemented on a Xilinx ZCU104 FPGA board to enable high-performance inference. To ensure fairness, all models mentioned above used the same ETT-small dataset, following a unified normalization process and sample construction pipeline.

This study did not employ conventional statistical methods such as confidence interval estimation, hypothesis testing, or significance analysis during experimental design and performance evaluation. The core objective of the study was not to perform statistical inference on minor performance differences among models. Instead, the goal was to develop a deployable and scalable optimization framework for task latency prediction in DCSs. The study focuses more on the practical effectiveness of predictions, system adaptability, and deployment feasibility under real-world engineering conditions. A comprehensive evaluation of model performance was conducted using a multi-dimensional metric system from the perspective of actual operational scenarios. Unlike typical machine learning studies that focus on statistical significance, this study emphasizes real-world usability and consistent improvements in system performance. It prioritizes practical enhancements rather than probabilistic reasoning over numerical differences. Additionally, the study does not provide a systematic modeling or categorization of all failure modes in control systems. The latency prediction model itself does not directly address low-level device fault diagnosis or control logic protection. Failures in DCS environments are often the result of complex interactions between hardware conditions, communication errors, and environmental disturbances. As a result, scenario-agnostic analysis is less meaningful. This study focuses on the impact of latency prediction deviation on task scheduling efficiency and system stability. It indirectly reflects the model's robustness using indicators such as task failure rate and mean time between failures (MTBF). More comprehensive fault modeling and coordinated control mechanisms are proposed as future research directions.

In summary, the omission of statistical significance testing and system-level fault modeling in this study is not a result of insufficient methodological rigor. Instead, it is a deliberate choice, driven by the study's scope, system-level objectives, and application-focused approach. The goal is to ensure that the study remains centered on the development of a deployable, verifiable latency prediction model and its integration into practical control systems.

5 Experimental comparison of latency prediction of the dcs based on FPGA and neural networks

5.1 Performance evaluation of the latency prediction model

To comprehensively evaluate the practical effectiveness of the proposed model in DCSs, this study categorizes performance evaluation metrics into four key dimensions: accuracy, robustness, timeliness, and resource efficiency. Within each dimension, two representative evaluation metrics are selected, resulting in a total of eight core indicators. This multi-faceted approach enables a systematic analysis of the model from various perspectives. In the accuracy dimension, the Mean Absolute Error (MAE) and MSE are adopted as primary evaluation metrics. MAE measures the average level of absolute error in the prediction process, offering intuitive interpretability. MSE, by squaring the prediction errors, emphasizes larger deviations and effectively reflects the model’s ability to control errors at outlier points. This dimension provides a direct assessment of the model's fitting accuracy with respect to task response latency and serves as the foundation for evaluating prediction quality. The robustness dimension includes the Root Mean

Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). RMSE, as the square root of MSE, maintains the same unit as the original data and is suitable for presenting the overall fluctuation intensity. MAPE considers the ratio of prediction error to the actual value, reflecting the model's stability and adaptability across different data scales, particularly under varying load intensities in DCS scenarios. This dimension is designed to evaluate the model's stable prediction capability under data perturbations and unexpected conditions. For the timeliness dimension, to verify whether the model is suitable for deployment in latency-sensitive control systems, two indicators are introduced: Mean Prediction Latency (MPL) and Maximum Prediction Latency (Max-PL). MPL represents the average time required for a single prediction, while Max-PL indicates the longest inference delay that may occur under extreme computational pressure. These metrics are directly related to the model's real-time inference capability and serve as core criteria for determining its viability in online DCS applications. In the resource efficiency dimension, RU and Throughput Per Second (TPS) are used as the primary metrics. RU denotes the proportion of logical resources consumed by the model when deployed on FPGA or embedded systems, providing a critical reference for deployment feasibility under resource constraints. TPS reflects the number of prediction tasks the model can complete per unit of time, directly indicating its execution efficiency under high-concurrency scheduling scenarios. This dimension comprehensively evaluates the deployment cost and runtime performance-to-cost ratio of the model.

The comparison results of accuracy dimensions are presented in Figure 2:

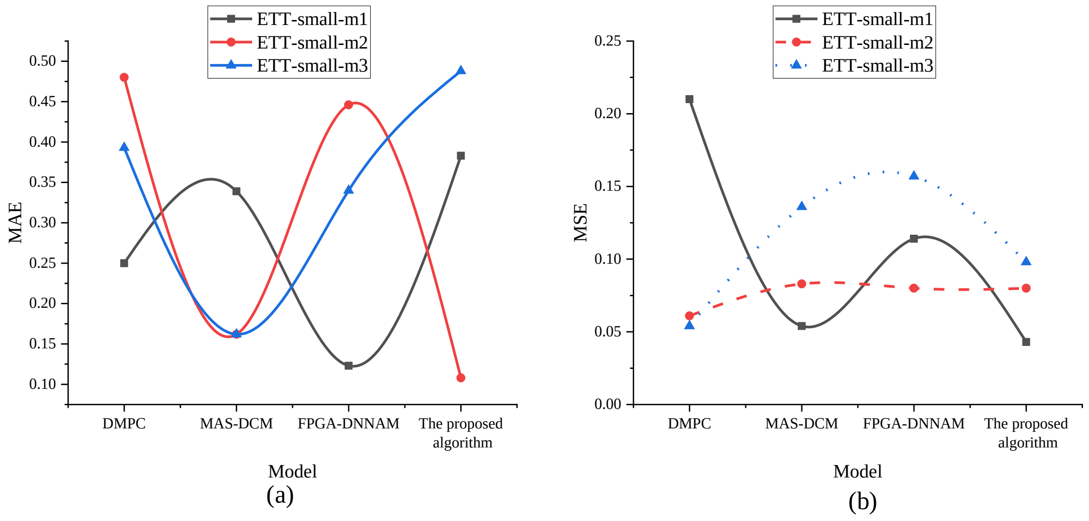

Figure 2: Accuracy dimension experimental result ((a): MAE; (b): MSE)

Figure 2 shows that the proposed optimized model achieves a MAE score of 0.108 on the ETT-small-m2 dataset. This significantly outperforms other models, such as DMPC's 0.480 and FPGA-DNNAM's 0.446. However, on the ETT-small-m1 and ETT-small-m3 datasets, the proposed model's MAE is 0.383 and 0.488, respectively. These values are higher than FPGA-DNNAM's 0.123 and 0.340, indicating slightly inferior performance. In terms of MSE, the proposed optimized model performs well across all datasets. It achieves scores of 0.043 (ETT-small-m1), 0.080 (ETT-small-m2), and 0.098 (ETT-small-m3). These are significantly lower than DMPC (e.g., 0.210 on ETT-small-m1) and MAS-DCM (e.g., 0.136 on ETT-small-m3). The proposed model has a notable advantage in the MSE metric, indicating better control over overall prediction errors. While its MAE performance is slightly lower than FPGA-DNNAM on some datasets, its overall performance remains stable and accurate. The comparison results for robustness are shown in Figure 3.

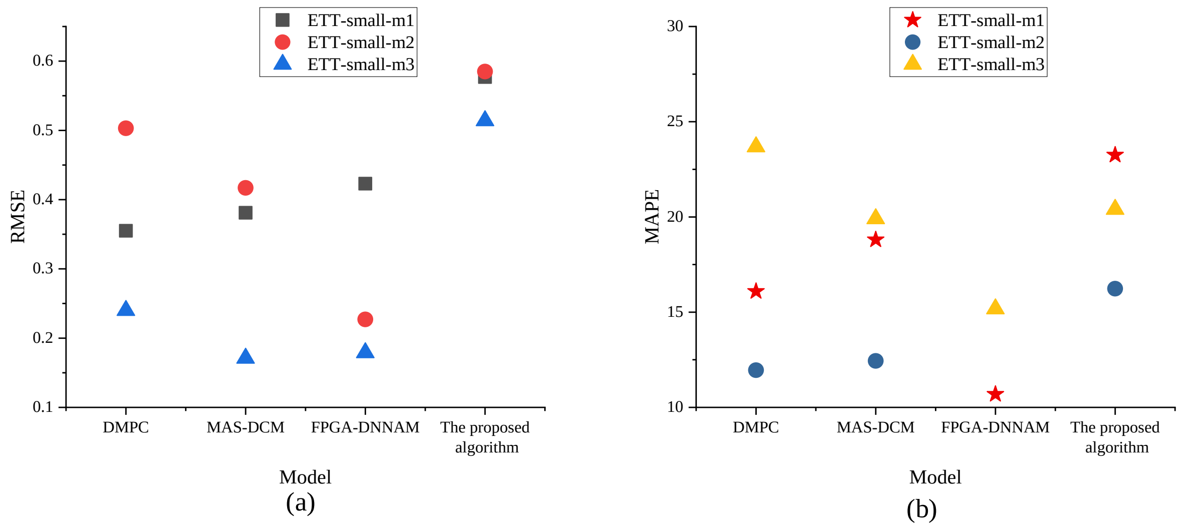

Figure 3: Robustness dimension experimental result ((a): RMSE; (b): MAPE)

The proposed model has RMSEs of 0.577 (ETT-small-m1), 0.585 (ETT-small-m2), and 0.514 (ETT-small-m3). These are higher than FPGA-DNNAM's 0.179 on ETT-small-m3 and MAS-DCM's 0.381 on ETT-small-m1. However, the overall difference with other models is not significant. On the ETT-small-m1 dataset, the proposed model's MAPE is 23.250, slightly higher than FPGA-DNNAM's 10.688 and MAS-DCM's 18.803. On the ETT-small-m2 and ETT-small-m3 datasets, the proposed model achieves MAPEs of 16.234 and 20.401, respectively, outperforming DMPC's 23.685. Overall, the proposed optimized model shows good robustness in both RMSE and MAPE. Despite a slightly higher RMSE due to outliers, it demonstrates reasonable control of MAPE. This indicates the model’s adaptability to complex real-world scenarios. The results for the timeliness dimension are presented in Figure 4.

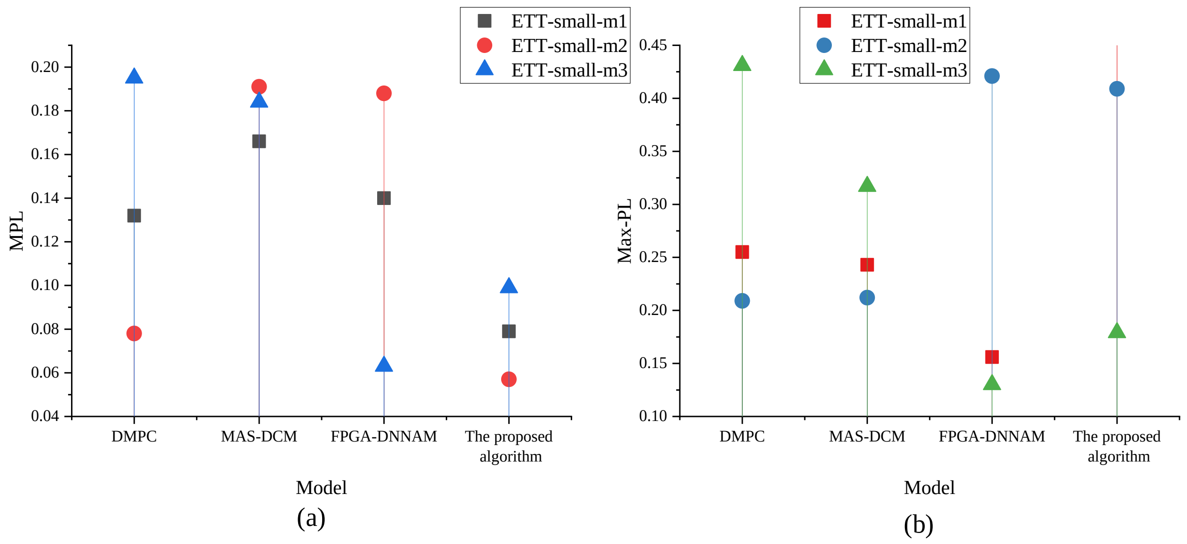

Figure 4: Timeliness Dimension experimental result ((a): MPL; (b): Max-PL)

Figure 4 shows that the proposed optimized model achieves the lowest MPL across all datasets. Specifically, it records scores of 0.079 for ETT-small-m1, 0.057 for ETT-small-m2, and 0.099 for ETT-small-m3. In comparison, the MPL values of DMPC and MAS-DCM both exceed 0.130. Although FPGA-DNNAM achieves a slightly lower latency of 0.063 on the ETT-small-m3 dataset, it remains close to the proposed model. For Max-PL, the proposed model again outperforms the baselines, achieving 0.495 (ETT-small-m1), 0.409 (ETT-small-m2), and 0.179 (ETT-small-m3). These values are consistently lower than DMPC’s 0.431 on ETT-small-m3 and MAS-DCM’s 0.212 on ETT-small-m2. Overall, the proposed model demonstrates excellent timeliness. It effectively controls both average and peak latency, making it highly suitable for DCSs applications where real-time performance is critical. The results of resource efficiency analysis are presented in Figure 5:

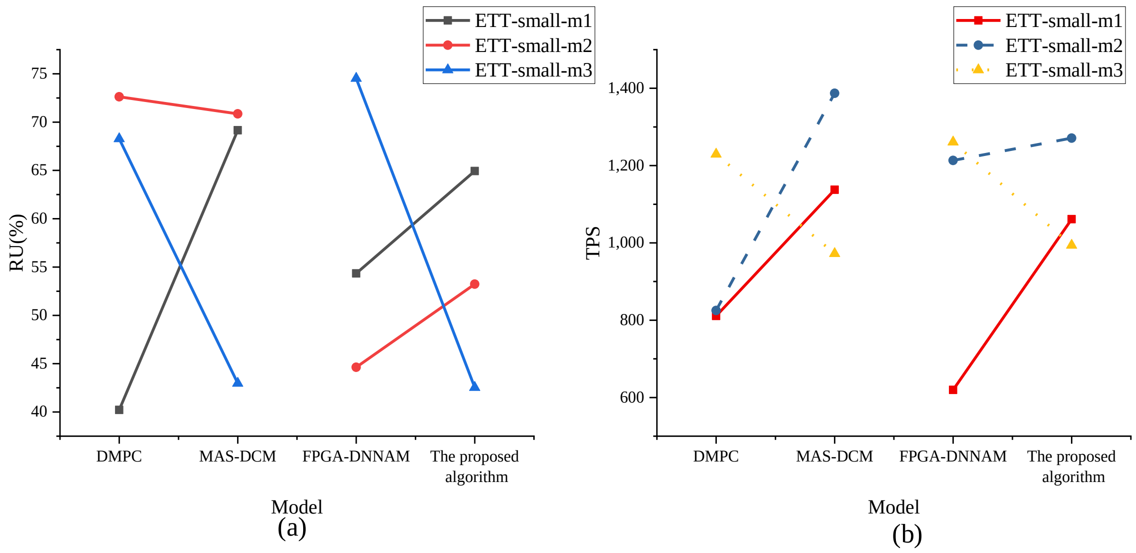

Figure 5: Resource Efficiency Dimension experimental result ((a): Resource Utilization; (b): Throughput)

Figure 5 shows the RU rates for the proposed optimized model across three datasets: 64.932% for ETT-small-m1, 53.236% for ETT-small-m2, and 42.542% for ETT-small-m3. While these rates are slightly higher than FPGA-DNNAM's 44.635% on the ETT-small-m2 dataset, they remain lower than MAS-DCM's 69.160% on ETT-small-m1, indicating a balanced performance in terms of resource usage. The model also excels in processing speed, with the number of samples processed per second reaching 1061.277 (ETT-small-m1), 1270.967 (ETT-small-m2), and 993.796 (ETT-small-m3). These figures significantly surpass DMPC's 810.982 (ETT-small-m1) and FPGA-DNNAM's 1260.785 (ETT-small-m3), demonstrating the model's efficiency in handling high throughput. Overall, the proposed optimized model performs exceptionally well in terms of resource efficiency. It achieves a high processing speed while maintaining effective control over hardware resource consumption, making it suitable for deployment in environments with limited resources but a need for efficient processing.

In terms of key performance dimensions—accuracy, timeliness, and resource efficiency—the model stands out, particularly with notable advantages in metrics such as MSE, MPL, and throughput. While its robustness is slightly lower compared to some other models, it still maintains good stability. This suggests that the proposed model is well-suited for general distributed control tasks, particularly in complex environments that demand high real-time performance and efficiency.

5.2 Testing of system operation effectiveness

In the system performance dimension, this study selects four key indicators: Task Completion Rate (TCR), Average Response Time (ART), Task Failure Rate (TFR), and System Throughput (ST). TCR reflects the proportion of tasks successfully completed within a given period and serves as a fundamental metric for evaluating the stability and controllability of the scheduling system. ART measures the average time from task reception to completion, indicating the system's real-time responsiveness. TFR assesses the proportion of tasks that fail to complete due to scheduling errors, system congestion, or control delays, offering valuable insights for optimizing scheduling strategies. ST quantifies the number of tasks processed by the system per unit of time, representing the system's efficiency under high concurrency. This dimension focuses on the quality of the system's load-handling response during real-world operation and serves as a core metric for assessing improvements in control system efficiency post-deployment. In the system reliability dimension, to evaluate the impact of model deployment on long-term operational stability, four additional indicators are introduced: MTBF, Mean Time to Repair (MTTR), Error Rate (ER), and System Availability (SA). MTBF indicates the average duration the system can operate normally between two consecutive failures—a higher value suggests greater system stability. MTTR measures the average time required for the system to return to normal operation following a failure, reflecting the responsiveness of fault recovery mechanisms. ER captures the proportion of errors arising during operation due to issues such as data transmission failures or module malfunctions, serving as a key metric for system robustness. SA evaluates the percentage of total operational time during which the system remains available, offering a comprehensive measure of system stability and sustained service capability. This dimension emphasizes the system's resilience to anomalies, unexpected failures, and uncertain disturbances during extended operation.

The comparison results of system performance dimensions are revealed in Figure 6:

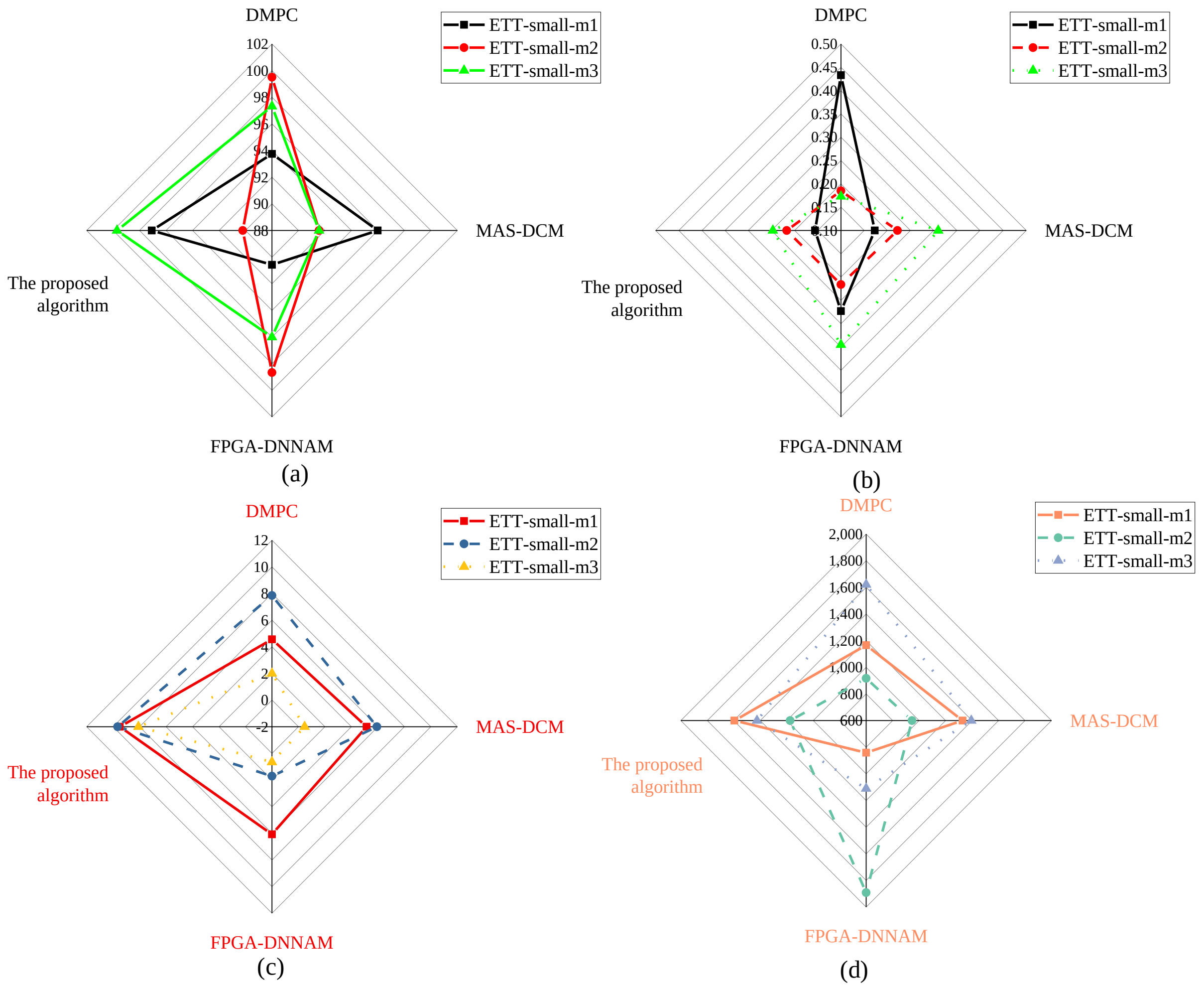

Figure 6: System Performance Dimension experimental result ((a): Task Completion Rate; (b): Average Response Time; (c): Task Failure Rate; (d): System Throughput)

Figure 6 presents the performance of the proposed optimized model in terms of task completion rate, average response time, task failure rate, and system throughput. On the ETT-small-m3 dataset, the model achieves a task completion rate of 99.699%, significantly higher than DMPC (97.320%) and MAS-DCM (91.560%). However, on the ETT-small-m2 dataset, its task completion rate drops to 90.206%, which is lower than FPGA-DNNAM's 98.662%. In terms of average response time, the proposed model performs well across all datasets. It records response times of 0.156 (ETT-small-m1), 0.217 (ETT-small-m2), and 0.247 (ETT-small-m3). Although slightly slower than DMPC's 0.185 on ETT-small-m2, it still outperforms MAS-DCM and FPGA-DNNAM overall. Regarding task failure rate, the model records 9.489% on the ETT-small-m1 dataset. While this is higher than DMPC (4.561%) and FPGA-DNNAM (6.075%), the failure rate improves on ETT-small-m3, decreasing to 8.084%. Despite this progress, further optimization is needed in this area. For system throughput, the proposed model delivers strong results. It achieves 1595.027 tasks per second on the ETT-small-m1 dataset, far exceeding DMPC's 1165.537. On the ETT-small-m3 dataset, throughput reaches 1424.082 tasks per second, slightly surpassing MAS-DCM's 1394.212. In summary, the proposed optimized model excels in task completion rate and system throughput, showing strong suitability for high-efficiency applications. Its average response time also reflects good real-time performance. However, improvements in reducing the task failure rate remain an area for future work. The comparison results for system reliability are illustrated in Figure 7:

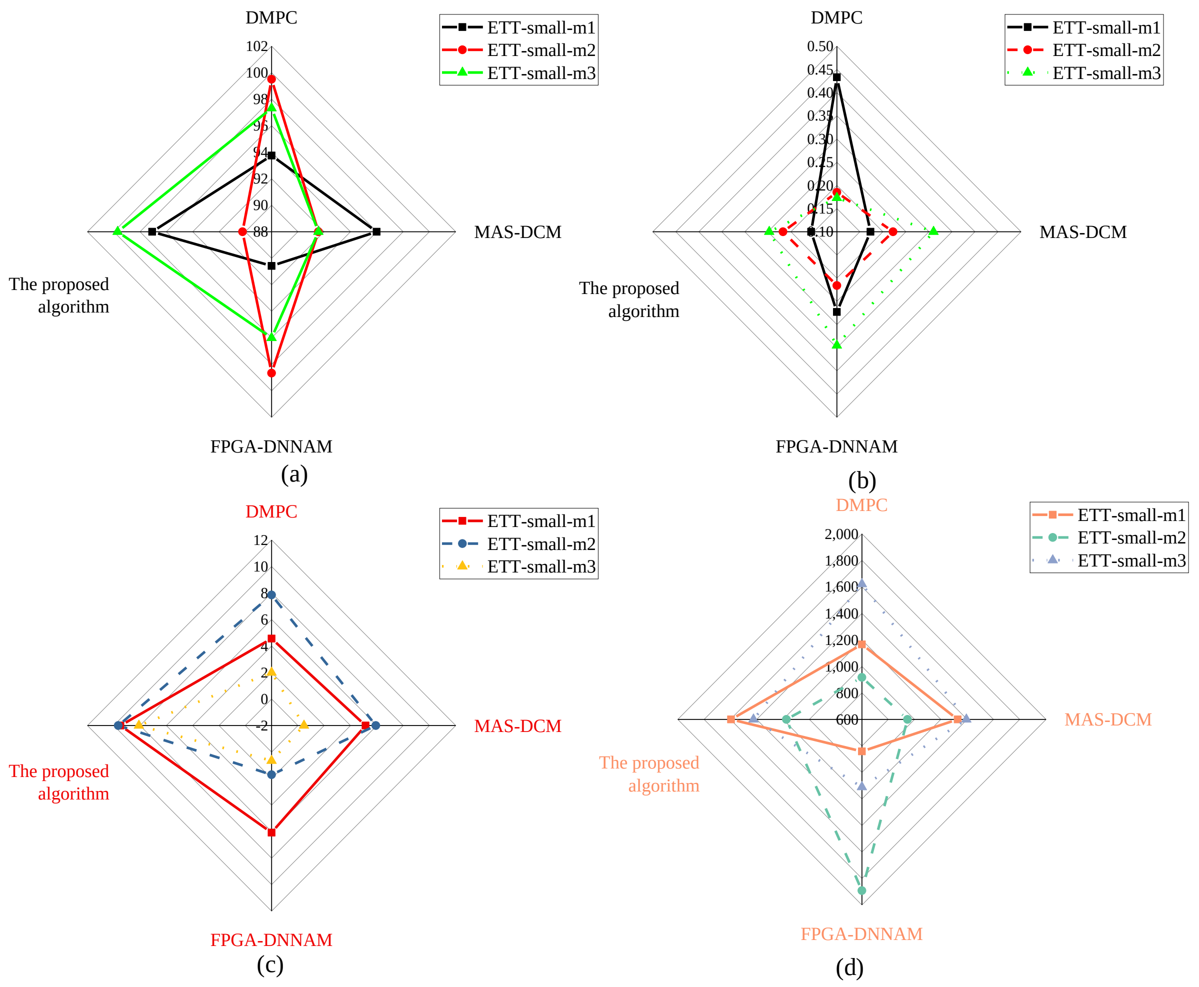

Figure 7: System Reliability Dimension experimental result ((a): Mean Time between Failures; (b): Mean Time to Repair; (c): Error Rate; (d): System Availability)

Figure 7 illustrates the reliability metrics of the proposed optimized model across different datasets. On the ETT-small-m1 dataset, the model achieves a MTBF of 178.393 hours—substantially lower than MAS-DCM's 410.053 hours and DMPC's 318.684 hours. On the ETT-small-m3 dataset, the MTBF increases to 230.132 hours, though it still falls short of the comparative models. The model's MTTR on the ETT-small-m1 dataset is 0.198 hours, which is slightly higher than FPGA-DNNAM's 0.071 hours. However, on the ETT-small-m3 dataset, the MTTR improves to 0.080 hours, aligning closely with other models in this category. In terms of error rate, the proposed model demonstrates strong performance, achieving the lowest value on the ETT-small-m3 dataset at just 0.318%, significantly outperforming FPGA-DNNAM (4.316%) and DMPC (3.534%). On the ETT-small-m1 dataset, the model records an error rate of 3.116%, comparable to MAS-DCM's 3.645%. The proposed model also maintains high availability across all datasets, exceeding 97% in every case. Notably, it reaches 98.855% on the ETT-small-m2 dataset, slightly below MAS-DCM's 99.436%. Overall, the proposed model exhibits excellent performance in terms of error rate and availability, making it well-suited for high-reliability applications. However, there remains considerable room for improvement in MTBF and MTTR. Future work should aim to enhance the system's fault tolerance and recovery efficiency to further bolster its robustness in real-world deployments.

5.3 Ablation study

To evaluate the contribution of key components within the proposed delay prediction optimization model and to further analyze the effectiveness of its internal mechanisms, an ablation study was conducted. By progressively removing core modules, several simplified variants of the model were constructed and evaluated under the same dataset and assessment framework. This approach allowed for a quantitative comparison of each component's impact on model accuracy, timeliness, and system control performance. Using the full model as the baseline, the following three ablated variants were implemented:

1) w/o Quantization: Disables weight quantization, using 32-bit floating-point parameters for computation.
2) w/o Online Update: Disables the online fine-tuning mechanism, maintaining a static model.

3) w/o System Features: Uses only historical delay sequences as input, removing all system context features.

The experimental results are summarized in Table 3:

Table 3: Ablation study results

Model Variant	MAE	MSE	Avg. Prediction Latency	Task Completion Rate (%)
Full Model	0.233	0.105	0.156	98.64
w/o Quantization	0.251	0.120	0.171	97.15
w/o Online Update	0.263	0.132	0.162	96.38
w/o System Features	0.285	0.149	0.165	94.72

As shown in Table 3, the full model performs best across all four evaluation metrics. Removing the quantization mechanism results in a slight decrease in accuracy but leads to a noticeable increase in inference time, highlighting the efficiency benefits of quantization for edge computing deployment. Disabling the online update mechanism leads to reduced model adaptability. The Task Completion Rate (TCR) drops by more than two percentage points, from 98.64% to 96.38%. This highlights the importance of feedback-based error correction in dynamic environments. The most significant performance degradation occurs when system context features are removed. In this variant, MAE increases to 0.285 and TCR falls to 94.72%, indicating that scheduling context is a key input for improving prediction accuracy and ensuring successful task execution.

6 Discussion

The FPGA-accelerated LSTM-based latency prediction model proposed in this study demonstrates strong performance across multiple dimensions, particularly excelling in prediction accuracy, task completion rate, and inference latency when compared to baseline models. However, a closer examination of the experimental results reveals several performance trade-offs and limitations that warrant further discussion and provide guidance for future optimization.

From a structural perspective, although the model achieves high prediction accuracy and low inference latency across various datasets, it performs slightly less robustly than some deep ensemble-based models in terms of robustness metrics. This may be attributed to the LSTM's limited adaptability to abnormal fluctuations and sudden load disturbances. Traditional control optimization models, which incorporate hard constraint mechanisms, tend to handle extreme variations more effectively. In contrast, while neural networks are well-suited for learning regular patterns, they may exhibit instability when handling boundary conditions or outlier predictions. Performance discrepancies across datasets also highlight the model's limitations. For instance, the model performs consistently better on ETT-small-m1 than on ETT-small-m3, likely due to the former's stronger periodicity and more controlled noise characteristics, which are advantageous for temporal modeling. In contrast, ETT-small-m3 may involve higher unpredictability in data sources, task switching frequency, or external disturbances, increasing the difficulty of model training. Future work could incorporate data augmentation techniques, attention mechanisms, or more sophisticated feature adaptation strategies to enhance the model's generalization across varied data distributions.

On the system implementation side, the FPGA acceleration scheme offers notable advantages over traditional software-based approaches. Customized data paths and parallel hardware structures significantly reduce inference latency, while resource control and low power consumption make the solution well-suited for deployment at industrial DCS edge nodes. Compared with processor-based solutions, the FPGA platform delivers superior task throughput and latency control within a given time frame, offering robust real-time guarantees—particularly important in high-frequency scheduling scenarios. However, FPGA deployment also introduces challenges in programmability and debugging complexity. Future research should explore the integration of automated deployment workflows with interpretability analysis tools. Regarding the quantization strategy, while 8-bit fixed-point quantization effectively reduces hardware resource consumption, its impact on model performance and RU still requires more systematic evaluation. Preliminary tests suggest that aggressive quantization may slightly degrade prediction accuracy, but it has a positive effect on throughput and on-chip resource availability. Balancing high-level compression with minimal loss in prediction precision remains a key direction for improving deployment efficiency in future work.

Furthermore, the experiments reveal inherent trade-offs between different performance dimensions. For instance, while the integration of an online update mechanism improves prediction accuracy and task completion rate, it also introduces additional computational overhead, which can negatively impact real-time performance. Conversely, disabling online learning accelerates inference but compromises the system's adaptability. Similarly, the task failure rate, a key indicator of system reliability, increases when the model lacks sufficient fault tolerance under high-load or complex scenarios. In real-world DCSs, such shortcomings may lead to scheduling delays or control disruptions, highlighting the importance of closely monitoring this metric.

Overall, the proposed method outperforms existing state-of-the-art (SOTA) models across most evaluation dimensions, with particularly notable advantages in inference latency and system throughput. However, there remains room for improvement in terms of robustness to anomalous predictions, cross-scenario adaptability, and the system's capacity for dynamic reconfiguration. Future work may explore the incorporation of attention mechanisms, heterogeneous input encoding strategies, and adaptive quantization techniques to better balance performance and deployability.

7 Conclusion

This study is the first to deeply integrate FPGA hardware acceleration with neural network–based latency prediction, resulting in a highly efficient predictive model. By offloading neural network inference to FPGA hardware, the model significantly reduces computational overhead during the prediction process, thereby enhancing the real-time performance of DCSs. This approach addresses the low efficiency typically seen in traditional neural network models when deployed on embedded platforms, offering a novel solution for real-time control tasks. The model's performance was rigorously evaluated across multiple dimensions—accuracy, robustness, timeliness, and resource efficiency. Experimental results demonstrate that the proposed model consistently outperforms baseline approaches, particularly in system throughput, average prediction latency, and task completion rate. These findings confirm the model's effectiveness and provide valuable performance benchmarks for future research. Despite these advances, certain limitations remain. The model performs well on the ETT-small dataset but has not been extensively validated in other scenarios, especially those involving unstructured or multimodal data. Future research could enhance the model's flexibility by incorporating multimodal fusion techniques, enabling it to integrate latency predictions with data from videos, sensors, and images for broader applicability in real-world settings. Additionally, the model shows some sensitivity under extreme conditions such as high load or the presence of outliers, where minor instability in key metrics was observed. To address these challenges, future work could explore adaptive thresholding, dynamic task allocation, and fault-tolerant mechanisms. Such enhancements would further improve the system's reliability and efficiency, especially in complex or unpredictable environments.

References

[1] Fei M, Zhang Z, Zhao W, et al. Optimal power distribution control in modular power architecture using hydraulic free piston engines. *Applied Energy*, 2024, 358(7): 122540. https://doi.org/10.1016/j.apenergy.2024.122540

[2] Kumar K, Pande S V, Kumar T C A, et al. Intelligent controller design and fault prediction using machine learning model. *International Transactions on Electrical Energy Systems*, 2023, 2023(1): 1056387. https://doi.org/10.1155/2023/1056387

[3] Liu G P. Tracking control of multi-agent systems using a networked predictive PID tracking scheme. *IEEE/CAA Journal of Automatica Sinica*, 2023, 10(1): 216–225. https://doi.org/10.1109/JAS.2023.120430

[4] Louati H, Niazi A U K, Dalam M E E, et al. Optimizing control efficiency in discrete-time multi-agent systems via event-triggered containment techniques combining disturbance handling and input delay management. *Heliyon*, 2024, 10(14): e26767. https://doi.org/10.1016/j.heliyon.2024.e26767

[5] Bououden S, Allouani F, Abboudi A, et al. Observer-based robust fault predictive control for wind turbine time-delay systems with sensor and actuator faults. *Energies*, 2023, 16(2): 858. https://doi.org/10.3390/en16020858

[6] Gams M, Kolenik T. Relations between electronics, artificial intelligence and information society through information society rules. *Electronics*, 2021, 10(4): 514. https://doi.org/10.3390/electronics10040514

[7] Ilyushin Y V, Asadulagi M A M. Development of a distributed control system for the hydrodynamic processes of aquifers, taking into account stochastic disturbing factors. *Water*, 2023, 15(4): 770. https://doi.org/10.3390/w15040770

[8] Nie Q, Tang D, Liu C, et al. A multi-agent and cloud-edge orchestration framework of digital twin for distributed production control. *Robotics and Computer-Integrated Manufacturing*, 2023, 82(4): 102543. https://doi.org/10.1016/j.rcim.2023.102543

[9] Dai X, Lederer A, Yang Z, et al. Can learning deteriorate control? Analyzing computational delays in Gaussian process-based event-triggered online learning[C]// *Learning for Dynamics and Control Conference*, PMLR, 2023: 445–457.

[10] Caballero-Águila R, Linares-Pérez J. Distributed fusion filtering for uncertain systems with coupled noises, random delays and packet loss prediction compensation. *International Journal of Systems Science*, 2023, 54(2): 371–390. https://doi.org/10.1080/00207721.2022.2126020

[11] Tiong K Y, Ma Z, Palmqvist C W. A review of data-driven approaches to predict train delays. *Transportation Research Part C: Emerging Technologies*, 2023, 148(21): 104027. https://doi.org/10.1016/j.trc.2023.104027

[12] Sridevi K, Saifulla M A. LBABC: Distributed controller load balancing using artificial bee colony optimization in an SDN. *Peer-to-Peer Networking and Applications*, 2023, 16(2): 947–957. https://doi.org/10.1007/s12083-022-01387-1

[13] Herrera M, Benítez D, Pérez-Pérez N, et al. Hybrid controller based on numerical methods for chemical processes with a long time delay. *ACS Omega*, 2023, 8(28): 25236–25253. https://doi.org/10.1021/acsomega.3c02747

[14] Olabi A G, Abdelghafar A A, Maghrabie H M, et al. Application of artificial intelligence for prediction, optimization, and control of thermal energy storage systems. *Thermal Science and Engineering Progress*, 2023, 39(21): 101730. https://doi.org/10.1016/j.tsep.2023.101730

[15] Jiang J, Han C, Zhao W X, et al. Pdformer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, 37(4): 4365–4373.

[16] Mbungu N T, Ismail A A, AlShabi M, et al. Control and estimation techniques applied to smart microgrids: A review. *Renewable and Sustainable Energy Reviews*, 2023, 179(32): 113251. https://doi.org/10.1016/j.rser.2023.113251

[17] Balali Y, Chong A, Busch A, et al. Energy modelling and control of building heating and cooling systems with data-driven and hybrid models—A review. *Renewable and Sustainable Energy Reviews*, 2023, 183(3): 113496. https://doi.org/10.1016/j.rser.2023.113496

[18] Wang Y A, Shen B, Zou L, et al. A survey on recent advances in distributed filtering over sensor networks subject to communication constraints. *International Journal of Network Dynamics and Intelligence*, 2023, 6(2): 100007. https://doi.org/10.1016/j.indani.2023.100007

[19] Modu B, Abdullah M P, Sanusi M A, et al. DC-based microgrid: Topologies, control schemes, and implementations. *Alexandria Engineering Journal*, 2023, 70(11): 61–92. https://doi.org/10.1016/j.aej.2023.03.015

[20] Asha A, Arunachalam R, Poonguzhali I, et al. Optimized RNN-based performance prediction of IoT and WSN-oriented smart city application using improved honey badger algorithm. *Measurement*, 2023, 210(13): 112505. https://doi.org/10.1016/j.measurement.2023.112505

[21] Duan G. Fully actuated system approaches for continuous-time delay systems: Part 1. Systems with state delays only. *Science China Information Sciences*, 2023, 66(1): 112201. https://doi.org/10.1007/s11432-021-3411-y

[22] Ghiasi M, Niknam T, Wang Z, et al. A comprehensive review of cyber-attacks and defense mechanisms for improving security in smart grid energy systems: Past, present and future. *Electric Power Systems Research*, 2023, 215(11): 108975. https://doi.org/10.1016/j.epsr.2022.108975

[23] Ashok Babu P, Mazher Iqbal J L, Siva Priyanka S, et al. Power control and optimization for power loss reduction using deep learning in microgrid systems. *Electric Power Components and Systems*, 2024, 52(2): 219–232. https://doi.org/10.1080/15325008.2023.2239573

[24] Simonetti F, D'Innocenzo A, Cecati C. Neural network model-predictive control for CHB converters with FPGA implementation. *IEEE Transactions on Industrial Informatics*, 2023, 19(9): 9691–9702. https://doi.org/10.1109/TII.2022.3154581

[25] Abbasi M, Abbasi E, Li L, et al. Review on the microgrid concept, structures, components, communication systems, and control methods. *Energies*, 2023, 16(1): 484. https://doi.org/10.3390/en16010484

[26] Hu J, Li J, Liu G P, et al. Optimized distributed filtering for time-varying saturated stochastic systems with energy harvesting sensors over sensor networks. *IEEE Transactions on Signal and Information Processing over Networks*, 2023, 9(3): 412–426. https://doi.org/10.1109/TSIPN.2023.3268275

[27] Rajagopal A, Chitraganti S. State estimation and control for networked control systems in the presence of correlated packet drops. *International Journal of Systems Science*, 2023, 54(11): 2352–2365. https://doi.org/10.1080/00207721.2023.2177829